Theory Interpretation in Simple Type Theory*

William M. Farmer

The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420, USA

farmer@mitre.org

March 28, 1994

Abstract. Theory interpretation is a logical technique for relating one axiomatic theory to another with important applications in mathematics and computer science as well as in logic itself. This paper presents a method for theory interpretation in a version of simple type theory, called LUTINS, which admits partial functions and subtypes. The method is patterned on the standard approach to theory interpretation in first-order logic. Although the method is based on a nonclassical version of simple type theory, it is intended as a guide for theory interpretation in classical simple type theories as well as in predicate logics with partial functions.

1 Introduction

Theory interpretation—in which one theory is interpreted in another via a syntactic mapping—is a fundamental logical technique which has important applications in mathematics and computer science as well as in logic itself. An *interpretation*[1] of a theory[2] T_1 in a theory T_2 is a mapping from the expressions of T_1 to the expressions of T_2 which preserves the validity of sentences. (T_1 and T_2 are called the *source theory* and the *target theory* of the interpretation, respectively.) In logic, interpretations are used to prove metamathematical properties about theories and to compare theories in terms of their "strength". In mathematics, theorems and problems are transported from one context to another via interpretations. In computer science, interpretations are a rigorous tool for documenting and verifying that one system specification is a refinement of another.

* Supported by the MITRE-Sponsored Research program. To appear in the proceedings of the International Workshop on Higher Order Algebra, Logic and Term Rewriting (HOA '93, Amsterdam, The Netherlands, September 1993), *Lecture Notes in Computer Science*, Springer-Verlag, 1994.

[1] Theory interpretations of this kind are also called *translations*, *theory morphisms*, *immersions*, and *realizations*.

[2] We take a *theory* to be a set of sentences in a formal language (that is not necessarily closed under logical consequence). The sentences are called the *axioms* of the theory.

Until recently, interpretations have been almost exclusively employed by theoreticians. However, implementors are now discovering that interpretations are useful for organizing and supporting mathematical reasoning in automated reasoning systems such as mechanical theorem provers and computer system specification and verification environments. Interpretations are used extensively with success in the IMPS Interactive Mathematical Proof System [10, 11, 12]. They are also a fundamental component in the following programming and verification environments: EHDM [27], m-EVES [5] and EVES [6], IOTA [24], and OBJ3 [14].

Theory interpretation has primarily been studied and applied in the context of first-order predicate logic. Logic textbooks like Enderton [7], Monk [22], and Shoenfield [28] present a fairly standard approach to theory interpretation in first-order logic. The approach revolves around a special class of interpretations that are well behaved both syntactically and semantically. Suppose Φ is an interpretation of T_1 in T_2 which is in this class. Then Φ will be a kind of homomorphism which preserves the structure of terms and formulas and which is completely determined by how it associates the sorts (if there are any) and constants of T_1 with objects of T_2. Moreover, Φ will define a way of extracting a model for T_1 from any model for T_2.

Although there is a wealth of writing on theory interpretation in first-order logic, the subject is only beginning to be seriously explored in higher-order logic (and type theory) [9, 17, 36]. There are, however, at least two good reasons to study theory interpretation in higher-order logic. First, higher-order logic is becoming increasingly important in computer science and mechanized mathematics. Second, since higher-order logic is much more expressive than first-order logic, the space of interpretations is much richer in higher-order logic than in first-order logic. This means that some techniques based on theory interpretation are more powerful in a higher-order logic than in a first-order logic (e.g., the technique of verifying that a theory T' is a model conservative extension of T by exhibiting an interpretation of T' in T which fixes T).

The most well-known and widely used form of higher-order logic is simple type theory [4, 1]. Since it has built-in support for functions—a hierarchy of function types, full quantification over functions, and (usually) λ-notation for specifying functions—it is a convenient logic for formalizing mathematics. For this reason, it is the logical basis for several automated reasoning systems, including EHDM, HOL [15], IMPS, Isabelle [25], PVS [26], and TPS [2]. In spite of its popularity and utility, there is not a well-developed approach to theory interpretation in simple type theory.

The goal of this paper is to develop a method for theory interpretation in simple type theory patterned on the standard first-order approach. We want the method to handle interpretations in which a base type (i.e., a type of individuals) of the source theory can be associated with either a (possibly higher-order) *type* or *subtype* of the target theory.

In first-order logic, an interpretation which associates base types with types is merely an interpretation which associates the universe (i.e., the implicit type of individuals) of the source theory with the universe of the target theory. An interpretation of this kind does not alter the quantifiers in expressions of the source theory. An interpretation which associates base types with subtypes is one which associates the universe of the source theory with a unary predicate of the target theory. An interpretation of this kind "relativizes" the quantifiers in expressions of the source theory. For example, if Φ is an interpretation which associates the universe of the source theory with the predicate φ, then

$$\Phi((\forall x)\psi) = (\forall x)(\varphi(x) \rightarrow \Phi(\psi)).$$

Many natural theory interpretations associate a base type with a subtype (i.e., part of a type). For example, suppose G is a theory of an abstract group in which α is a base type denoting the set of group elements; F is a theory of an abstract field in which β is a base type denoting the set of field elements; and Φ is the interpretation of G in F in which the group structure of G is "interpreted" as the structure of the multiplicative group of F. Then Φ would associate α with the subtype of β consisting of the nonzero field elements. Moreover, the most natural translation of the group operation of G via Φ would be an expression denoting the multiplication operation of F restricted to the nonzero field elements. Thus we see that associating base types with subtypes leads to functions with restricted domains. (This example is worked out in detail in Section 7.)

If only interpretations which associate base types with full types are considered, it is easy to lift the first-order notion of a theory interpretation to simple type theory. On the other hand, associating base types with subtypes is messy in simple type theory since one must deal with functions with restricted domains, as we have seen above. Restricting the domain of a function is unproblematic in informal mathematics, but there is no completely satisfactory way that it can be done in classical predicate logic since expressions cannot directly denote partial functions. (See [8] for a discussion on the various ways of dealing with partial functions in predicate logic.) In first-order logic, partial functions are avoided by relativizing quantifiers. This approach would be more complicated in simple type theory because more than just quantifiers would have to be relativized; in particular, all predicates on functions (such as those corresponding to universal and existential quantification) would have to be relativized.

Our method for theory interpretation is formulated in a version of simple type theory, called LUTINS[3] [8, 9, 16], which supports both partial functions and subtypes. We have chosen LUTINS over a classical simple type for three *pragmatic* reasons. First, as we have pointed out, partial functions naturally arise from interpretations that associate base types with subtypes. Consequently, interpretations of this kind can be formalized more directly in a logic which admits partial functions like LUTINS. Second, since LUTINS contains subtypes, an interpretation in LUTINS does not have to relativize quantifiers and other variable binders, provided appropriate subtypes are defined. Finally, as the logic of the IMPS interactive theorem proving system, LUTINS has been implemented and rigorously and extensively tested [12]. It is clearly an effective logic for formalizing a wide range of mathematics. Although the method is based on a nonclassical

[3] Pronounced as the word in French.

form of simple type theory, we expect it to be useful as a guide for theory interpretation in classical simple type theories as well as in predicate logics which admit partial functions.

The paper is organized as follows. An overview and discussion via examples of the standard approach to theory interpretation in first-order logic is given in Sections 2 and 3. Section 4 gives a quick introduction to $\mathbf{PF}^*$, an austere version of simple type theory with partial functions and subtypes on which LUTINS is based. The syntax and semantics of LUTINS are then presented in Section 5. The notion of a theory interpretation in LUTINS is defined in Section 6. Section 7 contains some examples in LUTINS of interpretations of groups in fields. The interpretation and relative satisfiability theorems for LUTINS are proved in Section 8. And a brief conclusion is found in Section 9.

Comparisons between our method of theory interpretation and the standard approach in first-order logic are made at several places in the paper.

2 Theory Interpretation in First-Order Logic

This section presents an outline of the standard approach to theory interpretation in first-order logic [7, 22, 28]. For the most part, we shall adopt in this section the definitions and notation of first-order logic (with equality) presented in [3]. An *expression* of a first-order language $\mathcal{L}$ or theory T is a term or a formula of $\mathcal{L}$ or T. An *n-ary expression function* is a λ-expression of the form $\lambda\{x_1, \ldots, x_n \,.\, E\}$ where E is an expression. Let $\theta = \lambda\{x_1, \ldots, x_n \,.\, E\}$ be an expression function. θ is a *term* [respectively, *formula*] *function* if E is a term [respectively, formula]. Given terms $t_1, \ldots, t_n$, $\theta(t_1, \ldots, t_n)$ denotes the result of simultaneously substituting t_i for all free occurrences of x_i in E, for all i with $1 \leq i \leq n$.

Let T_i be a first-order theory for $i = 1, 2$. A *standard translation from T_1 to T_2* is a pair (U, ν) where U is a closed formula function of the form $\lambda\{x \,.\, \varphi\}$ which represents a unary predicate and ν is a function from the nonlogical constants of T_1 to the nonlogical constants, expressions, and expression functions of T_2 such that:

1. If c is an individual constant symbol of T_1, then $\nu(c)$ is either an individual constant symbol or a closed term.
2. If F is an n-ary function symbol of T_1, then $\nu(F)$ is either an n-ary function symbol or a closed n-ary term function.
3. If P is an n-ary relation symbol of T_1, then $\nu(P)$ is either an n-ary relation symbol or a closed n-ary formula function.
4. $\nu(\equiv) = \equiv$.[4]

Let $\Phi = (U, \nu)$ be a standard translation from T_1 to T_2 throughout the rest of this section. For an expression E of T_1, the *translation of E via Φ*, written $\Phi(E)$, is the expression of T_2 defined inductively by:

[4] In [3], the binary relation symbol $\equiv$ denotes the equality relation.

1. $\Phi(x) = x$, if x is a variable.
2. $\Phi(c) = \nu(c)$, if c is an individual constant symbol.
3. $\Phi(S(t_1, \ldots, t_n)) = \nu(S)(\Phi(t_1), \ldots, \Phi(t_n))$, if S is an n-ary function or relation symbol.
4. $\Phi(\neg\varphi) = \neg\Phi(\varphi)$.
5. $\Phi(\varphi \,\square\, \psi) = \Phi(\varphi) \,\square\, \Phi(\psi)$, if $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
6. $\Phi((\square x)\varphi) = (\square x)\Phi(\varphi)$, if $\square \in \{\forall, \exists\}$ and $U = \lambda\{x \,.\, x \equiv x\}$.
7. $\Phi((\square x)\varphi) = \begin{cases} (\forall x)(U(x) \rightarrow \Phi(\varphi)) & \text{if } \square = \forall \\ (\exists x)(U(x) \wedge \Phi(\varphi)) & \text{if } \square = \exists \end{cases}$
   if $U \neq \lambda\{x \,.\, x \equiv x\}$.

A standard translation thus associates the universe of its source theory with a closed unary predicate of its target theory; the nonlogical constants of its source theory with closed expressions (of appropriate "type") of its target theory; and the variables and logical connectives with themselves. The quantifiers are relativized to the unary predicate if it is not $\lambda\{x \,.\, x \equiv x\}$. Except for the relativization of quantifiers, a standard translation preserves the structure of first-order syntax. Hence, a standard translation can be viewed as a homomorphism from the expressions of its source theory to the expressions of its target theory.

Φ is a *standard interpretation of* T_1 *in* T_2 if $\Phi(\varphi)$ is valid in T_2 for each sentence φ which is valid in T_1. That is, Φ is an interpretation if it maps valid sentences to valid sentences. The theorem below gives a sufficient condition for a standard translation to be a standard interpretation.

An *obligation* of Φ is any one of the following sentences of T_2:

1. $\Phi(\varphi)$ for each axiom φ of T_1.
2. $(\exists x)U(x)$.
3. $\Phi((\exists y)c \equiv y)$ for each individual constant symbol c of T_1.
4. $\Phi((\forall x_1 \cdots x_n)(\exists y)F(x_1, \ldots, x_n) \equiv y)$ for each function symbol F of T_1.

The four kinds of obligations are called, in order, *axiom*, *universe nonemptiness*, *individual constant symbol*, and *function symbol* obligations. The meaning of an individual constant symbol obligation is that the interpretation of the universe contains the interpretation of the individual constant symbol, and the meaning of a function symbol obligation is that the interpretation of the universe is closed under the interpretation of the function symbol. Note: The last three kinds of obligations are trivially valid in T_2 if $U = \lambda\{x \,.\, x \equiv x\}$.

Theorem 2.1 (Standard Interpretation Theorem) *A standard translation from T_1 to T_2 is a standard interpretation if each of its obligations is valid in T_2.*

T_1 is *interpretable in* T_2 *(in the standard sense)* if there is a standard interpretation of T_1 in T_2. The next theorem is the most important consequence of interpretability.

Theorem 2.2 (Standard Relative Satisfiability) *If T_1 is interpretable in T_2 and T_2 is satisfiable, then T_1 is satisfiable.*

The key idea in the proof of this theorem is to use the standard interpretation of T_1 in T_2 to extract a model of T_1 from a model of T_2. The Standard Interpretation Theorem and the Standard Relative Satisfiability theorem are the chief theorems of the standard approach to theory interpretation in first-order logic.

By virtue of being a validity preserving homomorphism, a standard interpretation syntactically and semantically embeds its source theory in its target theory. Standard interpretations are used to compare the strength of theories: T_2 is at least as strong as T_1, if T_1 is interpretable in T_2. Also, standard interpretations have long been used in logic to prove metamathematical properties about first-order theories, mainly relative consistency, decidability, and undecidability. For example, the classic work of Tarski, Mostowski, and Robinson [32] illustrates how the undecidability of T_1 can be reduced to the undecidability of T_2 by constructing an appropriate standard interpretation of T_2 in T_1. For other references on the theory and use of standard interpretations, see [13, 23, 30, 31, 34].

3 Some Simple Examples

This section contains three examples of standard first-order interpretations. Although the examples are very simple, they illustrate some of the power and versatility of theory interpretation.

The first example is an interpretation of a theory of an abstract nonstrict partial order in a theory of an abstract strict total order.

Example 3.1 Let PO be the theory consisting of the following three sentences in the first-order language of a binary relation symbol $\leq$:

1. *Reflexivity.* $(\forall x)(x \leq x)$.
2. *Transitivity.* $(\forall xyz)((x \leq y \wedge y \leq z) \rightarrow x \leq z)$.
3. *Antisymmetry.* $(\forall xy)((x \leq y \wedge y \leq x) \rightarrow x \equiv y)$.

PO clearly specifies $\leq$ to be a nonstrict partial order.

Similarly, let TO be the theory consisting of the following three sentences in the first-order language of a binary relation symbol $<$:

1. *Irreflexivity.* $(\forall x)\neg(x < x)$.
2. *Transitivity.* $(\forall xyz)((x < y \wedge y < z) \rightarrow x < z)$.
3. *Trichotomy.* $(\forall xy)(x < y \vee y < x \vee x \equiv y)$.

TO clearly specifies $<$ to be a strict total order.

Let Φ_1 be the standard translation (U_1, ν_1) from PO to TO where $U_1 = \lambda\{x \,.\, x \equiv x\}$ and $\nu_1(\leq) = \lambda\{x, y \,.\, (x < y \vee x \equiv y)\}$. Φ_1 is clearly an interpretation of PO in TO by the Standard Interpretation Theorem.□

Φ_1 links two *different* axiomatizations—different axioms in different formal languages—of the same mathematical domain (i.e., the notion of an order relation). Via Φ_1, theorems proved in PO about the nonstrict partial order $\leq$ can be transformed into corresponding theorems in TO about the strict total order $<$. In other words, Φ_1 serves as a conduit through which theorems can be freely "transported" from one mathematical formalization to another. Interpretations are used in this manner, at least informally, in many areas of mathematics to transport theorems from abstract to more concrete contexts or to move a problem to a more convenient setting (for examples, see [18]). Moreover, interpretations of this sort are fundamental to the "little theories" version of the axiom method in which mathematical reasoning is performed over a network of theories linked by interpretations—instead of entirely within one single "big theory" such as Zermelo-Fraenkel set theory. (In [11], we describe the little theories approach and argue in favor of its implementation in mechanical theorem provers.)

Φ_1 also establishes that TO is a "refinement" of PO. Actually, TO refines PO in two ways: (1) the models of TO are a special kind of partial order, namely a total order, and (2) the atomic relation $\leq$ is decomposed into a compound relation built from $<$ and $\equiv$ via Φ_1. (This latter sort of refinement is much like a definition done in reverse.) Refinement is a basic technique in computer science for specifying and building computer systems. Theory interpretation is an excellent tool for rigorizing computer system development based on refinement (for examples, see [20, 21, 29, 33, 35]).

The interpretation in the next example formalizes the symmetry between left and right multiplication in a monoid.

Example 3.2 Let M be the theory consisting of the following three sentences in the first-order language of a binary function constant $*$ and an individual constant e:

1. *Associativity.* $(\forall xyz)((x * y) * z \equiv x * (y * z))$.
2. *Left Identity.* $(\forall x)(e * x \equiv x)$.
3. *Right Identity.* $(\forall x)(x * e \equiv x)$.

M is a very standard formulation of a theory of an abstract monoid.

Now let Φ_2 be the standard translation (U_2, ν_2) from M to itself where $U_2 = \lambda\{x \,.\, x \equiv x\}$, $\nu_2(*) = \lambda\{x, y \,.\, y * x\}$, and $\nu_2(e) = e$. Φ_2 is a homomorphism from the expressions (terms and formulas) of M to the expressions of M, but (unlike Φ_1) Φ_2 alters terms. Each expression A of M is mapped by Φ_2 to an expression $\Phi_2(A)$ which is symmetric to A with respect to reversing the argument order of $*$. For instance, Φ_2 maps the Left Identity axiom to the Right Identity axiom and vice versa. Consequently, Φ_1 is an interpretation of M in itself by the Standard Interpretation Theorem.$\Box$

Interpretations like Φ_2 allow one to reason "by symmetry" in mechanized mathematics systems. For an illustration, suppose A and B are two sentences we want to prove which are symmetric to each other in some way. In informal mathematics, we would first construct a proof of A, and then we would prove

B by simply noting that there is a proof of B that is symmetric to the proof of A. We would not have bothered to construct this proof of B, for nothing new would be learned. In a mechanized mathematics system with interpretations, after proving A, we would find an interpretation which formalizes the symmetry between A and B and which maps A to B. Just as in informal mathematics, we would not bother to produce a proof of B, but we would know that B is valid because it is the image of a theorem by an interpretation. Moreover, the verification that the mapping is really an interpretation would effectively be a verification that this particular kind of reasoning by symmetry is valid.

The interpretation in the last example associates the universe of an abstract monoid with the singleton set of the identity element.

Example 3.3 Let Φ_3 be the standard translation (U_3, ν_3) from M to itself where $U_3 = \lambda\{x \,.\, x \equiv e\}$, $\nu(*) = *$, and $\nu_3(e) = e$. Since $(\{e\}, *, e)$ has the structure of a monoid, it is easy to see that Φ_3 is an interpretation of M in itself.$\Box$

Since U_3 denotes a proper subset of the universe of M, Φ_3 relativizes quantifiers. For example,

$$\Phi_3((\forall x)(e * x \equiv x)) = (\forall x)(x \equiv e \rightarrow e * x \equiv x).$$

Also, notice that Φ_3 fixes the primitive constants of M (i.e., $*$ and e). The act of verifying that Φ_3 is an interpretation is equivalent to proving that $\{e\}$ is a submonoid of M. A standard interpretation that fixes the primitive constants of the source theory but restricts the universe of the source theory is called a *relativization*. Relativizations are commonly used in set theory to establish relative consistency [19].

4 PF*

PF* is a version of simple type theory with partial functions and subtypes. This section gives a quick introduction to the syntax and semantics of **PF***. The next section introduces a "user-friendly" version of **PF*** called LUTINS. See [9] for the full, definitive presentation of **PF***.

Preliminary Definitions

We begin by defining the machinery we will use to specify the various symbols of a **PF*** language. In particular, the set of type or subtype symbols built from a set S of base type or subtype symbols will be the set $\Omega(S)$ defined below.

Let S be a set of symbols containing the symbol $*$. $\Omega(S)$ is the set defined inductively by:

1. $S \subseteq \Omega(S)$.
2. If $\alpha_1, \ldots, \alpha_n, \alpha_{n+1} \in \Omega(S)$ $(n \geq 1)$, then $[\alpha_1, \ldots, \alpha_n, \alpha_{n+1}] \in \Omega(S)$.

$\Omega^*(S)$ is the subset of $\Omega(S)$ defined inductively by:

1. $* \in \Omega^*(S)$.
2. If $\alpha_1, \ldots, \alpha_n \in \Omega(S)$ and $\alpha_{n+1} \in \Omega^*(S)$ $(n \geq 1)$, then $[\alpha_1, \ldots, \alpha_n, \alpha_{n+1}] \in \Omega^*(S)$.

Given a total function $f : S \to \Omega(S)$, $\preceq_f$ is the smallest binary relation on $\Omega(S)$ such that:

1. If $\alpha \in S$, then $\alpha \preceq_f f(\alpha)$.
2. $\preceq_f$ is reflexive, i.e., for all $\alpha \in \Omega(S)$, $\alpha \preceq_f \alpha$.
3. $\preceq_f$ is transitive, i.e., for all $\alpha, \beta, \gamma \in \Omega(S)$, if $\alpha \preceq_f \beta$ and $\beta \preceq_f \gamma$, then $\alpha \preceq_f \gamma$.
4. If $\alpha_1 \preceq_f \beta_1, \ \ldots, \ \alpha_n \preceq_f \beta_n, \ \alpha_{n+1} \preceq_f \beta_{n+1}$, then $[\alpha_1, \ldots, \alpha_n, \alpha_{n+1}] \preceq_f [\beta_1, \ldots, \beta_n, \beta_{n+1}]$.

$\preceq_f$ is *noetherian* if every ascending sequence of members of $\Omega(S)$,

$$\alpha_1 \preceq_f \alpha_2 \preceq_f \alpha_3 \preceq_f \cdots,$$

is eventually stationary, i.e., there is some m such that $\alpha_i = \alpha_m$ for all $i \geq m$. If $\preceq_f$ is noetherian, then $\preceq_f$ is obviously antisymmetric (i.e., for all $\alpha, \beta \in \Omega(S)$, if $\alpha \preceq_f \beta$ and $\beta \preceq_f \alpha$, then $\alpha = \beta$). Hence, $\preceq_f$ is a partial order if it is noetherian.

An *S-tagged symbol* is a symbol tagged with a member of $\Omega(S)$. A tagged symbol whose symbol is a and whose tag is α is written as a_α. A tagged symbol a_α, where $a = b_i$, will be written as b^i_α (instead of as $(b_i)_\alpha$). Two tagged symbols a_α and b_β are *distinct* if $a \neq b$.

Sort Systems

The types and subtypes of a $\mathbf{PF}^*$ language are called "sorts". Syntactically, they form a partial order. The semantics of the partial order of sorts is a mapping that takes each sort to a nonempty set and takes the order relation to set inclusion.

A *sort system* of $\mathbf{PF}^*$ is a pair $(\mathcal{A}, \xi)$ where $\mathcal{A}$ is a finite set of symbols such that, for some $m \geq 1$, $\mathcal{B}_m = \{*, \iota_1, \ldots, \iota_m\} \subseteq \mathcal{A}$ and ξ is a total function from $\mathcal{A}$ to $\Omega(\mathcal{A})$ such that:

1. For all $\alpha \in \mathcal{A}$, $\alpha \in \mathcal{B}_m$ iff $\xi(\alpha) = \alpha$.
2. For all $\alpha \in \mathcal{A}$, $\xi(\alpha) \in \Omega^*(\mathcal{A})$ iff $\alpha = *$.
3. $\preceq_\xi$ is noetherian.

A *sort* of the system is any member of $\Omega(\mathcal{A})$, and a *type* of the system is any member of $\Omega(\mathcal{B}_m)$. The sorts in $\mathcal{A}$, $\Omega(\mathcal{A}) \setminus \mathcal{A}$, $\mathcal{B}_m$, and $\Omega(\mathcal{B}_m) \setminus \mathcal{B}_m$ are called the *atomic sorts*, *compound sorts*, *base types*, and *function types* of the system, respectively. ($*$ is the type of propositions, and each ι_k is a type of individuals.) The *enclosing sort* of $\alpha \in \mathcal{A}$ is the sort $\xi(\alpha)$.

Let $\mathcal{S}$ be a sort system $(\mathcal{A}, \xi)$. By $\alpha \in \mathcal{S}$ we shall mean $\alpha \in \Omega(\mathcal{A})$. Also, let $\mathcal{T}$ be the set of types of $\mathcal{S}$. The definition of a sort system implies that, for each $\alpha \in \mathcal{S}$, there is a unique $\beta \in \mathcal{T}$, called the *type* of α, such that $\alpha \preceq_\xi \beta$. The type of α is denoted by $\tau(\alpha)$. A sort α is of *kind* $*$ [respectively, ι] if $\alpha \in \Omega^*(\mathcal{A})$

[respectively, $\alpha \in \Omega(\mathcal{A}) \setminus \Omega^*(\mathcal{A})$]. Let $\mathcal{S}^*$ [$\mathcal{S}^\iota$] denote the set of all $\alpha \in \mathcal{S}$ of kind $*$ [ι]. The *least upper bound* of α and β, written $\alpha \sqcup_\xi \beta$, is the least upper bound of α and β in the partial order $\preceq_\xi$. Since each atomic sort has just a single enclosing sort, the least upper bound of two sorts with the same type is always defined.

A *standard sort frame* for $\mathcal{S}$ is a set $\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}$ of nonempty domains (sets) such that:

1. $\mathcal{D}_* = \{\mathrm{T}, \mathrm{F}\}$. ($\mathrm{T} \neq \mathrm{F}$.)
2. If $\alpha \preceq_\xi \beta$, then $\mathcal{D}_\alpha \subseteq \mathcal{D}_\beta$.
3. If $\alpha = [\alpha_1, \ldots, \alpha_n, \alpha_{n+1}]$ is of kind ι, then $\mathcal{D}_\alpha$ is the set of all *partial* functions $f : \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n} \rightarrow \mathcal{D}_{\alpha_{n+1}}$.
4. If $\alpha = [\alpha_1, \ldots, \alpha_n, \alpha_{n+1}]$ is of kind $*$, then $\mathcal{D}_\alpha$ is the set of all *total* functions $f : \mathcal{D}_{\tau(\alpha_1)} \times \cdots \times \mathcal{D}_{\tau(\alpha_n)} \rightarrow \mathcal{D}_{\alpha_{n+1}}$ such that, for all $\langle b_1, \ldots, b_n \rangle \in \mathcal{D}_{\tau(\alpha_1)} \times \cdots \times \mathcal{D}_{\tau(\alpha_n)}$, $f(b_1, \ldots, b_n) = \mathrm{F}_{\tau(\alpha_{n+1})}$ (see definition below) whenever $b_i \notin \mathcal{D}_{\alpha_i}$ for at least one i with $1 \leq i \leq n$.

For a type $\alpha \in \mathcal{S}^*$, F_α is defined inductively by:

1. $\mathrm{F}_* = \mathrm{F}$.
2. If $\alpha = [\alpha_1, \ldots, \alpha_n, \alpha_{n+1}]$, then F_α is the function which maps every n-tuple $\langle a_1, \ldots, a_n \rangle \in \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n}$ to $\mathrm{F}_{\alpha_{n+1}}$.

It follows from the definition of a standard sort frame that $\mathrm{F}_{\tau(\alpha)} \in \mathcal{D}_\alpha$ for all $\alpha \in \mathcal{S}^*$.

Languages

A *language* of $\mathbf{PF}^*$ is a tuple $(\mathcal{A}, \xi, \mathcal{V}, \mathcal{C})$ such that:

1. $(\mathcal{A}, \xi)$ is a sort system of $\mathbf{PF}^*$.
2. $\mathcal{V}$ and $\mathcal{C}$ are disjoint countable sets of pairwise distinct $\mathcal{A}$-tagged symbols, whose members are called *variables* and *constants*, respectively.
3. For each $\alpha \in \Omega(\mathcal{A})$, there is an infinite subset $\mathcal{V}_\alpha$ of $\mathcal{V}$ such that $x_\beta \in \mathcal{V}_\alpha$ iff $\beta = \alpha$.
4. For each $\alpha \in \Omega(\mathcal{A})$, $=_{[\alpha,\alpha,*]}$ and $\iota_{[[\alpha,*],\alpha]}$ are members of $\mathcal{C}$ called *logical constants*.

Variables are denoted by $f_\alpha, g_\alpha, h_\alpha, x_\alpha, y_\alpha, z_\alpha$, etc. For the rest of this section, let $\mathcal{L} = (\mathcal{A}, \xi, \mathcal{V}, \mathcal{C})$ be a language of $\mathbf{PF}^*$, $\mathcal{S}$ be the sort system $(\mathcal{A}, \xi)$, and $\mathcal{T}$ be the set of types of $\mathcal{S}$.

An *expression of $\mathcal{L}$ of sort α* is a finite sequence of members of $\mathcal{V} \cup \mathcal{C} \cup \{@, \lambda, \text{“\{”}, \text{“\}”}, \text{“,”}, \text{“.”}\}$ defined inductively by:

1. Each $x_\alpha \in \mathcal{V}$ and $c_\alpha \in \mathcal{C}$ is an expression of sort α.
2. If F is an expression of sort $[\alpha_1, \ldots, \alpha_n, \beta]$; $A_1, \ldots, A_n$ are expressions of sort $\alpha'_1, \ldots, \alpha'_n$; and $\tau(\alpha_1) = \tau(\alpha'_1), \ \ldots, \ \tau(\alpha_n) = \tau(\alpha'_n)$, then $@\{F, A_1, \ldots, A_n\}$ is an expression of sort β.

3. If $x^1_{\alpha_1}, \ldots, x^n_{\alpha_n}$ are distinct variables and B is an expression of sort β, then $\lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}$ is an expression of sort $[\alpha_1, \ldots, \alpha_n, \beta]$.

Expressions are denoted by A, B, C, etc., and expressions of sort α are denoted by $A_\alpha, B_\alpha, C_\alpha$, etc. "Free variable", "closed expression", and similar notions are defined in the obvious way. A *formula* is an expression of sort $*$. A *sentence* is a closed formula.

Intuitively, when an expression $@\{F, A_1, \ldots, A_n\}$ has a denotation, it denotes the value of the function denoted by F applied to the arguments denoted by $A_1, \ldots, A_n$. Also, an expression $\lambda\{x_1, \ldots, x_n \,.\, B\}$ denotes the function whose value, when given arguments $x_1, \ldots, x_n$, is the denotation of B (which generally depends on $x_1, \ldots, x_n$) if B has a denotation and is undefined otherwise.

The following abbreviations provide notation for the major logical operations of $\mathbf{PF}^*$:

$F(A_1, \ldots, A_n)$	for	$@\{F, A_1, \ldots, A_n\}$
$(A_\alpha =_\gamma B_\beta)$	for	$=_{[\gamma,\gamma,*]} (A_\alpha, B_\beta)$
$(A_* \equiv B_*)$	for	$(A_* =_* B_*)$
T_*	for	$(=_{[*,*,*]} \; =_{[*,*,*]} \; =_{[*,*,*]})$
F_*	for	$(\lambda\{x_* \,.\, \mathsf{T}_*\} =_{[*,*]} \lambda\{x_* \,.\, x_*\})$
$\neg_{[*,*]}$	for	$\lambda\{x_* \,.\, (x_* \equiv \mathsf{F}_*)\}$
$(\neg A_*)$	for	$\neg_{[*,*]}(A_*)$
$\wedge_{[*,*,*]}$	for	$\lambda\{x_*, y_* \,.\, (\lambda\{g_{[*,*,*]} \,.\, g_{[*,*,*]}(\mathsf{T}_*, \mathsf{T}_*)\} =_{[[*,*,*],*]}$ $\lambda\{g_{[*,*,*]} \,.\, g_{[*,*,*]}(x_*, y_*)\})\}$
$(A_* \wedge B_*)$	for	$\wedge_{[*,*,*]}(A_*, B_*)$
$\vee_{[*,*,*]}$	for	$\lambda\{x_*, y_* \,.\, (\neg((\neg x_*) \wedge (\neg y_*)))\}$
$(A_* \vee B_*)$	for	$\vee_{[*,*,*]}(A_*, B_*)$
$\supset_{[*,*,*]}$	for	$\lambda\{x_*, y_* \,.\, (x_* \equiv (x_* \wedge y_*))\}$
$(A_* \supset B_*)$	for	$\supset_{[*,*,*]} (A_*, B_*)$
$\mathit{\Pi}_{[[\alpha_1,\ldots,\alpha_n,*],*]}$	for	$\lambda\{p_{[\alpha_1,\ldots,\alpha_n,*]} \,.\, (p_{[\alpha_1,\ldots,\alpha_n,*]} =_{[\alpha_1,\ldots,\alpha_n,*]}$ $\lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, \mathsf{T}_*\})\}$
$\forall\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, A_*\}$	for	$\mathit{\Pi}_{[[\alpha_1,\ldots,\alpha_n,*],*]}(\lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, A_*\})$
$\exists\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, A_*\}$	for	$(\neg\forall\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, (\neg A_*)\})$
$\mathrm{I}\{x_\alpha \,.\, A_*\}$	for	$\iota_{[[\alpha,*],\alpha]}(\lambda\{x_\alpha \,.\, A_*\})$
$(A_\alpha \downarrow \beta)$	for	$\lambda\{x_\beta \,.\, \mathsf{T}_*\}(A_\alpha)$
$(A_\alpha \uparrow \beta)$	for	$(\neg(A_\alpha \downarrow \beta))$
$(A_\alpha\downarrow)$	for	$(A_\alpha \downarrow \alpha)$
$(A_\alpha\uparrow)$	for	$(\neg(A_\alpha\downarrow))$
$(A_\alpha \simeq_\gamma B_\beta)$	for	$((A_\alpha\downarrow) \vee (B_\beta\downarrow)) \supset (A_\alpha =_\gamma B_\beta)$
$(A_\alpha \neq_\gamma B_\beta)$	for	$(\neg(A_\alpha =_\gamma B_\beta))$
$(A_\alpha \not\simeq_\gamma B_\beta)$	for	$(\neg(A_\alpha \simeq_\gamma B_\beta))$
$\bot_\alpha$	for	$\mathrm{I}\{x_\alpha \,.\, \mathsf{F}_*\}$ where $\alpha \in \mathcal{S}^\iota$
$\mathsf{F}_{[\alpha_1,\ldots,\alpha_n,\beta]}$	for	$\lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, \mathsf{F}_\beta\}$ where $\beta \in \mathcal{S}^*$
$\alpha \subseteq \beta$	for	$\forall\{x_\alpha \,.\, (x_\alpha \downarrow \beta)\}$
$\mathrm{if}\{A_*, B_\beta, C_\gamma\}$	for	$\mathrm{I}\{x_\alpha \,.\, ((A_* \supset (x_\alpha =_\alpha B_\beta)) \wedge ((\neg A_*) \supset$ $(x_\alpha =_\alpha C_\gamma)))\}$ where $\alpha = \beta \sqcup_\xi \gamma$ and x_α does not occur in A_*, B_β, or C_γ.

Parentheses in expressions may be suppressed when meaning is not lost.

Standard Models

In [9] we define two semantics for $\mathbf{PF}^*$, one based on "general models" and the other based on "standard models". In this paper we are interested only in the standard models semantics. We begin by describing the objects that the logical constants are intended to denote.

Let $\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}$ be a standard sort frame for $\mathcal{S}$. Fix $\beta \in \mathcal{S}$ with $\gamma = \tau(\beta)$. The *identity relation for* $\mathcal{D}_\beta$ is the total function $p : \mathcal{D}_\gamma \times \mathcal{D}_\gamma \to \mathcal{D}_*$ such that, for all $a, b \in \mathcal{D}_\gamma$, $p(a, b) = \mathrm{T}$ iff $a, b \in \mathcal{D}_\beta$ and $a = b$. Given $b \in \mathcal{D}_\gamma$, the *b-descriptor for* $\mathcal{D}_\gamma$ is the total function $p : \mathcal{D}_\gamma \to \mathcal{D}_*$ such that, for all $a \in \mathcal{D}_\gamma$, $p(a) = \mathrm{T}$ iff $a = b$. When $\beta \in \mathcal{S}^\iota$, the *definite description function for* $\mathcal{D}_\beta$ is the partial function $f : \mathcal{D}_{[\gamma,*]} \to \mathcal{D}_\gamma$ such that $f(p) = b$ if p is the b-descriptor for $\mathcal{D}_\gamma$ for some $b \in \mathcal{D}_\beta$, and $f(p)$ is undefined otherwise. When $\beta \in \mathcal{S}^*$, the *definite description function for* $\mathcal{D}_\beta$ is the total function $f : \mathcal{D}_{[\gamma,*]} \to \mathcal{D}_\gamma$ such that $f(p) = b$ if p is the b-descriptor for $\mathcal{D}_\gamma$ for some $b \in \mathcal{D}_\beta$, and $f(p) = \mathrm{F}_\gamma$ otherwise.

A *standard model for* $\mathcal{L}$ is a pair $\mathcal{M} = (\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}, I)$ where $\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}$ is a standard sort frame for $\mathcal{S}$ such that $\perp \notin \mathcal{D}_\alpha$ for all $\alpha \in \mathcal{S}$ and I is a function which maps each $c_\alpha \in \mathcal{C}$ to an element of $\mathcal{D}_\alpha$ such that, for each $\alpha \in \mathcal{S}$, (1) $I(=_{[\alpha,\alpha,*]})$ is the identity relation for $\mathcal{D}_\alpha$ and (2) $I(\iota_{[[\alpha,*],\alpha]})$ is the definite description function for $\mathcal{D}_\alpha$. A *$\mathcal{V}$-assignment* into $\mathcal{M}$ is a function which maps each $x_\alpha \in \mathcal{V}$ to an element of $\mathcal{D}_\alpha$. Given a $\mathcal{V}$-assignment φ into $\mathcal{M}$; distinct variables $x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \in \mathcal{V}$ $(n \geq 1)$; and $a_1 \in \mathcal{D}_{\alpha_1}, \ldots a_n \in \mathcal{D}_{\alpha_n}$, let $\varphi[x^1_{\alpha_1} \mapsto a_1, \ldots, x^n_{\alpha_n} \mapsto a_n]$ be the $\mathcal{V}$-assignment ψ such that $\psi(x^i_{\alpha_i}) = a_i$ for all i with $1 \leq i \leq n$ and $\psi(y_\beta) = \varphi(y_\beta)$ for all $y_\beta \notin \{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n}\}$.

Let $\mathcal{M} = (\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}, I)$ be a standard model for $\mathcal{L}$. Then there exists a binary function $V = V^{\mathcal{M}}$ such that: (1) for each $\mathcal{V}$-assignment φ into $\mathcal{M}$ and each expression A_α, $V_\varphi(A_\alpha) \in \mathcal{D}_\alpha \cup \{\perp\}$ if $\alpha \in \mathcal{S}^\iota$ and $V_\varphi(A_\alpha) \in \mathcal{D}_\alpha$ if $\alpha \in \mathcal{S}^*$, and (2) the following conditions are satisfied for all $\mathcal{V}$-assignments φ into $\mathcal{M}$ and all expressions:

1. $V_\varphi(x_\alpha) = \varphi(x_\alpha)$ if $x_\alpha \in \mathcal{V}$.
2. $V_\varphi(c_\alpha) = I(c_\alpha)$ if $c_\alpha \in \mathcal{C}$.
3. Let $A_\alpha = @\{F, A_1, \ldots, A_n\}$ with $\alpha \in \mathcal{S}^\iota$. If each of $V_\varphi(F)$, $V_\varphi(A_1), \ldots,$ $V_\varphi(A_n)$ does not equal $\perp$ and $V_\varphi(F)$ is defined at $\langle V_\varphi(A_1), \ldots, V_\varphi(A_n)\rangle$, then
$$V_\varphi(A_\alpha) = (V_\varphi(F))(V_\varphi(A_1), \ldots, V_\varphi(A_n));$$
otherwise $V_\varphi(A_\alpha) = \perp$.
4. Let $A_\alpha = @\{F, A_1, \ldots, A_n\}$ with $\alpha = [\alpha_1, \ldots, \alpha_n, \beta] \in \mathcal{S}^*$. If each of $V_\varphi(A_1)$,...,$V_\varphi(A_n)$ does not equal $\perp$, then
$$V_\varphi(A_\alpha) = (V_\varphi(F))(V_\varphi(A_1), \ldots, V_\varphi(A_n));$$
otherwise $V_\varphi(A_\alpha) = \mathrm{F}_{\tau(\beta)}$.

5. Let $A_\alpha = \lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B_\beta\}$ with $\beta \in \mathcal{S}^\iota$. $V_\varphi(A_\alpha)$ is the partial function $f : \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n} \to \mathcal{D}_\beta$ such that

$$f(a_1, \ldots, a_n) = V_{\varphi[x^1_{\alpha_1} \mapsto a_1, \ldots, x^n_{\alpha_n} \mapsto a_n]}(B_\beta)$$

   if $V_{\varphi[x^1_{\alpha_1} \mapsto a_1, \ldots, x^n_{\alpha_n} \mapsto a_n]}(B_\beta)$ is defined; otherwise $f(a_1, \ldots, a_n)$ is undefined.
6. Let $A_\alpha = \lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B_\beta\}$ with $\beta \in \mathcal{S}^*$. $V_\varphi(A_\alpha)$ is the total function $f : \mathcal{D}_{\tau(\alpha_1)} \times \cdots \times \mathcal{D}_{\tau(\alpha_n)} \to \mathcal{D}_\beta$ such that

$$f(a_1, \ldots, a_n) = V_{\varphi[x^1_{\alpha_1} \mapsto a_1, \ldots, x^n_{\alpha_n} \mapsto a_n]}(B_\beta)$$

   if $\langle a_1, \ldots, a_n \rangle \in \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n}$; otherwise $f(a_1, \ldots, a_n) = \mathrm{F}_{\tau(\beta)}$.

When $V_\varphi(A_\alpha) \neq \bot$, $V_\varphi(A_\alpha)$ is called the *value* of A_α in $\mathcal{M}$ with respect to φ. Intuitively, $V_\varphi(A_\alpha) = \bot$ means that A_α has no denotation or value in $\mathcal{M}$ with respect to φ. That is, V is intuitively a *partial* valuation function. For a closed expression A_α, it is clear that $V_\varphi(A_\alpha)$ does not depend on φ and thus $V(A_\alpha)$ is meaningful.

A formula A_* of $\mathcal{L}$ is *valid in* $\mathcal{M}$ if $V_\varphi(A_*) = \mathrm{T}$ for every $\mathcal{V}$-assignment φ into $\mathcal{M}$.

Theories

A *theory* of $\mathbf{PF}^*$ is a pair $T = (\mathcal{L}, \Gamma)$ where $\mathcal{L}$ is a language of $\mathbf{PF}^*$ and Γ is a set of sentences of $\mathcal{L}$. The members of Γ are called the *axioms* of T. T, T', etc. denote theories.

Let $T = (\mathcal{L}, \Gamma)$. A *standard model for* T is a standard model for $\mathcal{L}$ in which each member of Γ is valid. T is *satisfiable (in the standard sense)* if there is some standard model for T. A sentence A_* is a *(semantic) theorem of T (in the standard sense)*, written $T \models A_*$, if A_* is valid in every standard model for T.

5 LUTINS

LUTINS (Logic of Undefined Terms for Inference in a Natural Style) is the logic of IMPS. It is essentially the same as $\mathbf{PF}^*$ plus several additional expression constructors, which are defined in terms of the primitive notions of $\mathbf{PF}^*$: function application, λ-abstraction, equality, and definite description. This section gives a brief presentation of the syntax and semantics of LUTINS. A more detailed presentation of a slightly different version of LUTINS is found in [16].[5] We assume that each concept and notation defined in the previous section for $\mathbf{PF}^*$ is defined for LUTINS in the obvious way if it is not explicitly defined in this section.

[5] The reader should be aware that in [16] LUTINS is called $\mathbf{PF}$.

Syntax

A *language* of LUTINS is a tuple $(\mathcal{A}, \xi, \mathcal{V}, \mathcal{C})$ such that $(\mathcal{A}, \xi, \mathcal{V}, \widetilde{\mathcal{C}})$ is a language of $\mathbf{PF}^*$, where

$$\widetilde{\mathcal{C}} = \mathcal{C} \cup \{=_{[\alpha,\alpha,*]}, \iota_{[[\alpha,*],\alpha]} : \alpha \in \Omega(\mathcal{A})\}.$$

For any language $\mathcal{L} = (\mathcal{A}, \xi, \mathcal{V}, \mathcal{C})$ of LUTINS, let $\widetilde{\mathcal{L}}$ be the language $(\mathcal{A}, \xi, \mathcal{V}, \widetilde{\mathcal{C}})$ of $\mathbf{PF}^*$.

For the remainder of this section, let $\mathcal{L} = (\mathcal{A}, \xi, \mathcal{V}, \mathcal{C})$ be a language of LUTINS, $\mathcal{S}$ be the sort system $(\mathcal{A}, \xi)$, and $\mathcal{T}$ be the set of types of $\mathcal{S}$. When there is no possibility of confusion, the components of a language $\mathcal{L}_i$ will be denoted by $\mathcal{A}_i, \xi_i, \mathcal{V}_i, \mathcal{C}_i$, and the sort system and set of types of $\mathcal{L}_i$ will be denoted by $\mathcal{S}_i$ and $\mathcal{T}_i$, respectively.

Our version of LUTINS has 21 expression constructors: `apply`, `lambda`, `equals`, `quasi-equals`, `iota`, `iota-p`, `the-true`, `the-false`, `not`, `and`, `or`, `implies`, `iff`, `if-form`, `forall`, `forsome`, `defined-in`, `is-defined`, `if`, `undefined`, and `falselike`. An *expression of $\mathcal{L}$ of sort $\alpha \in \mathcal{S}$* is a finite sequence of variables, constants, expression constructors, sorts, and punctuation symbols defined inductively by:

1. Each $x_\alpha \in \mathcal{V}$ and $c_\alpha \in \mathcal{C}$ is an expression of sort α.
2. If F is an expression of sort $[\alpha_1, \ldots, \alpha_n, \beta]$; $A_1, \ldots, A_n$ are expressions of sort $\alpha'_1, \ldots, \alpha'_n$; and $\tau(\alpha_1) = \tau(\alpha'_1), \ldots, \tau(\alpha_n) = \tau(\alpha'_n)$, then $\mathtt{apply}\{F, A_1, \ldots, A_n\}$ is an expression of sort β.
3. If $x^1_{\alpha_1}, \ldots, x^n_{\alpha_n}$ are distinct variables and B is an expression of sort β, then $\mathtt{lambda}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}$ is an expression of sort $[\alpha_1, \ldots, \alpha_n, \beta]$.
4. If A and B are expressions of sort α and β, respectively, with $\tau(\alpha) = \tau(\beta)$, then $\mathtt{equals}\{A, B\}$ and $\mathtt{quasi\text{-}equals}\{A, B\}$ are expressions of sort $*$.
5. If x_α, y_β are variables; α, β are of kind $\iota, *$, respectively; and C is an expression of sort $*$, then $\mathtt{iota}\{x_\alpha \,.\, C\}$ and $\mathtt{iota\text{-}p}\{y_\beta \,.\, C\}$ are expressions of sort α and β, respectively.
6. If $A, B, C, A_1, \ldots, A_n$ are expressions of sort $*$ with $n \geq 0$, then $\mathtt{the\text{-}true}\{\}$, $\mathtt{the\text{-}false}\{\}$, $\mathtt{not}\{A\}$ $\mathtt{and}\{A_1, \ldots, A_n\}$, $\mathtt{or}\{A_1, \ldots, A_n\}$, $\mathtt{implies}\{A, B\}$, $\mathtt{iff}\{A, B\}$, and $\mathtt{if\text{-}form}\{A, B, C\}$ are expressions of sort $*$.
7. If $x^1_{\alpha_1}, \ldots, x^n_{\alpha_n}$ are distinct variables and B is an expression of sort $*$, then $\mathtt{forall}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}$ and $\mathtt{forsome}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}$ are expressions of sort $*$.
8. If A is an expression of sort α and $\tau(\alpha) = \tau(\beta)$, then $\mathtt{defined\text{-}in}\{A, \beta\}$ and $\mathtt{is\text{-}defined}\{A\}$ are expressions of sort $*$.
9. If A, B, C are expressions of sort $*, \beta, \gamma$, respectively, with $\tau(\beta) = \tau(\gamma)$, then $\mathtt{if}\{A, B, C\}$ is an expression of sort $\beta \sqcup_\xi \gamma$.
10. If α is of kind ι, then $\mathtt{undefined}\{\alpha\}$ is an expression of sort α.
11. If α is of kind $*$, then $\mathtt{falselike}\{\alpha\}$ is an expression of sort α.

Expressions that are not variables or constants are called *compound expressions.* The *length* of an expression A, written $|A|$, is the number of occurrences of expression constructors in A. The set of expressions of $\mathcal{L}$ [respectively, $\mathcal{L}_i$] is denoted by $\mathcal{E}$ [respectively, $\mathcal{E}_i$], and the set of expressions of $\widetilde{\mathcal{L}}$ [$\widetilde{\mathcal{L}_i}$] is denoted by $\widetilde{\mathcal{E}}$ [$\widetilde{\mathcal{E}}_i$].

Semantics

The semantics of LUTINS is defined in terms of the standard models semantics of $\mathbf{PF}^*$.

Let $\zeta : \mathcal{E} \to \widetilde{\mathcal{E}}$ be the mapping defined by the following statements:

1. $\zeta(a_\alpha) = a_\alpha$ where $a_\alpha \in \mathcal{V} \cup \mathcal{C}$.
2. $\zeta(\texttt{apply}\{F, A_1, \ldots, A_n\}) = @\{\zeta(F), \zeta(A_1), \ldots, \zeta(A_n)\}$.
3. $\zeta(\texttt{lambda}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}) = \lambda\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, \zeta(B)\}$.
4. $\zeta(\texttt{equals}\{A_\alpha, B_\beta\}) = (\zeta(A_\alpha) =_\gamma \zeta(B_\beta))$ where $\gamma = \tau(\alpha) = \tau(\beta)$.
5. $\zeta(\texttt{quasi-equals}\{A_\alpha, B_\beta\}) = (\zeta(A_\alpha) \simeq_\gamma \zeta(B_\beta))$ where $\gamma = \tau(\alpha) = \tau(\beta)$.
6. $\zeta(\Box\{x_\alpha \,.\, A_*\}) = \mathrm{I}\{x_\alpha \,.\, \zeta(A_*)\}$ where $\Box$ is `iota` or `iota-p`.
7. $\zeta(\Box\{\}) = \mathsf{T}_*$ where $\Box$ is `the-true` or `and`.
8. $\zeta(\Box\{\}) = \mathsf{F}_*$ where $\Box$ is `the-false` or `or`.
9. $\zeta(\texttt{not}\{A\}) = \neg\zeta(A)$.
10. $\zeta(\texttt{and}\{A_1, \ldots, A_n\}) = (\zeta(A_1) \wedge (\zeta(A_2) \wedge \cdots))$ where $n \geq 1$.
11. $\zeta(\texttt{or}\{A_1, \ldots, A_n\}) = (\zeta(A_1) \vee (\zeta(A_2) \vee \cdots))$ where $n \geq 1$.
12. $\zeta(\texttt{implies}\{A, B\}) = (\zeta(A) \supset \zeta(B))$.
13. $\zeta(\texttt{iff}\{A, B\}) = (\zeta(A) \equiv \zeta(B))$.
14. $\zeta(\Box\{A, B, C\}) = \mathrm{if}\{\zeta(A), \zeta(B), \zeta(C)\}$ where $\Box$ is `if-form` or `if`.
15. $\zeta(\texttt{forall}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}) = \forall\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, \zeta(B)\}$.
16. $\zeta(\texttt{forsome}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}) = \exists\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, \zeta(B)\}$.
17. $\zeta(\texttt{defined-in}\{A, \beta\}) = (\zeta(A) \downarrow \beta)$.
18. $\zeta(\texttt{is-defined}\{A\}) = \zeta(A)\!\downarrow$.
19. $\zeta(\texttt{undefined}\{\alpha\}) = \perp_\alpha$.
20. $\zeta(\texttt{falselike}\{\alpha\}) = \mathsf{F}_\alpha$.

Notice that A_α and $\zeta(A_\alpha)$ have the same sort for every expression A_α of $\mathcal{L}$.

A *model for* $\mathcal{L}$ is a standard model for $\widetilde{\mathcal{L}}$. Let $\mathcal{M} = (\{\mathcal{D}_\alpha : \alpha \in \mathcal{S}\}, I)$ be a model for $\mathcal{L}$, and let $V = V^{\mathcal{M}}$ be the valuation function for $\widetilde{\mathcal{L}}$ with respect to $\mathcal{M}$. $U = U^{\mathcal{M}}$ is the binary function, on $\mathcal{V}$-assignments φ into $\mathcal{M}$ and expressions A_α of $\mathcal{L}$, defined by

$$U_\varphi(A_\alpha) = V_\varphi(\zeta(A_\alpha)).$$

Clearly, for each $\mathcal{V}$-assignment φ into $\mathcal{M}$ and each expression A_α of $\mathcal{L}$, $U_\varphi(A_\alpha) \in \mathcal{D}_\alpha \cup \{\perp\}$ if $\alpha \in \mathcal{S}^\iota$ and $U_\varphi(A_\alpha) \in \mathcal{D}_\alpha$ if $\alpha \in \mathcal{S}^*$. Thus U is a (partial) valuation function for $\mathcal{L}$ with respect to $\mathcal{M}$.

A formula A_* of $\mathcal{L}$ is *valid in* $\mathcal{M}$ if $U_\varphi(A_*) = \mathrm{T}$ for every $\mathcal{V}$-assignment φ into $\mathcal{M}$.

Let $\mathcal{M}_i = (\{\mathcal{D}^i_\alpha : \alpha \in \mathcal{S}_i\}, I_i)$ be a model for $T_i = (\mathcal{L}_i, \Gamma_i)$ for $i = 1, 2$. $\mathcal{L}_2$ is an *expansion* of $\mathcal{L}_1$, written $\mathcal{L}_1 \leq \mathcal{L}_2$, if $\mathcal{A}_1 \subseteq \mathcal{A}_2$, ξ_1 is a subfunction of ξ_2, $\mathcal{V}_1 \subseteq \mathcal{V}_2$, and $\mathcal{C}_1 \subseteq \mathcal{C}_2$. T_2 is an *extension* of T_1, written $T_1 \leq T_2$, if $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\Gamma_1 \subseteq \Gamma_2$. $\mathcal{M}_2$ is an *expansion* of $\mathcal{M}_1$ to $\mathcal{L}_2$ (and $\mathcal{M}_2$ is the *reduct* of $\mathcal{M}_1$ to $\mathcal{L}_1$) if $\mathcal{D}^1_\alpha = \mathcal{D}^2_\alpha$ for all $\alpha \in \mathcal{S}_1$ and I_1 is a subfunction of I_2.

6 Theory Interpretation in LUTINS

We have seen that, in first-order predicate logic, an interpretation of one theory in another is a certain kind of homomorphism on expressions that preserves the validity of sentences. In this section we define an interpretation of one LUTINS theory in another which is very similar to the notion of a standard first-order interpretation. We will first define a notion of a *translation* from a source theory to a target theory. A translation is determined by how the primitive symbols of the source theory (i.e., the atomic sorts, variables, and constants of the source theory) are associated with objects of the target theory. In the definition given here, atomic sorts are associated with sorts and closed unary predicates; variables are associated with variables; and constants are associated with closed expressions (of appropriate sort). Then an *interpretation* is defined to be a translation which maps every theorem of the source theory to a theorem of the target theory.

Our notion of a translation in LUTINS extends the notion of a translation in $\mathbf{PF}^*$ defined in [9] in two ways:

- A LUTINS translation directly handles all 21 expression constructors of LUTINS, while a $\mathbf{PF}^*$ translation handles just the constructors corresponding to `apply` and `lambda`.
- A LUTINS translation can associate an atomic sort with either a sort or a closed unary predicate, but a $\mathbf{PF}^*$ translation can associate an atomic sort with only a sort.

This section is the heart of the paper. Most of the complexity in the definitions is a result of allowing sorts to be associated with unary predicates.

Quasi-Sorts

Let us define a *quasi-sort* to be any closed unary predicate, i.e., any expression whose sort is of the form $[\alpha, *]$. Then let $\mathcal{Q}$ $[\mathcal{Q}_i]$ denote the set of quasi-sorts of the language $\mathcal{L}$ $[\mathcal{L}_i]$.

For $q \in \mathcal{S} \cup \mathcal{Q}$ and $A_\alpha \in \mathcal{E}$, the *domain* of q, written $\delta(q)$, is defined by

$$\delta(q) = \begin{cases} q \text{ if } q \in \mathcal{S} \\ \alpha \text{ if } q \in \mathcal{Q} \text{ and } [\alpha, *] \text{ is the sort of } q \end{cases}$$

and, provided $\delta(q) = \alpha$, the *condition of q on A_α*, written $\kappa(q, A_\alpha)$, is defined by

$$\kappa(q, A_\alpha) = \begin{cases} \texttt{the-true}\{\} & \text{if } q \in \mathcal{S} \\ \texttt{apply}\{q, A_\alpha\} & \text{if } q \in \mathcal{Q} \end{cases}$$

We will need a quasi-sort constructor (denoted by $[\![\cdots]\!]$) for building "compound" quasi-sorts that is analogous to the $[\cdots]$ sort constructor for building compound sorts. Let $q_1, \ldots, q_{n+1} \in \mathcal{S} \cup \mathcal{Q}$ with $n \geq 1$. When $\delta(q_{n+1}) \in \mathcal{S}^\iota$, $[\![q_1, \ldots, q_{n+1}]\!]$ will be a quasi-sort whose extension (in a model) is the set of partial functions $f : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \rightharpoonup \mathcal{D}_{n+1}$, where $\mathcal{D}_i$ is the extension of q_i for i with $1 \leq i \leq n+1$.

$[\![q_1, \ldots, q_{n+1}]\!]$ is defined as follows. Let $\gamma = [\delta(q_1), \ldots, \delta(q_{n+1})]$. Choose distinct variables $f_\gamma, x^1_{\delta(q_1)}, \ldots, x^n_{\delta(q_n)}$ which do not occur in $q_1, \ldots, q_{n+1}$. Make the following definitions:

$$A = \texttt{and}\{\kappa(q_1, x^1_{\delta(q_1)}), \ldots, \kappa(q_n, x^n_{\delta(q_n)})\}.$$
$$B = \texttt{apply}\{f_\gamma, x^1_{\delta(q_1)}, \ldots, x^n_{\delta(q_n)}\}.$$
$$C = \begin{cases} \texttt{if-form}\{A, & \text{if } \gamma \text{ is of kind } \iota \\ \quad \texttt{implies}\{\texttt{is-defined}\{B\}, \kappa(q_{n+1}, B)\}, & \\ \quad \texttt{not}\{\texttt{is-defined}\{B\}\}\} & \\ \texttt{if-form}\{A, & \text{if } \gamma \text{ is of kind } * \\ \quad \kappa(q_{n+1}, B), & \\ \quad \texttt{equals}\{B, \texttt{falselike}\{\delta(q_{n+1})\}\}\} & \end{cases}$$

Then define $[\![q_1, \ldots, q_{n+1}]\!]$ to be

$$\texttt{lambda}\{f_\gamma \,.\, \texttt{forall}\{x^1_{\delta(q_1)}, \ldots, x^n_{\delta(q_n)} \,.\, C\}\}.$$

Clearly, for all $q_1, \ldots, q_{n+1} \in \mathcal{S} \cup \mathcal{Q}$, $[\![q_1, \ldots, q_{n+1}]\!]$ is a well-formed member of $\mathcal{E}$.

Proposition 6.1

1. *If* $\alpha_1, \ldots, \alpha_{n+1} \in \mathcal{S}$, *then* $\delta([\alpha_1, \ldots, \alpha_{n+1}]) = [\delta(\alpha_1), \ldots, \delta(\alpha_{n+1})]$.
2. *If* $q_1, \ldots, q_{n+1} \in \mathcal{S} \cup \mathcal{Q}$, *then* $\delta([\![q_1, \ldots, q_{n+1}]\!]) = [\delta(q_1), \ldots, \delta(q_{n+1})]$.

Translations

Let $T_i = (\mathcal{L}_i, \Gamma_i)$ be a theory of LUTINS for $i = 1, 2$. Given a function $\mu : \mathcal{A}_1 \to \mathcal{S}_2 \cup \mathcal{Q}_2$, $\bar{\mu} : \mathcal{S}_1 \to \mathcal{S}_2 \cup \mathcal{Q}_2$ is the canonical extension of μ defined inductively by:

1. If $\alpha \in \mathcal{A}_1$, then $\bar{\mu}(\alpha) = \mu(\alpha)$.
2. If $\alpha = [\alpha_1, \ldots, \alpha_{n+1}] \in \mathcal{S}_1$ and $\{\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_{n+1})\} \subseteq \mathcal{S}_2$, then $\bar{\mu}(\alpha) = [\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_{n+1})]$.
3. If $\alpha = [\alpha_1, \ldots, \alpha_{n+1}] \in \mathcal{S}_1$ and $\{\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_{n+1})\} \not\subseteq \mathcal{S}_2$, then $\bar{\mu}(\alpha) = [\![\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_{n+1})]\!]$.

Given $\alpha \in \mathcal{A}_1$, let $\mu[\alpha] = \delta(\mu(\alpha))$, and given $\alpha \in \mathcal{S}_1$, let $\bar{\mu}[\alpha] = \delta(\bar{\mu}(\alpha))$.

A *translation from* T_1 *to* T_2 is a pair (μ, ν), where $\mu : \mathcal{A}_1 \to \mathcal{S}_2 \cup \mathcal{Q}_2$ and $\nu : \mathcal{V}_1 \cup \mathcal{C}_1 \to \mathcal{E}_2$, such that:

1. $\mu(*) = *$.
2. For all $\alpha \in \mathcal{A}_1$ with $\alpha \neq *$, $\mu[\alpha]$ is a sort of type $\tau(\bar{\mu}[\tau(\alpha)])$ of kind ι.
3. For all $x_\alpha \in \mathcal{V}_1$, $\nu(x_\alpha)$ is a variable of sort $\bar{\mu}[\alpha]$.
4. ν is injective on $\mathcal{V}_1$.
5. For all $c_\alpha \in \mathcal{C}_1$, $\nu(c_\alpha)$ is a closed expression of type $\tau(\bar{\mu}[\alpha])$.

(μ, ν) is *normal* if $\mu(\alpha) \in \mathcal{S}_2$ for all $\alpha \in \mathcal{A}_1$. Obviously, $\bar{\mu}[\alpha] = \bar{\mu}(\alpha)$ for all $\alpha \in \mathcal{S}_1$ if (μ, ν) is normal.

Let $\Phi = (\mu, \nu)$ be a translation from T_1 to T_2 throughout the rest of this section. For $A \in \mathcal{E}_1$, the *translation of* A *via* Φ, written $\Phi(A)$, is the member of $\mathcal{E}_2$ defined inductively by:

1. $\Phi(a) = \nu(a)$ if $a \in \mathcal{V}_1 \cup \mathcal{C}_1$.
2. $\Phi(\Box\{A_1, \ldots, A_n\}) = \Box\{\Phi(A_1), \ldots, \Phi(A_n)\}$ if $\Box$ is `apply`, `equals`, `quasi-equals`, `the-true`, `the-false`, `not`, `and`, `or`, `implies`, `iff`, `if-form`, `is-defined`, or `if`.
3. $\Phi(\Box\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}) = \Box\{\Phi(x^1_{\alpha_1}), \ldots, \Phi(x^n_{\alpha_n}) \,.\, \Phi(B)\}$ if $\Box$ is `lambda`, `iota`, `iota-p`, `forall`, or `forsome` and $\{\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_n)\} \subseteq \mathcal{S}_2$.
4. $\Phi(\Box\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}) =$

$$\begin{cases} \Box\{\Phi(x^1_{\alpha_1}), \ldots, \Phi(x^n_{\alpha_n}) \,.\, \texttt{if}\{C, \Phi(B), \Box'\{\gamma\}\}\} & \text{if } \Box \text{ is } \texttt{lambda} \\ \Box\{\Phi(x^1_{\alpha_1}), \ldots, \Phi(x^n_{\alpha_n}) \,.\, \texttt{implies}\{C, \Phi(B)\}\} & \text{if } \Box \text{ is } \texttt{forall} \\ \Box\{\Phi(x^1_{\alpha_1}), \ldots, \Phi(x^n_{\alpha_n}) \,.\, \texttt{and}\{C, \Phi(B)\}\} & \text{if } \Box \text{ is } \texttt{forsome}, \\ & \texttt{iota}, \text{ or } \texttt{iota-p} \end{cases}$$

   if $\{\bar{\mu}(\alpha_1), \ldots, \bar{\mu}(\alpha_n)\} \not\subseteq \mathcal{S}_2$, where

$$C = \texttt{and}\{\kappa(\bar{\mu}(\alpha_1), \Phi(x^1_{\alpha_1})), \ldots, \kappa(\bar{\mu}(\alpha_n), \Phi(x^n_{\alpha_n}))\},$$

   γ is the sort of $\Phi(B)$, and $\Box'$ is `undefined` [`falselike`] if γ is of kind ι [$*$].
5. $\Phi(\Box\{\alpha\}) = \Box\{\bar{\mu}[\alpha]\}$ if $\Box$ is `undefined` or `falselike`.
6. $\Phi(\texttt{defined-in}\{A, \beta\}) = \begin{cases} \texttt{defined-in}\{\Phi(A), \bar{\mu}(\beta)\} & \text{if } \bar{\mu}(\beta) \in \mathcal{S}_2 \\ \texttt{apply}\{\bar{\mu}(\beta), \Phi(A)\} & \text{if } \bar{\mu}(\beta) \in \mathcal{Q}_2 \end{cases}$

Proposition 6.2 *Let $\alpha, \beta \in \mathcal{S}_1$.*

1. *α and $\bar{\mu}[\alpha]$ are of the same kind.*
2. *$\tau(\bar{\mu}[\alpha]) = \tau(\bar{\mu}[\tau(\alpha)])$.*
3. *$\tau(\alpha) = \tau(\beta)$ implies $\tau(\bar{\mu}[\alpha]) = \tau(\bar{\mu}[\beta])$.*

Proof The same as the proof of Proposition 11.1 in [9].$\Box$

Proposition 6.3 *Let $A_\alpha \in \mathcal{E}_1$.*

1. *$\Phi(A_\alpha)$ is a (well-formed) member of $\mathcal{E}_2$ of type $\tau(\bar{\mu}[\alpha])$.*
2. *If the sort of $\nu(c_\gamma)$ is $\bar{\mu}[\gamma]$ for each constant c_γ occurring in A_α, then the sort of $\Phi(A_\alpha)$ is $\bar{\mu}[\alpha]$.*

Proof By induction on $|A_\alpha|$.$\Box$

Before proceeding to the notion of an interpretation, we shall make a few comments about this definition of a translation:

- The mapping μ plays the role of U in a standard translation.
- The variables of the source theory are injectively mapped to the variables of the target theory by both a standard and LUTINS translation. In (single-sorted) first-order logic, variables all have effectively the same sort. Consequently, they can be uniformly mapped to themselves and need not be in the domain of ν in a standard translation. Although variables have different sorts in a sorted logic such as LUTINS, one might think that a variable of the form x_α could be uniformly mapped to $x_{\bar{\mu}[\alpha]}$. However, this mapping would

clearly not be injective if there exists variables $x_\alpha, x_\beta \in \mathcal{V}_1$ such that $\alpha \neq \beta$ and $\bar{\mu}[\alpha] = \bar{\mu}[\beta]$. Therefore, in our definition of Φ, the mapping of variables is specified explicitly by ν.

- Since simple type theory has a more uniform syntax than first-order logic, the definition of ν on constants is less complicated in a LUTINS translation than in a standard translation.
- A normal LUTINS translation preserves the structure of LUTINS syntax, as can be seen clearly by the definition given above.
- In some cases, a nonnormal LUTINS translation relativizes the constructor `defined-in` and the variable-binding constructors: `lambda`, `iota`, `iota-p`, `forall`, `forsome`. For each of these expression constructors, the relativization is defined in the obvious way. In particular, the relativization of `lambda` in an expression $A = \texttt{lambda}\{x^1_{\alpha_1}, \ldots, x^n_{\alpha_n} \,.\, B\}$ produces an expression beginning with `lambda` which is the appropriate restriction of the unrelativized translation of A.

Interpretations

Φ is an *interpretation of* T_1 *in* T_2 if $\Phi(A_*)$ is a theorem of T_2 for each theorem A_* of T_1. T_1 is *interpretable in* T_2 if there is an interpretation of T_1 in T_2.

A *pre-obligation* of Φ is any one of the following theorems of T_1:

1. An axiom of T_1.
2. $\texttt{forsome}\{x_\alpha \,.\, \texttt{the-true}\{\}\}$ where $\alpha \in \mathcal{A}_1$.
3. $\texttt{defined-in}\{c_\alpha, \alpha\}$ where $c_\alpha \in \mathcal{C}_1$.
4. $\texttt{forall}\{x_\alpha \,.\, \texttt{defined-in}\{x_\alpha, \xi_1(\alpha)\}\}$ where $\alpha \in \mathcal{A}_1$.

An *obligation* of Φ is any sentence $\Phi(A_*)$ where A_* is a pre-obligation of Φ. The four kinds of pre-obligations [obligations] are called, in order, *axiom*, *sort nonemptiness*, *constant sort*, and *sort inclusion* pre-obligations [obligations]. Note: The sort nonemptiness obligations of Φ are trivially theorems of T_2 if Φ is normal.

The first three kinds of obligations correspond very closely to the four kinds of obligations for a standard first-order translation (constant sort obligations correspond to both individual constant symbol and function symbol obligations). There are no standard translation obligations corresponding to sort inclusion obligations because there are no subtypes in first-order logic. Obligations serve the same purpose for LUTINS translations as they do for standard translations: if all of the obligations of a translation are theorems of the target theory, then the translation is an interpretation. This result is proved for LUTINS in Section 8.

7 Some Examples Involving Groups and Fields

In this section we construct some interpretations of a theory G of an abstract group in a theory F of an abstract field. We will employ the following abbreviations:

- $F(A_1, \ldots, A_n)$ for $\mathtt{apply}\{F, A_1, \ldots, A_n\}$.
- $A = B$ for $\mathtt{equals}\{A, B\}$.
- $A \neq B$ for $\neg\{\mathtt{equals}\{A, B\}\}$.

Define $G = ((\mathcal{A}_G, \xi_G, \mathcal{V}_G, \mathcal{C}_G), \Gamma_G)$ to be a theory of an abstract group where:

1. $\mathcal{A}_G = \{*, \alpha\}$.
2. ξ_G is the identity function.
3. $\mathcal{C}_G = \{e_\alpha, \mathrm{mul}_{[\alpha,\alpha,\alpha]}, \mathrm{inv}_{[\alpha,\alpha]}\}$.
4. Γ_G is the usual set of axioms for a group.

Define $F = ((\mathcal{A}_F, \xi_F, \mathcal{V}_F, \mathcal{C}_F), \Gamma_F)$ to be a theory of an abstract field where:

1. $\mathcal{A}_F = \{*, \beta\}$.
2. ξ_F is the identity function.
3. $\mathcal{C}_F = \{0_\beta, 1_\beta, +_{[\beta,\beta,\beta]}, \times_{[\beta,\beta,\beta]}\}$.
4. Γ_F is the usual set of axioms for a field.

Except for $\mathcal{V}_G$ and $\mathcal{V}_F$, the languages of G and F are determined by the conditions given above.

Example 7.1 Let $\Phi_1 = (\mu_1, \nu_1)$ be an interpretation of G in F such that:

1. $\mu_1(\alpha) = \beta$.
2. $\nu_1(e_\alpha) = 0_\beta$.
3. $\nu_1(\mathrm{mul}_{[\alpha,\alpha,\alpha]}) = +_{[\beta,\beta,\beta]}$.
4. $\nu_1(\mathrm{inv}_{[\alpha,\alpha]}) = \mathtt{lambda}\{x_\beta \,.\, \mathtt{iota}\{y_\beta \,.\, +_{[\beta,\beta,\beta]}(x_\beta, y_\beta) = 0_\beta\}\}$.

Φ_1 shows that the elements of a field form a group under addition.□

Example 7.2 Let $\Phi_2 = (\mu_2, \nu_2)$ be an interpretation of G in F such that:

1. $\mu_2(\alpha) = \mathtt{lambda}\{x_\beta \,.\, x_\beta \neq 0_\beta\}$.
2. $\nu_2(e_\alpha) = 1_\beta$.
3. $\nu_2(\mathrm{mul}_{[\alpha,\alpha,\alpha]}) = \mathtt{lambda}\{x_\beta, y_\beta \,.\, \mathtt{if}\{\mathtt{and}\{x_\beta \neq 0_\beta, y_\beta \neq 0_\beta\},$
   $\times_{[\beta,\beta,\beta]}(x_\beta, y_\beta),$
   $\mathtt{undefined}\{\beta\}\}\}$.
4. $\nu_2(\mathrm{inv}_{[\alpha,\alpha]}) = \mathtt{lambda}\{x_\beta \,.\, \mathtt{iota}\{y_\beta \,.\, \times_{[\beta,\beta,\beta]}(x_\beta, y_\beta) = 1_\beta\}\}$.

Φ_2 shows that the nonzero elements of a field form a group under multiplication. Since α is associated with a quasi-sort, Φ_2 is nonnormal and thus relativizes the quantifiers in expressions of G. Notice that $\mathrm{mul}_{[\alpha,\alpha,\alpha]}$ is associated with the restriction of $\times_{[\beta,\beta,\beta]}$ to the nonzero elements of β, and $\nu_2(\mathrm{inv}_{[\alpha,\alpha]})$ is undefined at 0_β.□

Now define $F' = ((\mathcal{A}_F \cup \{\gamma\}, \xi_{F'}, \mathcal{V}_{F'}, \mathcal{C}_F), \Gamma_F \cup \{A_*\})$ to be an extension of F such that:

1. $\xi_{F'}(\gamma) = \beta$.
2. $A_* = \texttt{forall}\{x_\beta \,.\, \texttt{iff}\{\texttt{defined-in}\{x_\beta, \gamma\}, x_\beta \neq 0_\beta\}\}$.

Thus F' is obtained by defining in F an atomic sort γ with the same extension as the quasi-sort $\texttt{lambda}\{x_\beta \,.\, x_\beta \neq 0_\beta\}$.

Example 7.3 Let $\Phi_3 = (\mu_3, \nu_3)$ be an interpretation of G in F' such that:

1. $\mu_3(\alpha) = \gamma$.
2. $\nu_3(e_\alpha) = 1_\beta$.
3. $\nu_3(\mathrm{mul}_{[\alpha,\alpha,\alpha]}) = \texttt{lambda}\{x_\gamma, y_\gamma \,.\, \times_{[\beta,\beta,\beta]}(x_\gamma, y_\gamma)\}$.
4. $\nu_3(\mathrm{inv}_{[\alpha,\alpha]}) = \texttt{lambda}\{x_\gamma \,.\, \texttt{iota}\{y_\gamma \,.\, \times_{[\beta,\beta,\beta]}(x_\gamma, y_\gamma) = 1_\beta\}\}$.

Like Φ_2, Φ_3 shows that the nonzero elements of a field form a group under multiplication. In a certain sense, Φ_2 and Φ_3 are two renditions of the same interpretation. However, since Φ_3 associates α with a sort, Φ_3 is syntactically more economical than Φ_2. In particular, Φ_3 is normal while Φ_2 is nonnormal.$\Box$

Examples 7.2 and 7.3 together illustrate how a nonnormal translation can be transformed into a normal translation. Since LUTINS admits subtypes, any nonnormal translation can be "normalized" in this way as long as one is willing to define new atomic sorts. The general idea is as follows. Suppose the translation Φ is nonnormal. For each $\alpha \in \mathcal{A}_1$ such that $\mu(\alpha) \notin \mathcal{S}_2$, (1) add to T_2 a new atomic sort α' and a new axiom which says that $\mu(\alpha)$ and α' are coextensional and (2) redefine μ so that $\mu(\alpha) = \alpha'$. The resulting translation is obviously normal. (This construction is described in detail in the next section.) Hence, nonnormal translations are unnecessary in LUTINS if there is no restriction on defining atomic sorts. However, in practice the strict avoidance of nonnormal translations can easily lead to theories with a large number of atomic sorts which are almost never used.

8 Interpretation and Satisfiability Theorems

We prove in this section the interpretation and relative satisfiability theorems for LUTINS. The former establishes a sufficient condition for a translation to be an interpretation, and the latter says that a theory which is interpretable in a satisfiable theory is itself satisfiable. They correspond, respectively, to the Standard Interpretation Theorem (Theorem 2.1) and Standard Relative Satisfiability (Theorem 2.2) for first-order logic. The proofs are based on the interpretation and relative satisfiability theorems for $\mathbf{PF}^*$ proved in [9].

Let $T_i = (\mathcal{L}_i, \Gamma_i)$ be a LUTINS theory for $i = 1, 2$.

$\mathbf{PF}^*$ Theories and Translations

Given a LUTINS theory $T = (\mathcal{L}, \Gamma)$, let $\widetilde{T} = (\widetilde{\mathcal{L}}, \{\zeta(A_*) : A_* \in \Gamma\})$, which is a $\mathbf{PF}^*$ theory.

Proposition 8.1 *Let $T = (\mathcal{L}, \Gamma)$ be a* LUTINS *theory and $\mathcal{M}$ be a model for $\mathcal{L}$. Then:*

1. *$\mathcal{M}$ is a model for T iff $\mathcal{M}$ is a standard model for $\widetilde{T}$.*
2. *$T \models A_*$ iff $\widetilde{T} \models \zeta(A_*)$, for all sentences $A_* \in \mathcal{E}$.*

Proof Let φ be a $\mathcal{V}$-assignment into $\mathcal{M}$ and A_* be a formula of $\mathcal{L}$. Then, by definition, $U_\varphi^{\mathcal{M}}(A_*) = V_\varphi^{\mathcal{M}}(\zeta(A_*))$. This identity implies (1), and (1) and the identity imply (2).□

Suppose $\Phi_{\mathrm{n}} = (\mu, \nu)$ is a normal LUTINS translation from T_1 to T_2. Let $\widetilde{\Phi_{\mathrm{n}}} = (\mu, \widetilde{\nu})$ where $\widetilde{\nu} : \mathcal{V}_1 \cup \widetilde{\mathcal{C}}_1 \to \widetilde{\mathcal{E}}_2$ is defined by:

1. For all $x_\alpha \in \mathcal{V}_1$, $\widetilde{\nu}(x_\alpha) = \nu(x_\alpha)$.
2. For all $c_\alpha \in \mathcal{C}_1$, $\widetilde{\nu}(c_\alpha) = \zeta(\nu(c_\alpha))$.
3. For all $\alpha \in \mathcal{S}_1$, $\widetilde{\nu}(=_{[\alpha,\alpha,*]}) = =_{[\bar{\mu}(\alpha),\bar{\mu}(\alpha),*]}$ and $\widetilde{\nu}(\iota_{[[\alpha,*],\alpha]}) = \iota_{[[\bar{\mu}(\alpha),*],\bar{\mu}(\alpha)]}$.

$\widetilde{\Phi_{\mathrm{n}}}$ is easily verified to be a $\mathbf{PF}^*$ translation from $\widetilde{T_1}$ to $\widetilde{T_2}$. For $E \in \widetilde{\mathcal{E}_1}$, let $\widetilde{\Phi_{\mathrm{n}}}(E) = \widetilde{\nu}(E)$, the translation of E via $\widetilde{\Phi_{\mathrm{n}}}$.

The key lemma of this section is

Lemma 8.2 *Let Φ be a normal* LUTINS *translation from T_1 to T_2 such that each of its constant sort and sort inclusion obligations is a theorem of T_2. Also, let $\mathcal{M}_2$ be a model for T_2 and φ be a $\mathcal{V}_2$-assignment into $\mathcal{M}_2$. Then:*

1. *$V_\varphi^{\mathcal{M}_2}(\zeta(\Phi(E))) = V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(E)))$, for all expressions $E \in \mathcal{E}_1$.*
2. *$\widetilde{T_2} \models \zeta(\Phi(A_*))$ iff $\widetilde{T_2} \models \widetilde{\Phi}(\zeta(A_*))$, for all sentences $A_* \in \mathcal{E}_1$.*

The proof of this lemma requires the following technical lemma:

Lemma 8.3 *Let $\Phi = (\mu, \nu)$ be a normal* LUTINS *translation from T_1 to T_2 such that each of its constant sort and sort inclusion obligations is a theorem of T_2. Also, let $\mathcal{M}_2$ be a model for T_2, φ be a $\mathcal{V}_2$-assignment into $\mathcal{M}_2$, and $A_\alpha, B_\beta, C_\gamma, X_* \in \mathcal{E}_1$. Then:*

1. *If $\square \in \{=, \simeq\}$, δ is the type of $\Phi(A_\alpha)$, and $\epsilon = \bar{\mu}(\tau(\alpha))$, then $V_\varphi^{\mathcal{M}_2}(E^\delta) = V_\varphi^{\mathcal{M}_2}(E^\epsilon)$ where*
$$E^\theta = \zeta(\Phi(A_\alpha)) \,\square_\theta\, \zeta(\Phi(B_\beta)).$$
2. *If δ is the least upper bound in $\preceq_{\xi_2}$ of the sorts of $\Phi(B_\beta)$ and $\Phi(C_\gamma)$, and $\epsilon = \bar{\mu}(\beta \sqcup_{\xi_2} \gamma)$, then $V_\varphi^{\mathcal{M}_2}(E^\delta) = V_\varphi^{\mathcal{M}_2}(E^\epsilon)$ where*
$$E^\theta = \mathrm{I}\{x_\theta \,.\, (\zeta(\Phi(X_*)) \supset (x_\theta =_\theta \zeta(\Phi(B_\beta)))) \wedge (\neg\zeta(\Phi(X_*)) \supset (x_\theta =_\theta \zeta(\Phi(C_\gamma))))\}.$$

3. If δ is the sort of $\Phi(A_\alpha)$ and $\epsilon = \bar{\mu}(\alpha)$, then $V_\varphi^{\mathcal{M}_2}(E^\delta) = V_\varphi^{\mathcal{M}_2}(E^\epsilon)$ where

$$E^\theta = (\zeta(\Phi(A_\alpha)) \downarrow \theta).$$

Proof Follows from the lemmas in Section 11 of [9].□

Proof of Lemma 8.2 Part (2) follows from part (1) by Proposition 8.1.

Let $\Phi = (\mu, \nu)$ and $E \in \mathcal{E}_1$. The proof of (1) is by induction on $|E|$.

Basis. Assume $|E| = 0$; then $E \in \mathcal{V}_1 \cup \mathcal{C}_1$.

Let $E \in \mathcal{V}_1$. Then

$$\begin{aligned} V_\varphi^{\mathcal{M}_2}(\zeta(\Phi(E))) &= V_\varphi^{\mathcal{M}_2}(\Phi(E)) & (1)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(E)) & (2)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(E))) & (3) \end{aligned}$$

(1) and (3) hold since ζ is the identity on $\mathcal{V}_1$, and (2) holds since $\Phi = \widetilde{\Phi}$ on $\mathcal{V}_1$.

Let $E \in \mathcal{C}_1$. Then

$$\begin{aligned} V_\varphi^{\mathcal{M}_2}(\zeta(\Phi(E))) &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(E)) & (4)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(E))) & (5) \end{aligned}$$

(4) holds since $\zeta \circ \Phi = \widetilde{\Phi}$ on $\mathcal{C}_1$, and (5) holds since ζ is the identity on $\mathcal{C}_1$.

Induction step. Assume $|E| > 0$; then E is a compound expression beginning with the expression constructor □.

Let $E = \texttt{defined-in}\{A_\alpha, \beta\}$. Then

$$\begin{aligned} & V_\varphi^{\mathcal{M}_2}(\zeta(\Phi(\texttt{defined-in}\{A_\alpha, \beta\}))) & \\ &= V_\varphi^{\mathcal{M}_2}(\zeta(\texttt{defined-in}\{\Phi(A_\alpha), \bar{\mu}(\beta)\})) & (6)\\ &= V_\varphi^{\mathcal{M}_2}(\zeta(\Phi(A_\alpha)) \downarrow \bar{\mu}(\beta)) & (7)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(A_\alpha)) \downarrow \bar{\mu}(\beta)) & (8)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(A_\alpha) \downarrow \beta)) & (9)\\ &= V_\varphi^{\mathcal{M}_2}(\widetilde{\Phi}(\zeta(\texttt{defined-in}\{A_\alpha, \beta\}))) & (10) \end{aligned}$$

(6) is by the definition of the application of a normal LUTINS translation to an expression; (7) and (10) are by the definition of ζ; (8) is by the induction hypothesis; and (9) is by Lemma 11.5 in [9].

The derivation is very similar when □ is any one of the other expression constructors except for `equals`, `quasi-equals`, `if`, or `is-defined`. The derivations for these last four expression constructors require Lemma 8.3.

This completes the proof.□

A Special Case of the Interpretation Theorem

Lemma 8.2 and the Semantic Interpretation Theorem for $\mathbf{PF}^*$ imply the Interpretation Theorem for LUTINS restricted to normal translations:

Theorem 8.4 *A normal* LUTINS *translation from T_1 to T_2 is an interpretation if each of its obligations is a theorem of T_2.*

Proof Let Φ be a normal LUTINS translation from T_1 to T_2 such that each of its obligations is a theorem of T_2.

We begin by showing that $\widetilde{\Phi}$ is an interpretation. Let O be an obligation of $\widetilde{\Phi}$. Then O has the form $\widetilde{\Phi}(\zeta(A_*))$ where A_* is an axiom, constant sort, or sort inclusion pre-obligation of Φ. By hypothesis, $T_2 \models \Phi(A_*)$. Then

$$\begin{aligned} T_2 \models \Phi(A_*) \text{ iff } \widetilde{T_2} &\models \zeta(\Phi(A_*)) && (11)\\ \text{iff } \widetilde{T_2} &\models \widetilde{\Phi}(\zeta(A_*)) && (12)\end{aligned}$$

(11) holds by Proposition 8.1, and (12) holds by Lemma 8.2. Hence, each obligation of $\widetilde{\Phi}$ is a theorem of $\widetilde{T_2}$, and so $\widetilde{\Phi}$ is a $\mathbf{PF}^*$ interpretation by the Semantic Interpretation Theorem for $\mathbf{PF}^*$ [9, Theorem 12.4].

Let A_* be a theorem of T_1. Then

$$\begin{aligned} T_1 \models A_* \quad \text{iff} \quad \widetilde{T_1} &\models \zeta(A_*) && (13)\\ \text{implies } \widetilde{T_2} &\models \widetilde{\Phi}(\zeta(A_*)) && (14)\\ \text{iff} \quad \widetilde{T_2} &\models \zeta(\Phi(A_*)) && (15)\\ \text{iff} \quad T_2 &\models \Phi(A_*) && (16)\end{aligned}$$

(13) and (16) hold by Proposition 8.1; (14) holds because $\widetilde{\Phi}$ is a $\mathbf{PF}^*$ interpretation; and (15) holds by Lemma 8.2. Therefore, the sequence of implications shows that Φ maps each theorem of T_1 to a theorem of T_2, that is, that Φ is an interpretation.□

The Translation Φ' and Its Properties

Before we can strengthen Theorem 8.4 to the full interpretation theorem for LUTINS, we must show how an arbitrary LUTINS translation can be transformed into a normal translation.

Let $\Phi = (\mu, \nu)$ be a LUTINS translation from T_1 to T_2. The key idea is to add new atomic sorts to T_2. Let $T_2' = (\mathcal{L}_2', \Gamma_2')$ be the theory

$$((\mathcal{A} \cup \mathcal{A}_2, \xi \cup \xi_2, \mathcal{V} \cup \mathcal{V}_2, \mathcal{C}_2), \Gamma \cup \Gamma_2)$$

such that:

1. $\mathcal{A} = \{\gamma^{\mu(\alpha)} : \alpha \in \mathcal{A}_1 \text{ and } \mu(\alpha) \in \mathcal{Q}_2\}$.
2. $\xi : \mathcal{A} \rightarrow \mathcal{A}_2$ and $\xi(\gamma^{Q_{[\alpha,*]}}) = \alpha$.
3. $\mathcal{V}$ is an appropriate set of new variables.

4. $\Gamma = \{A^{Q_{[\alpha,*]}} : \gamma^{Q_{[\alpha,*]}} \in \mathcal{A}\}$ where $A^{Q_{[\alpha,*]}}$ is

$$\texttt{forall}\{x_{\tau(\alpha)}\,.\;\texttt{iff}\{\texttt{defined-in}\{x_{\tau(\alpha)}, \gamma^{Q_{[\alpha,*]}}\}, \texttt{apply}\{Q_{[\alpha,*]}, x_{\tau(\alpha)}\}\}\}.$$

T_2' is a "definitional" extension of T_2; its new atomic sorts are axiomatically defined to be coextensional with the quasi-sorts in the range of μ. Consequently, the following proposition is easy to prove.

Proposition 8.5 *Suppose each sort nonemptiness obligation of Φ is a theorem of T_2. Then:*

1. *Each model for T_2 expands to a model for T_2'.*
2. *Each model for T_2' reduces to a model for T_2.*

Let $\Phi' = (\mu', \nu')$ be a translation from T_1 to T_2' such that:

1. If $\mu(\alpha) \in \mathcal{S}_2$, then $\mu'(\alpha) = \mu(\alpha)$.
2. If $\mu(\alpha) \in \mathcal{Q}_2$, then $\mu'(\alpha) = \gamma^{\mu(\alpha)}$.
3. If $c_\alpha \in \mathcal{C}_1$, then $\nu'(c_\alpha) = \nu(c_\alpha)$.

Φ' is clearly normal. Notice that $\Phi \neq \Phi'$ on $\mathcal{V}_1$ unless Φ is normal.

Lemma 8.6 *Let $\mathcal{M}_2$ and $\mathcal{M}_2'$ be models for T_2 and T_2', respectively, such that $\mathcal{M}_2'$ is an expansion of $\mathcal{M}_2$. Then*

$$U_\varphi^{\mathcal{M}_2}(\Phi(E)) = U_{\varphi'}^{\mathcal{M}_2'}(\Phi'(E))$$

for all $E \in \mathcal{E}_1$, $\mathcal{V}_2$-assignments φ into $\mathcal{M}_2$, and $\mathcal{V}_2'$-assignments φ' into $\mathcal{M}_2'$ such that $\varphi(\Phi(x_\alpha)) = \varphi'(\Phi'(x_\alpha))$ for all $x_\alpha \in \mathcal{V}_1$.

Proof By induction on $|E|$.□

Lemma 8.7 *Suppose each sort nonemptiness obligation of Φ is a theorem of T_2. Then*

$$T_2 \models \Phi(A_*) \quad \textit{iff} \quad T_2' \models \Phi'(A_*)$$

for all sentences $A_* \in \mathcal{E}_1$.

Proof By Proposition 8.5 and Lemma 8.6.□

The Theorems

Theorem 8.8 (Interpretation Theorem) *A translation from T_1 to T_2 is an interpretation if each of its obligations is a theorem of T_2.*

Proof Let Φ be a LUTINS translation from T_1 to T_2 such that each of its obligations is a theorem of T_2. As above, construct the theory T_2' from T_2 and the normal translation Φ' from Φ. Since Φ and Φ' have the same source theory, they also have the same pre-obligations. Hence, each obligation of Φ' is a theorem of T_2' by Lemma 8.7. Φ' is normal, so Φ' is an interpretation by Theorem 8.4. This implies that Φ is an interpretation by Lemma 8.7.□

Theorem 8.9 (Relative Satisfiability) *If T_1 is interpretable in T_2 and T_2 is satisfiable, then T_1 is satisfiable.*

Proof Let Φ be a LUTINS interpretation of T_1 in T_2 and $\mathcal{M}_2$ be a model for T_2. As above, construct the theory T_2' from T_2 and the normal translation Φ' from Φ. By Proposition 8.5, there is a model $\mathcal{M}_2'$ for T_2' which is an expansion of $\mathcal{M}_2$ since Φ is an interpretation. $\mathcal{M}_2'$ is also a standard model for $\widetilde{T_2'}$ by Proposition 8.1. By Lemma 8.7, Φ' is an interpretation, and so by the proof of Theorem 8.4, $\widetilde{\Phi'}$ is a also an interpretation. Therefore, there is a standard model $\mathcal{M}_1$ for $\widetilde{T_1}$ by Relative Satisfiability for $\mathbf{PF}^*$ [9, Theorem 12.3]. $\mathcal{M}_1$ is also a model for T_1 by Proposition 8.1, so T_1 is satisfiable.□

9 Conclusion

In this paper we have developed a method for theory interpretation in LUTINS, a version of simple type theory which supports partial functions and subtypes. The method embodies the principal characteristics of the standard approach to theory interpretation in first-order logic:

- A translation is a kind of homomorphism that is determined by how the primitive symbols of the source theory are associated with objects of the target theory.
- When sort symbols are associated with unary predicates, the translation of an expression may involve the relativization of variable-binding constructors such as the universal and existential quantifiers.
- A translation is an interpretation if the "obligations" of the translation are theorems of the target theory (interpretation theorem).
- A theory which is interpretable in a satisfiable theory is itself satisfiable (relative satisfiability).

Since the method is based on a logic with partial functions and subtypes, it has two advantages over the standard first-order approach:

- The functions with restricted domains that naturally arise from interpretations which associate base types with subtypes are handled directly as partial functions.
- Relativization of variable-binding constructors can be avoided completely, provided that appropriate sort symbols are defined.

The method is intended as a guide for theory interpretation in classical simple type theories as well as in predicate logics which admit partial functions.

Acknowledgments

I am grateful for the suggestions received from the referees. Many of the ideas in this paper grew out of conversations I have had with Dr. Joshua Guttman and Dr. Javier Thayer in the course of developing IMPS.

References

1. P. B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press, 1986.
2. P. B. Andrews, S. Issar, D. Nesmith, and F. Pfenning. The TPS theorem proving system (system abstract). In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 641–642. Springer-Verlag, 1990.
3. C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, 1990.
4. A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.
5. D. Craigen, S. Kromodimoeljo, I. Meisels, A. Neilson, B. Pase, and M. Saaltink. m-EVES: A tool for verifying software. In *11th International Conference on Software Engineering (ICSE'11)*, Singapore, 1988.
6. D. Craigen, S. Kromodimoeljo, I. Meisels, B. Pase, and M. Saaltink. EVES: An overview. Technical Report CP-91-5402-43, ORA Corporation, 1991.
7. H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.
8. W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–1291, 1990.
9. W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.
10. W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: System description. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag, 1992.
11. W. M. Farmer, J. D. Guttman, and F. J. Thayer. Little theories. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 567–581. Springer-Verlag, 1992.
12. W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: an Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.
13. F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence*, 57:323–389, 1992.
14. J. A. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-99-9, SRI International, August 1988.
15. M. J. C. Gordon. HOL: A proof generating system for higher-order logic. In G. Birtwistle and P. A. Surahmanyam, editors, *VLSI Specification, Verification, and Synthesis*, pages 73–128. Kluwer, 1987.
16. J. D. Guttman. A proposed interface logic for verification environments. Technical Report M91-19, The MITRE Corporation, 1991.
17. R. W. Harper and F. Pfenning. A module system for a programming language based on the LF logical framework. *Journal of Functional Programming.* Forthcoming.
18. D. Hilbert. *The Foundations of Geometry.* Open Court, Chicago, 1902.
19. K. Kunen. *Set Theory: An Introduction to Independence Proofs.* North-Holland, 1980.
20. B. H. Levy. *An Approach to Compiler Correctness Using Interpretation Between Theories.* PhD thesis, University of California, Los Angeles, 1986. Also Technical Report ATR-86(8454)-4, The Aerospace Corporation, El Segundo, California.
21. T. S. E. Maibaum, P. A. S. Veloso, and M. R. Sadler. A theory of abstract data types for program development: Bridging the gap? In H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, editors, *Formal Methods and Software Development,*

Volume 2, volume 186 of *Lecture Notes in Computer Science*, pages 214–230. Springer-Verlag, 1985.
22. J. D. Monk. *Mathematical Logic*. Springer-Verlag, 1976.
23. J. Mycielski. A lattice of interpretability types of theories. *Journal of Symbolic Logic*, 42:297–305, 1977.
24. R. Nakajima and T. Yuasa, editors. *The* IOTA *Programming System*, volume 160 of *Lecture Notes in Computer Science*. Springer-Verlag, 1982.
25. T. Nipkow and L. C. Paulson. Isabelle-91. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 673–676. Springer-Verlag, 1992.
26. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer-Verlag, 1992.
27. J. Rushby, F. von Henke, and S. Owre. An introduction to formal specification and verification using EHDM. Technical Report SRI-CSL-91-02, SRI International, 1991.
28. J. R. Shoenfield. *Mathematical Logic*. Addison-Wesley, 1967.
29. D. R. Smith and M. R. Lowry. Algorithmic theories and design tactics. *Science of Computer Programming*, 14:305–321, 1990.
30. L. W. Szczerba. Interpretability of elementary theories. In R. E. Butts and J. Hintikka, editors, *Logic, Foundations of Mathematics, and Computability Theory*, pages 129–145. Reidel, 1977.
31. L. W. Szczerba. Interpretability and axiomatizability. *Bulletin de L'Académie Polonaise des Sciences*, 27:425–429, 1979.
32. A. Tarski, A. Mostowski, and R. M. Robinson. *Undecidable theories*. North-Holland, 1953.
33. W. M. Turski and T. S. E. Maibaum. *The Specification of Computer Programs*. Addison-Wesley, 1987.
34. J. van Bentham and D. Pearce. A mathematical characterization of interpretation between theories. *Logica Studia*, 43:295–303, 1984.
35. P. J. Windley. Formal modeling and verification of microprocessors. *IEEE Transactions on Computers*. Forthcoming.
36. P. J. Windley. Abstract theories in HOL. In L. Claesen and M. J. C. Gordon, editors, *Proceedings of the 1992 International Workshop on the* HOL *Theorem Prover and its Applications*. North-Holland, November 1992.

This article was processed using the LaTeX macro package with LLNCS style