Abstract

The purpose of a logical framework such as LF is to provide a language for defining logical systems suitable for use in a logic-independent proof development environment. All inferential activity in an object logic (in particular, proof search) is to be conducted in the logical framework via the representation of that logic in the framework. An important tool for controlling search in an object logic is the use of structured theory presentations. In order to apply these ideas to the setting of a logical framework, we study the behavior of structured theory presentations under representation in a framework, focusing on the problem of “lifting” presentations from the object logic to the metalogic of the framework. We also consider imposing structure on logic presentations so that logical systems may themselves be defined in a modular fashion. This opens the way to a CLEAR-like language for defining both theories and logics in a logical framework.

1 Introduction

The Logical Framework (LF) [12] is a language for defining formal systems. The language is a three-level typed λ-calculus with II-types, closely related to the AUTOMATH type theories [7,20]. A formal system is specified by giving an LF signature, a finite list of constant declarations that specifies the syntax, judgement forms, and inference rules of the system.

All of the syntactic apparatus of a formal system, including proofs, are represented as LF terms. The LF type system is sufficiently expressive to capture the uniformities of a large class of logical systems of interest to computer science, including notions of schematic rules and proofs, derived rules of inference, and higher-order judgement forms expressing consequence and generality.

According to the methodology of [12,2], a necessary condition for the correctness of an encoding of an object logic \( \mathcal{L} \) in LF is that the consequence relation \( \vdash_{\mathcal{L}} \) of \( \mathcal{L} \) be fully and faithfully embedded in the consequence relation \( \vdash_{LF} \) of LF by an encoding of the syntax of \( \mathcal{L} \) as LF terms. (The consequence relation of LF is given by considering type inhabitation assertions, as in NuPRL [6].) By focusing on the embedding of consequence relations LF may be viewed as a “universal metalogic” in which all inferential activity is to be conducted: object logics “exist” (for the purposes of implementation) only insofar as they are encodable in LF.

One important form of inferential activity in a logical system \( \mathcal{L} \) is proof search: given a set of axioms or assumptions \( \Phi \) and a conjecture \( \phi \), determine whether or not \( \Phi \vdash_{\mathcal{L}} \phi \). In keeping with the view of LF as a universal metalogic, search in \( \mathcal{L} \) is to be reduced to search in LF via the encoding of \( \mathcal{L} \) in LF. This entails “lifting” operations on the search space of \( \mathcal{L} \) to operations on the search space of LF. Numerous interesting questions arise in the process of carrying out this program. Pym [15] considers a variety of issues related to search, in particular the definition of a unification algorithm and methods for inducing metalogical search operations from LF signatures. Elliott [8] has also developed a unification algorithm for LF, and Pfenning [14] bases a logic programming language on it.

In order to facilitate the process of proof search, it is important to consider “modular” or “structured”...
theory presentations that provide the basis for guiding search in large theories. This approach was first considered in [19] in the context of LCF. An LCF theory is presented by declaring base types, constants, and function symbols (i.e., by giving an LCF signature), and by giving a set of axioms over the language induced by these declarations. The fundamental idea is to exploit the invariance of consequence in large theories. This approach was first introduced in the context of LCF with the notion of a signature morphism. The language of structured presentations considered here (and here) uses signature morphisms to mediate the combination of theories and to provide a form of “information hiding.” The primitives of the presentation language are sufficient for the definability of a variety of interesting constructions such as instantiation of parametric presentations. (See [19] and Section 3 below for more details.) One purpose of this paper is to consider these ideas in the context of a universal metalogic such as LF. We focus on “lifting” structured presentations from the level of the object logic to the level of the metalogic, in particular, on the conditions under which structured search in the metalogic may be specialized to structured search in an object logic theory.

Another important aspect of LF is that it opens up the possibility of using several logical systems at once. For example, one may view the encoding of S4 modal logic given in [2] as a combination of the truth and validity consequence relations of S4. In this paper we develop the basic machinery of a language of structured logic presentations that allows for “putting together logics,” just as structured theory presentations provide the machinery for “putting together theories.” This machinery may be used to formalize examples such as adding a connective to a logic, or the parameterization of Hoare logic by the logic of assertions.

This paper is organized as follows. In Section 2 we introduce a general definition of a logical system as a family of consequence relations indexed by signatures that satisfies a certain uniformity condition with respect to change of signature. This resembles the formalization of a logical system as an institution from [10]; the crucial difference is that institutions present a model-theoretic view of logical systems while our formulation is centered directly around the notion of a consequence relation. (See also [9].) The sorts of consequence relations that we consider are motivated by the strictures of encoding in LF, and thus are limited to one-sided consequence relations that are closed under weakening, permutation, contraction, and cut, and which satisfy compactness. Generalizing the methodology of [12], we introduce the notion of a representation of one logical system in another, taking account of variability in signatures. In Section 3 we consider structured presentations in an arbitrary logical system. Structured presentations denote theories (sets of sentences closed under consequence), and the structure of the presentations induces a natural search procedure guided by this structure. In Section 4 we consider the problem of “lifting” a structured presentation along a representation of one logical system in another. Structured presentations may not be simply translated via the representation and used in the target logic. Instead, we define a notion of search that is conditioned by the representation, and give conditions under which we may achieve the goal of working entirely within the metalogic. In Section 5 we introduce the metalogic of interest, LF, as a logical system, and define the notion of a logic presentation. A logic presentation is essentially an LF signature (with an indication of which terms encode the judgements of the object logic), together with a representation of the object logic in the logical system given by the presentation. In Section 6 we explore the coinduction construction as a tool for building logics in a structured way. Finally, in Section 7 we suggest directions for future research.

2 Consequence Relations and Logical Systems

Our treatment of logical systems centers on consequence relations (see [1] for a survey). We take a consequence relation to be a binary relation between finite subsets and elements of a set of “sentences” satisfying three conditions to be given below. We use $\phi$ and $\psi$ to range over sentences, $\Phi$ to range over arbitrary sets of sentences, and $\Delta$ to range over finite sets of sentences. We write $\Delta, \Delta'$ for union, and write $\phi, \Delta$ for $\{\phi\}, \Delta$. If $s : S_1 \to S_2$ is a function, then the extension of $s$ to subsets of $S_1$ is denoted by $s$ as well. Function application will often be denoted by concatenation, e.g., $s \phi$ stands for $s(\phi)$.

Definition 2.1 A consequence relation (CR) is a pair $(S, \vdash)$ where $S$ is a set of sentences and $\vdash \subseteq \text{Fin}(S) \times S$ is a binary relation such that

1. (Reflexivity) $\phi \vdash \phi$;
2. (Transitivity) If $\Delta \vdash \phi$ and $\phi, \Delta' \vdash \psi$, then $\Delta, \Delta' \vdash \psi$.
3. (Weakening) If $\Delta \vdash \psi$, then $\phi, \Delta \vdash \psi$. 

227
The choice of conditions on consequence relations is motivated by the intention to consider those that are encodable in LF.

Let \((S, \vdash)\) be a consequence relation. If \(S' \subseteq S\), then \((S', \vdash)\restriction S'\) is defined to be the consequence relation \((S', \vdash \restriction \Fin(S') \times S')\). The closure \(\Phi\) of a set \(\Phi \subseteq S\) of sentences under \(\vdash\) is defined in the usual way. A set \(\Phi \subseteq S\) is a theory (wrt \(\vdash\)) iff \(\Phi = \Phi\).

**Definition 2.2** A morphism of consequence relations (CR morphism) \(s : (S_1, \vdash_1) \to (S_2, \vdash_2)\) is a function \(s : S_1 \to S_2\) (the translation of sentences) such that if \(\Delta \vdash_1 \phi\), then \(s\Delta \vdash_2 s\phi\). Identity and composition are inherited from the category of sets. The CR morphism \(s\) is an inclusion if it is an inclusion as a function in the category of sets, and is conservative if \(\Delta \vdash_1 \phi\) whenever \(s\Delta \vdash_2 s\phi\). CR is the category whose objects are consequence relations and whose morphisms are CR morphisms.

CLEAR-like techniques for structuring theory presentations are based on the separation between the language of a theory and the set of axioms that generates it [4]. We therefore consider a logical system to be a family of consequence relations indexed by a collection of signatures which determine the language of a theory. Moreover, it is important for the development that consequence be preserved under variation in signature (for example, renaming constants or replacing constants by terms over another signature). This leads to the following definition:

**Definition 2.3** A logical system, or logic, is a functor \(L : \Sig \to \Cr\).

The category \(\Sig\) is called the category of signatures of \(L\), with objects denoted by \(\Sigma\) and morphisms by \(\sigma : \Sigma_1 \to \Sigma_2\). A signature morphism \(\sigma : \Sigma_1 \to \Sigma_2\) is to be thought of as specifying a "relative interpretation" of the language defined by \(\Sigma_1\) into the language defined by \(\Sigma_2\). Writing \(L(\Sigma) = (L_{\Sigma_1}, \vdash_{\Sigma_2})\), the definition of logical system implies that if \(\sigma : \Sigma_1 \to \Sigma_2\) and \(\Delta \vdash_{\Sigma_2} \phi\), then \(L(\sigma)(\Delta) \vdash_{\Sigma_2} L(\sigma)(\phi)\). The function \(L(\sigma)\) underlying the CR morphism is called the translation function induced by \(\sigma\). To simplify notation, we write \(\sigma(\phi)\) for \(L(\sigma)(\phi)\) and \(\sigma(\Delta)\) for \(L(\sigma)(\Delta)\) when no confusion is likely.

A logical system \(L\) has inclusions iff the objects of \(\Sig\) are pre-ordered by a distinguished subcategory of morphisms, which will be referred to as inclusions, and \(L\) maps signature inclusions to inclusions of consequence relations. Inclusions are designated by \(\iota : \Sigma_1 \hookrightarrow \Sigma_2\). The requirement that \(L\) preserve inclusions means that if \(\iota : \Sigma_1 \hookrightarrow \Sigma_2\) and \(\Delta \vdash_{\Sigma_2} \phi\), then \(\Delta \vdash_{\Sigma_1} \phi\). If \(C\) is a category with a distinguished preorder subcategory of inclusions, then we say that \(C\) has pushouts along inclusions iff whenever \(f : A \to A'\) and \(\iota : A \hookrightarrow A''\) are morphisms of \(C\), the pushout of \(f\) and \(\iota\) exists, and, moreover, the morphism opposite the inclusion in the pushout diagram is itself an inclusion:

Although we usually designate the pushout object and morphisms as in the above diagram, we do not require that either \(p(f, A'')\) or \(f'' A''\) to be chosen as a function of \(f\) and \(A''\) (cf. [5]).

A simple example of a logical system is many-sorted equational logic. Let \(\Sig\) be the category of many-sorted algebraic signatures as defined in [10], with inclusions given by containment. Let \(\Sigma = (S, \Omega)\) be an algebraic signature. Define the set \(Eq(\Sigma)\) of \(\Sigma\)-equations to be the set of sentences \(\forall X.t_1 = t_2\) where \(X\) is an \(S\)-indexed family of mutually disjoint finite sequences of variables and \(t_1, t_2\) are \(\Sigma\)-terms of the same sort with variables from \(X\). Equations with no variables will be called ground equations, and are written without a quantifier. The consequence relation \((Eq(\Sigma), \vdash)\) is defined by the usual rules of equational deduction [11]. This definition extends to a functor \(\Leq : \Sig \to \Cr\) by taking \(\Leq(\sigma)\) to be the usual extension of \(\sigma\) to algebraic terms. The functor \(\Leq : \Sig \to \Cr\) is defined to be the restriction of \(\Leq\) to ground equations.

**Proposition 2.4**

1. \(\Leq\) and \(\Leq \Sig\) are logical systems with inclusions.
2. \(\Sig\) has pushouts along inclusions (in fact, is co-complete).

**Definition 2.5** A morphism of logics \(\gamma : L \to L'\) is a pair \((\gamma^{\sigma}, \gamma^{\sigma'})\) where \(\gamma^{\sigma} : \Sig \to \Sig'\) is a func-
tor and $\gamma^R: L \rightarrow \log$ is a natural transformation. The identity is the pair consisting of the identity functor on $\log$ and the identity natural transformation on $L$. Composition is defined by

$$ (\gamma_1^R, \gamma_1^S; \gamma_2^R; \gamma_2^S) = (\gamma_1^R(\gamma_2^S); \gamma_1^S; \gamma_2^R; \gamma_1^R(\gamma_2^S)). $$

$\log$ is the category of logics and logic morphisms.

A morphism of logics is to be thought of as an "encoding" of one logical system in another in such a way that consequence is preserved. Let $\gamma: L \rightarrow L'$ be a morphism of logics. The requirement that $\gamma^R$ be a natural transformation may be expressed by the equation

$$ \gamma^R(\sigma(\phi)) = \gamma^S(\sigma)(\gamma^R(\phi)). $$

In words: it doesn't matter whether we encode the translation $\sigma(\phi)$ of $\phi$, or translate the encoding $\gamma^R(\phi)$ of $\phi$ along the encoding $\gamma^R(\sigma)$ of $\sigma$. To simplify notation, we write $\gamma(\Sigma)$ for $\gamma^S(\Sigma)$, and $\gamma(\phi)$ for $\gamma^R(\phi)$ (for appropriate choice of $\Sigma$.)

**Definition 2.6** A logic morphism $\gamma: L \rightarrow L'$ is a representation iff $\gamma^S$ is an embedding and each $\gamma^R$ is conservative. A representation is surjective iff each $\gamma^S$ is surjective as a function on the underlying sets. A logic $L$ is representable in a logic $L'$ iff there is a representation $p: L \rightarrow L'$.

Let $p$ range over representations. It is easy to see that identities are representations and that the composition of two representations is again a representation. The requirement that $p^S$ be an embedding implies that the category of signatures of the source logic is faithfully encoded in the target logic, and the requirement of conservativity implies that each consequence relation of the source is fully and faithfully encoded in the target. Thus if $p: L \rightarrow L'$, then $\Phi \vdash^p \phi$ iff $\Phi \vdash^L p^S \phi$. Note that surjectivity of $p$ does not entail that $p^S$ be full, only that $p^R$ be onto. We will give an example of a representation in Section 5 below.

## 3 Theory Presentations

In this section we define structured theory presentations for an arbitrary logical system. The presentation language that we choose is adapted from [10].

---

1We use $\circ$ to denote composition of functors, vertical composition of natural transformations, and composition of a functor with a natural transformation.

---

**Definition 3.1** Let $L$ be a logical system. An $L$-theory with signature $\Sigma$ is a set $T \subseteq [L]_\Sigma$ of sentences closed under $\vdash^L$.

**Definition 3.2** A structured theory presentation in $L$ ($L$-presentation) is an expression generated by the following grammar:

$$ P :: = \langle \Sigma, \Phi \rangle $$

$$ \mid P_1 \cup P_2 $$

$$ \mid \text{translate} P \text{ along } \sigma $$

$$ \mid \text{derive} P \text{ via } \sigma $$

(Here $\Sigma$ is a $\log$-signature, $\sigma$ is a $\log$-morphism and $\Phi$ is a set of $L$-sentences.) Structured presentations of the form $(\Sigma, \Phi)$ are called basic presentations. A structured presentation is finite if all the basic presentations it contains involve only finite sets of sentences.

In the above grammar we do not specify how signatures, signature morphisms, or sets of sentences are presented. For logics with finite signatures, it is unproblematic to define a presentation language for signatures and signature morphisms (e.g., [22]). In practice infinite presentations are given using some form of schematization. For the sake of simplicity we do not make this explicit here.

**Definition 3.3** The signature $\log(P)$ of a $L$-presentation $P$ is defined by induction on the structure of $P$ as follows: $\log(P) = \Sigma$ iff

- $P = \langle \Sigma, \Phi \rangle$, or
- $P = P_1 \cup P_2$ and $\log(P_1) = \log(P_2) = \Sigma$, or
- $P = \text{translate} P_1 \text{ along } \sigma, \sigma : \Sigma_1 \rightarrow \Sigma$, and $\log(P_1) = \Sigma_1$, or
- $P = \text{derive} P_1 \text{ via } \sigma, \sigma : \Sigma \rightarrow \Sigma_1$, and $\log(P_1) = \Sigma_1$.

$P$ is well-formed iff $\log(P)$ is defined.

**Definition 3.4** Let $P$ be a well-formed $L$-presentation. The theory determined by $P$ is defined as follows:

$$ \Th(L) = \Phi $$

$$ \Th(P_1 \cup P_2) = \Th(P_1) \cup \Th(P_2) $$

$$ \Th(\text{translate } P_1 \text{ along } \sigma) = \Th(\text{derive } P_1 \text{ via } \sigma) = \Th(P_1) $$
Proposition 3.5 \( \text{Th}^\ell(P) \) is an \( \mathcal{L} \)-theory with signature \( \text{Sg}^\ell(P) \).

The language of structured presentations is chosen to allow large theories to be build in a flexible and modular fashion. The primitives of the presentation language suffice for the definability of the theory-structuring operations of CLEAR [4], and may be used as the basis for an ML-like language for theory structuring [17,16]. Although space limitations prevent us from giving many details, we sketch a few examples of the use of structured presentations. See [3,19] for further examples. The translate and union operations may be used to combine theories. Let \( \text{Group} \) be the \( \mathcal{L} \)-presentation of the theory of groups with sort \( o \), operation \( * \), and inverse \( (\cdot)^{-1} \). Let \( \text{Abelian} \) be the \( \mathcal{L} \)-presentation of the theory of an abelian operator \(+\) on sort \( o \). Then

\[
\text{AbelianGroup} = \text{Group} \cup \text{translate Abelian} \text{ along }\iota
\]
is an \( \mathcal{L} \)-presentation of the theory of abelian groups, where \( \sigma \) is the \( \mathcal{L} \)-signature morphism that renames \(+\) to \( * \). The derive operation may be used for information hiding. The \( \mathcal{L} \)-theory of abelian monoids is presented by derive \( \text{AbelianGroup} \text{ via } \iota \) where \( \iota \) is the inclusion of the \( \mathcal{L} \)-signature of monoids into the \( \mathcal{L} \)-signature of groups.

CLEAR-style instantiation of parameterized theories [4] may be defined here using pushouts in the category of presentations.

Definition 3.6 A \( \mathcal{L} \)-presentation morphism \( \sigma : P \rightarrow P' \) is a \( \text{Sg}^\ell \)-morphism \( \sigma : \text{Sg}^\ell(P) \rightarrow \text{Sg}^\ell(P') \) such that \( \sigma(\text{Th}^\ell(P)) \subseteq \text{Th}^\ell(P') \). Identities, composition, and inclusions are inherited from \( \text{Sg}^\ell \). \( \text{ThPres}^\ell \) is the category of \( \mathcal{L} \)-presentations and morphisms between them.

A presentation \( P \) is "parametric" in a presentation \( R \) if there is a \( \text{ThPres}^\ell \)-inclusion \( \iota : R \hookrightarrow P \). The idea is that \( R \) is a "requirement" specification for the theory \( P \) which may be regarded as taking any theory "matching" \( R \) as a parameter. The parametric presentation \( P \) may be instantiated by any presentation \( A \) provided that there is a "fitting morphism" \( \sigma : R \rightarrow A \) specifying how \( A \) is to be regarded as satisfying the requirements of \( R \). The instance of \( P \) by \( A \) via \( \sigma \), written \( P(A[\sigma]) \), is obtained by taking the pushout of \( \sigma \) and \( \iota \) in \( \text{ThPres}^\ell \). The important point is that \( P(A[\sigma]) \) is definable in the language of structured presentations:

Proposition 3.7 If \( \text{Sg}^\ell \) has pushouts along inclusions, then so does \( \text{ThPres}^\ell \).

For example, we may define an \( \mathcal{L} \)-presentation \( \text{Seq} \) of the equational theory of sequences of objects of a sort \( \text{obj} \). If \( \text{Nat} \) is an \( \mathcal{L} \)-presentation of the theory of natural numbers, (with sort \( \text{nat} \)), then \( \text{Seq}(\text{Nat}[\sigma]) \) is a presentation of the equational theory of sequences of natural numbers, where \( \sigma \) is the \( \mathcal{L} \)-signature morphism sending \( \text{obj} \) to \( \text{nat} \).

A variety of other constructions are definable in \( \text{ThPres}^\ell \). For example, \( \text{ThPres}^\ell \) has coproducts whenever \( \text{Sg}^\ell \) does, and the theory of the coproduct is the disjoint union of the theories of the components. Colimits of more complex diagrams in \( \text{ThPres}^\ell \) may be used to express sharing; such colimits exist if they exist in \( \text{Sg}^\ell \). In particular, diagrams in \( \text{Sg}^\ell \) consisting only of inclusions arise in a natural way from the hierarchical construction of theories by extension. In many interesting cases all such diagrams have colimits, and we may therefore use colimits as the basis for a CLEAR-like or ML-like syntax for managing sharing [17,16].

4 Proof Search in Structured Presentations

Definition 4.1 The relation \( P \models^\ell \phi \), where \( \phi \) is a \( \text{Sg}^\ell(P) \)-sentence, holds iff \( \phi \in \text{Th}^\ell(P) \). This relation is extended to finite sets of sentences by defining \( P \models^\ell \Delta \) iff \( P \models^\ell \phi \) for each \( \phi \in \Delta \).

The following characterization of \( \models^\ell \) forms the basis for a search space based on structured presentations.

Proposition 4.2 Let \( P \) be a well-formed \( \mathcal{L} \)-presentation with \( \text{Sg}^\ell(P) = \Sigma \), and let \( \phi \in \{\mathcal{L}\}^\Sigma \). The relation \( P \models^\ell \phi \) may be characterized as follows:

1. \( \{\Sigma, \Phi\} \models^\ell \phi \) iff there exists \( \Delta \subseteq \Phi \) such that \( \Delta \vdash^\ell \phi \).
2. \( P_1 \cup P_2 \models^\ell \phi \) iff there exists \( \Delta_1 \subseteq \{\mathcal{L}\}^\Sigma \) and \( \Delta_2 \subseteq \{\mathcal{L}\}^\Sigma \) such that \( P_1 \models^\ell \Delta_1 \), \( P_2 \models^\ell \Delta_2 \), and \( \Delta_1, \Delta_2 \vdash^\ell \phi \).
3. translate \( P_1 \) along \( \sigma \models^\ell \phi \) (where \( \sigma : \Sigma_1 \rightarrow \Sigma \)) iff there exists \( \Delta_1 \subseteq \{\mathcal{L}\}^\Sigma_1 \) such that \( P_1 \models^\ell \Delta_1 \) and \( \mathcal{L}(\sigma)[\Delta_1] \vdash^\ell \phi \).
4. derive \( P_1 \) via \( \sigma \models^\ell \phi \) iff \( P_1 \models^\ell \mathcal{L}(\sigma)[\phi] \).
To illustrate how a search procedure may take advantage of the structure of a presentation, consider a logical system \( \mathcal{L} \) with inclusions. Let \( P_1 \) be an \( \mathcal{L} \)-presentation with signature \( \Sigma_1 \), and let \( \xi : \Sigma_1 \rightarrow \Sigma \) be an inclusion. If \( \phi \in [\mathcal{L} \Sigma_1] \), then a useful heuristic for testing \( \xi(P_1) \) alone \( \xi \vdash \mathcal{L} \phi \) is to take \( \Delta_1 \) in the above proposition to be \( \{ \phi \} \), and to test \( P_1 \vdash_{\mathcal{L}} \phi \). This is sufficient (but not necessary), for since \( \mathcal{L} \) preserves inclusions, \( \xi \Delta_1 \setminus \xi \Delta_2 \) is trivial. For further examples of how a search procedure may take advantage of the structure of presentations, see [19].

If \( \rho \) is a representation of \( \mathcal{L} \) in \( \mathcal{L}' \), then we may use \( \mathcal{L}' \) as an "inference engine" for \( \mathcal{L} \) by replacing all uses of \( \Delta \vdash_{\mathcal{L}} \phi \) by \( \rho(\Delta) \vdash_{\mathcal{L}'} \rho(\phi) \). In particular, a search procedure based on the above characterization need not make any use of \( \mathcal{L} \) for elementary inference. The search process is still, however, driven by an \( \mathcal{L} \)-presentation \( P \), and so involves the sentences, signatures and translations induced by signature morphisms of \( \mathcal{L} \). But if our goal is to reduce all inferential activity in \( \mathcal{L} \) to inferential activity in \( \mathcal{L}' \), then we would like to "lift" \( P \) to an \( \mathcal{L}' \)-presentation, and perform structured search in \( \mathcal{L}' \) guided by the lifted presentation. To make this precise, we first define a natural lifting of presentations.

**Definition 4.3** Suppose that \( \rho : \mathcal{L} \rightarrow \mathcal{L}' \) is a representation, and let \( P \) be a \( \mathcal{L} \)-presentation with signature \( \Sigma \). The representation of \( P \) in \( \mathcal{L}' \) wrt \( \rho \) is given by the following function defined by induction on the structure of \( P \):

\[
\tilde{\rho}(\{ \xi, \Phi \}) = (\rho \circ \xi)_{\mathcal{L}'}(\Phi)
\]

\[
\tilde{\rho}(P_1 \cup P_2) = \tilde{\rho}(P_1) \cup \tilde{\rho}(P_2)
\]

\[
\tilde{\rho}(\text{translate } P_1 \text{ along } \sigma) = \text{translate } \tilde{\rho}(P_1) \text{ along } \rho \circ \sigma
\]

\[
\tilde{\rho}(\text{derive } P_1 \text{ via } \sigma) = \text{derive } \tilde{\rho}(P_1) \text{ via } \rho \circ \sigma
\]

**Proposition 4.4** If \( P \) is an \( \mathcal{L} \)-presentation, then \( \tilde{\rho}(P) \) is a \( \mathcal{L}' \)-presentation with \( \text{Sg}_{\mathcal{L}'}(\tilde{\rho}(P)) = \rho \circ \text{Sg}_{\mathcal{L}'}(P) \).

The above discussion may be summarized by the following conjecture.

**Conjecture 4.5** \( P \vdash_{\mathcal{L}} \phi \) iff \( \tilde{\rho}(P) \vdash_{\mathcal{L}'} \rho(\phi) \).

In other words, to test \( P \vdash_{\mathcal{L}} \phi \), encode \( P \) and \( \phi \) in \( \mathcal{L}' \), and test \( \tilde{\rho}(P) \vdash_{\mathcal{L}'} \rho(\phi) \).

Although the implication from left to right does indeed hold (the naive lifting of presentations is complete), the converse fails (it is not sound.) The following crucial property is lost:

\[
\text{Th}^{\mathcal{L}'}(\tilde{\rho}(P)) \subseteq \rho(\text{Th}^{\mathcal{L}}(P))
\]

The discrepancy can be traced to derive: if \( P = \text{derive } P_1 \text{ via } \sigma \), then

\[
\text{Th}^{\mathcal{L}'}(\tilde{\rho}(P)) = \text{Th}^{\mathcal{L}'}(\text{derive } \tilde{\rho}(P_1) \text{ via } \rho(\sigma)) = \rho(\sigma)^{-1}(\text{Th}^{\mathcal{L}'}(\tilde{\rho}(P_1)))
\]

can be strictly larger than

\[
\rho(\text{Th}^{\mathcal{L}}(P)) = \rho(\sigma)^{-1}(\text{Th}^{\mathcal{L}}(P_1))
\]

even if \( \text{Th}^{\mathcal{L}'}(\tilde{\rho}(P_1)) = \rho(\text{Th}^{\mathcal{L}}(P_1)) \). The "excess" must necessarily be a \( \mathcal{L}' \)-sentence \( \psi' \) which does not lie in the image of \( \rho \) (i.e., does not represent any \( \mathcal{L} \)-sentence). This by itself does not refute the conjecture, but in conjunction with another presentation this discrepancy may be exploited to yield a counterexample to the conjecture. (We omit details due to space limitations.) We stress that a counterexample of this kind is by no means artificial. Many LF encodings employ "auxiliary" judgements (e.g., the judgement \( \text{valid}(\cdot) \) in the encoding of S4 [2]) which may be used to refute the conjecture.

It makes sense, then, to ask in what sense we can lift structured presentations from \( \mathcal{L} \) to \( \mathcal{L}' \). The answer is given by considering an alternative definition of the theory of an \( \mathcal{L} \)-presentation that is conditioned by the representation \( \rho \).

**Definition 4.6** Let \( P \) be a \( \mathcal{L} \)-presentation with signature \( \Sigma \). The \( \mathcal{L}' \)-theory of \( P \) wrt \( \rho \) is defined as follows:

\[
\text{Th}_{\mathcal{L}'}(\{ \xi, \Phi \}) = (\rho(\xi), \rho(\Phi))
\]

\[
\text{Th}_{\mathcal{L}'}(P_1 \cup P_2) = \text{Th}_{\mathcal{L}'}(P_1) \cup \text{Th}_{\mathcal{L}'}(P_2)
\]

\[
\text{Th}_{\mathcal{L}'}(\text{translate } P_1 \text{ along } \sigma) = \rho(\sigma)(\text{Th}_{\mathcal{L}'}(P_1))
\]

\[
\text{Th}_{\mathcal{L}'}(\text{derive } P_1 \text{ via } \sigma) = \rho(\sigma)^{-1}(\text{Th}_{\mathcal{L}'}(P_1))
\]

Note that we are defining the \( \mathcal{L}' \)-theory of an \( \mathcal{L} \)-presentation, conditioned by the representation \( \rho \) of \( \mathcal{L} \) in \( \mathcal{L}' \). This is because in the case of derive, the restriction to the range of \( \rho \) makes reference to the \( \mathcal{L} \)-signature \( \Sigma \) of the \( \mathcal{L} \)-presentation \( P \). Although this restriction ensures that only \( \mathcal{L} \)-sentence images are taken from \( P \), the closure of the result under \( \vdash_{\mathcal{L}'} \) admits non-sentence images into the result. In effect, in the case of derive, only \( \mathcal{L} \)-sentence images are...
admitted as intermediate lemmas, whereas arbitrary $L'$-sentences are admitted as consequences of these lemmas. This will be reflected in the search space associated with this definition.

**Theorem 4.7** If $P$ is a well-formed $L$-presentation, then $\text{Th}_\rho^L(P)$ is an $L'$-theory with signature $\rho(\text{Sg}^L(P))$ such that $\text{Th}_\rho^L(P) = \rho(\text{Th}_\rho^L(P))$.

This may be expressed in terms of search by introducing the following relation.

**Definition 4.8** The relation $P \models L' \phi'$, where $P$ is a well-formed $L$-presentation with signature $\Sigma$ and $\phi' \in \xi L'_1(\Sigma)$, holds iff $\phi' \in \text{Th}_\rho^L(P)$. This relation is extended to finite sets of $L'$-sentences as before.

Note that this relation is between $L$-presentations $P$ and $L'$-sentences $\phi'$ with signature $\rho(\text{Sg}^L(P))$. Using this relation, Theorem 4.7 may be restated as follows:

**Corollary 4.9** $P \models L' \phi$ iff $P \models L' \rho(\phi)$.

The following proposition provides a useful characterization of the relation $P \models L' \phi'$:

**Proposition 4.10** Let $P$ be an $L$-presentation with $\text{Sg}^L(P) = \Sigma$, and let $\phi' \in \xi L'_1(\Sigma)$. The relation $P \models L' \phi'$ may be characterized as follows:

1. $(\Sigma, \Phi) \models L' \phi'$ iff there exists $\Delta \subseteq \Phi$ such that $\rho(\Delta) \models L' \phi'$.
2. $P_1 \cup P_2 \models L' \phi'$ iff there exists $\Delta'_1, \Delta'_2 \subseteq \xi L'_1(\Sigma)$ such that $P_1 \models L' \Delta'_1$, $P_2 \models L' \Delta'_2$, and $\Delta'_1 \cup \Delta'_2 \models L' \phi'$.
3. Translate $P_1$ along $\sigma$ to $L' \phi'$ (where $\sigma : \Sigma \rightarrow \Sigma$) iff there exists $\Delta'_1 \subseteq \xi L'_1(\Sigma)$ such that $P_1 \models L' \Delta'_1$ and $\rho(\sigma)(\Delta'_1) \models L' \phi'$.
4. Derive $P_1$ via $\sigma$ to $L' \phi'$ iff there exists $\Delta' \subseteq \rho(\xi L'_1(\Sigma))$ such that $P_1 \models L' \rho(\sigma)(\Delta')$ and $\Delta' \models L' \phi'$.

This proposition provides the basis for a search procedure for $L'$ sentences relative to an $L$-presentation. As we remarked above, we would like to achieve a complete reduction to $L'$ by working with the representation $\tilde{\rho}(P)$ of $P$. The conditions under which we can achieve this may be derived by comparing the search procedure determined by Proposition 4.10 for $P \models L' \rho(\phi)$ with that determined by Proposition 4.2 for the case of $\rho(P) \models L' \rho(\phi)$.

First, if we restrict attention to $L'$-sentences $\phi'$ in the image of $\rho$ (i.e., such that there exists an $L'$-sentence $\phi$ with $\phi' = \rho(\phi)$), then case (4) of Proposition 4.10 may be simplified to

\[ \text{derive } P_1 \text{ via } \sigma \models L' \phi' \text{ iff } P_1 \models L' \rho(\sigma)(\phi'), \]

since

\[ \rho(\sigma)(\phi') = \rho(\sigma)(\rho(\phi)) = \rho(\sigma(\phi)) \]

(the last step by naturality), and so we can take $\Delta' = \{ \rho(\phi) \}$, for which the condition $\Delta' \models L' \rho(\phi)$ is trivial. However, as we shall see below, in practical situations it is necessary to admit the use of $L'$-sentences lying outside of the range of $\rho$ in the process of search for sentences lying within the range of $\rho$. It is therefore important to admit arbitrary $L'$ sentences as goals of the "lifted" search procedure.

Second, although the search procedure described by Proposition 4.10 is guided by an $L$-presentation $P$, it does not make direct use of any of the components of $P$, but rather only of their representations. For example, in the case of translate, the procedure applies $\rho(\sigma)$, not $\sigma$ (i.e., $L'(\rho(\sigma))$, not $L'(\sigma)$). In a sense the search procedure forms $\tilde{\rho}(P)$ "on the fly," taking the representations of each component of $P$ in order to carry out the search. The essential difference between an $L'$ search guided by $\tilde{\rho}(P)$ and the above $P$-guided search procedure lies in the restriction on $\Delta'$ in the case of derive. To enforce this restriction, the search procedure must be able to decide, given $\rho$ and $\phi' \in \xi L'_1(\Sigma)$, whether $\rho(\phi')$ for some $\phi \in \xi L_1$. Such a test requires only the signature $\Sigma$ of $\rho(\phi)$ and the representation $\rho$. But since $\rho$ is a representation, the component $\rho(\Sigma)$ is an embedding, and hence $\Sigma$ is determined by $\rho(\Sigma)$. Therefore no $\Sigma$-entities are needed; it is enough to have the image $\rho(\Sigma)$ of $\Sigma$. We may therefore use $\tilde{\rho}(P)$ to guide the search, provided that $\rho$ is a representation and we can test membership in the range of $\rho$. To assess the practical implications of this requirement, we turn to the representation of logics in $L_P$.  

232
5 Logical Systems and LF

In order to discuss representations of logical systems in LF, we first define the logical system associated with the LF type theory. The basic form of assertion in this logic is that a closed type is inhabited. The restriction to closed types is a simplification that suffices for the purposes of this paper, but would have to be relaxed in practice. (See Section 7 for further discussion.)

Definition 5.1 An LF signature morphism \( \sigma : \Sigma_1 \to \Sigma_2 \) is a function \( \sigma \) mapping constants to closed terms such that if \( c : A (c : K) \) occurs in \( \Sigma_1 \), then \( \vdash_{\Sigma_2} \sigma(c) : \sigma^1 A \) (\( \vdash_{\Sigma_2} \sigma(c) : \sigma^1 K \)). (The function \( \sigma^1 \) is the natural extension of \( \sigma \) to LF terms.) The identity morphism on \( \Sigma \) is the identity function on \( \text{dom}(\Sigma) \), and composition is defined by \( \sigma_1 ; \sigma_2 = \sigma_1 ; \sigma_2^1 \). Inclusions are the inclusion functions on the underlying sets of constants. \( \text{Sig}^{LF} \) is the category of LF signatures and LF signature morphisms.

Note that if \( \iota \) is an inclusion, then \( \iota^1 \) is the inclusion, and hence is usually omitted.

Definition 5.2 Let \( \Sigma \) be an LF signature. \( \mathcal{L}F(\Sigma) \) is the pair \( (\text{Types}_\Sigma, \vdash_{\Sigma}^{\mathcal{L}F}) \) where \( \text{Types}_\Sigma = \{ A \mid \vdash_{\Sigma} A : \text{Type} \} \) and

\[
A_1, \ldots, A_n \vdash_{\Sigma}^{\mathcal{L}F} A \quad \text{iff} \quad \forall x_1 : A_1, \ldots, x_n : A_n \vdash_{\Sigma} M : A
\]

for some \( M \) and any pairwise distinct variables \( x_1, \ldots, x_n \).

This consequence relation has a straightforward Gentzen-style axiomatization similar to that used in NuPRL [6] that may be used as the basis for interactive proof search.

This construction extends to a functor \( \mathcal{L}F : \text{Sig}^{LF} \to \text{CR} \) defined by taking \( \mathcal{L}F(\sigma) \), for \( \sigma : \Sigma_1 \to \Sigma_2 \), to be \( \sigma^1 \vdash_{\Sigma_1} \), the restriction of \( \sigma^1 \) to closed \( \Sigma_1 \)-types. It is easy to verify that \( \mathcal{L}F \) is a logical system with inclusions.

For the purposes of encoding a logical system \( \mathcal{L} \), we are interested in "specializations" of \( \mathcal{L}F \) obtained by fixing a "base" signature \( \Sigma_2 \) specifying the syntax, assertions, and rules of \( \mathcal{L} \) [12]. The signatures of \( \mathcal{L} \) are then represented as extensions to \( \Sigma_2 \), and signature morphisms are represented as LF signature morphisms on these extensions leaving \( \Sigma_2 \) fixed. Inferential activity for \( \mathcal{L} \) is then reduced to inferential activity in the specialization of \( \mathcal{L}F \) to \( \Sigma_2 \). To make this precise, some additional machinery is needed.

Definition 5.3 Let \( \Sigma \) be an LF signature. The category of extensions of \( \Sigma \), written \( \text{Sig}^{LF}_\Sigma \), is the full subcategory of the slice category \( \Sigma \downarrow \text{Sig}^{LF} \) determined by the inclusions \( \iota : \Sigma \to \Sigma' \). A morphism from \( \iota_1 : \Sigma \to \Sigma_1 \) to \( \iota_2 : \Sigma \to \Sigma_2 \) in \( \text{Sig}^{LF}_\Sigma \) is a signature morphism \( \sigma : \Sigma_1 \to \Sigma_2 \) in \( \text{Sig}^{LF} \) such that \( \iota_1 ; \sigma = \iota_2 \). The identities and composition are inherited from \( \text{Sig}^{LF} \).

Every LF signature induces a logical system based on that signature as follows:

Definition 5.4 Let \( \Sigma \) be an LF signature. The logical system \( \mathcal{L}F_\Sigma : \text{Sig}^{LF}_\Sigma \to \text{CR} \) is defined on objects by \( \mathcal{L}F_\Sigma(\iota : \Sigma \to \Sigma') = \mathcal{L}F(\Sigma') \) and on morphisms \( \sigma : \Sigma' \to \Sigma'' \) (in the category of extensions of \( \Sigma \)) by \( \mathcal{L}F_\Sigma(\iota) = \mathcal{L}F(\sigma) \).

It is easy to verify that \( \mathcal{L}F_\Sigma \) is a logical system with inclusions.

An encoding of a logical system \( \mathcal{L} \) in LF consists not only of an LF signature \( \Sigma_2 \), but also an "internal type family" distinguishing the basic judgements of \( \mathcal{L} \) in the encoding. For example, in the encoding of first-order logic given in [12], the constant true of kind \( \sigma \to \text{Type} \) represents the basic judgement form of first-order logic. The significance of true for the encoding becomes apparent in the statement of the adequacy theorem: terms of type \( \text{true}(\phi) \) in a context with variables \( \phi \) of type \( \text{true}(\phi) \) represent proofs of \( \phi \) from the \( \phi \)'s. This methodology is formalized in our setting as follows.

Definition 5.5 An internal type family of \( \Sigma \) is a term \( F \) such that \( \vdash_{\Sigma} F : K \) for some kind \( K \). The range \( \text{Rng}_{\Sigma}(F) \) of an internal type family \( F \) wrt \( \Sigma \) is defined to be the set

\[
\{ \text{lnf}(F M_1 \ldots M_k) \mid \vdash_{\Sigma} F M_1 \ldots M_k : \text{Type} \}
\]

where \( \text{lnf}(M) \) is the long \( \beta \eta \)-normal form of \( M \).

The range of a set of internal type families is the union of the ranges of its elements.
Definition 5.6. A logic presentation is a pair $(\Sigma, J)$ where $\Sigma$ is an LF signature and $J$ is a finite set of internal type families of $C$.

Definition 5.7. Let $(\Sigma, J)$ be a logic presentation. The logical system presented by $(\Sigma, J)$, $\mathcal{P}(\Sigma, J)$, is the restriction of $\mathcal{LF}_\Sigma$ to the range of $J$. Specifically, $\mathcal{P}(\Sigma, J) : \text{Sig}_0^\Sigma \to \text{CR}$ is defined on objects by

$$\mathcal{P}(\Sigma, J)(\iota : \Sigma \hookrightarrow \Sigma') = \mathcal{LF}(\Sigma') \upharpoonright \text{Rng}_{\Sigma'}(J)$$

and on morphisms $\sigma : \Sigma' \to \Sigma''$ in the slice category by

$$\mathcal{P}(\Sigma, J)(\sigma) = \mathcal{LF}(\sigma) \upharpoonright \text{Rng}_{\Sigma''}(J).$$

Definition 5.8. A logical system is uniformly encodable (in LF) iff there exists a logic presentation $(\Sigma_L, J_L)$ and a surjective representation $\rho_L : \mathcal{L} \to \mathcal{P}(\Sigma_L, J_L)$. The triple $(\Sigma_L, J_L, \rho_L)$ is called a uniform encoding of $\mathcal{L}$.

The word “uniform” reflects the fact that we require a “natural” encoding of the entire family of consequence relations of $\mathcal{L}$ in LF, rather than a signature-by-signature encoding as is suggested by the account in [12]. The requirement of surjectivity ensures that $J$ accurately describes the images of $\mathcal{L}$-sentences in LF. For example, in the encoding of first-order logic in [12], all closed long normal forms of the shape $\text{true}(M)$ represent first-order sentences.

Since the representation part of a uniform encoding of a logic $\mathcal{L}$ is required to be surjective, it might be thought that we may use the naïve lifting of $\mathcal{L}$-presentations to LF. But this is not the case, for in practice we work not in $\mathcal{P}(\Sigma_L, J_L)$, but in $\mathcal{LF}(\Sigma_L)$, which is to say that we cannot restrict attention to sentences in the range of $J_L$. For example, in the encoding of $\mathcal{S}_4$ [2], sentences are represented by terms of the form $\text{true}(M)$. But to prove, say, $\text{true}(\text{false}(M))$, we must, in certain cases, prove $\text{valid}(M)$. But this type lies outside of the image of $\rho$ (and cannot be soundly included in it.)

Now since $\mathcal{P}(\Sigma, J)$ is defined to be the restriction of $\mathcal{LF}_\Sigma$ to the range of $J$, there is an obvious inclusion of $\mathcal{P}(\Sigma, J)$ into $\mathcal{LF}_\Sigma$ which is typically not surjective. If $(\Sigma_L, J_L, \rho_L)$ is a uniform encoding of $\mathcal{L}$, then to test whether a type $A$ of $\mathcal{LF}_\Sigma(\iota : \Sigma \hookrightarrow \Sigma')$ is in the image of $\rho_L$, we must test whether or not $A \in \text{Rng}_{\Sigma'}(J_L)$. This amounts to matching in the LF type theory: $A \in \text{Rng}_{\Sigma'}(J_L)$ iff there exists $J \in J$ and $M_1, \ldots, M_k$ (where $k$ is determined by $J$) such that $A$ is convertible to $J(M_1, \ldots, M_k)$. This test may be implemented using the unification algorithms developed by Pym [15] or Elliot [8]. In practice, $J$ is often a single constant, in which case this test is trivial; it is an open problem to determine whether the matching problem is, in general, decidable.

As an example of a uniform encoding, we consider the logical system $\mathcal{L}_Q$ defined in Section 2, but restricted to the single-sorted case for the sake of simplicity. Thus algebraic signatures are families of the form $(\Omega_n)_{n \geq 0}$ where the type of individuals is left implicit and for $n \geq 0$, $\Omega_n$ is the set of $n$-ary function symbols. Call this logical system $\mathcal{L}_Q$. We begin with the even simpler case of single-sorted ground equational logic, $\mathcal{L}_Q^*$. Let $\mathcal{L}_Q^*$ be the LF signature:

$$\begin{align*}
  \iota & : \text{Type} \\
  o & : \iota \to \iota \\
  \text{true} & : o \to o \\
  \text{eq} & : \iota \to \iota \to o \\
  \text{refl} & : \Pi \iota. \text{true}([x = x]) \\
  \text{sym} & : \Pi \iota. \Pi \iota. \Pi \iota. \text{true}([x = y]) \to \text{true}([y = x]) \\
  \text{trans} & : \Pi \iota. \Pi \iota. \Pi \iota. \text{true}([x = y]) \to \text{true}([y = z]) \to \text{true}([x = z]) \\
  \text{eq} & : \Pi \iota. \Pi \iota. \Pi \iota. \text{true}([x = y]) \to \text{true}([\phi(f(x))] \\
  & \to \text{true}([\phi(f(y)])
\end{align*}$$

and let $J \mathcal{L}_Q^* = \{ \text{true} \}$. (A simpler presentation not involving the type $o$ may be given, but we use this formulation for the sake of the generalization to $\mathcal{L}_Q$ to follow.)

A uniform encoding of $\mathcal{L}_Q^*$ is the triple $(\Sigma \mathcal{L}_Q^*, \mathcal{L}_Q^*, \rho)$ where $\rho : \mathcal{L}_Q^* \to \mathcal{P}(\Sigma \mathcal{L}_Q^*, \mathcal{L}_Q^*)$ is the surjective representation determined as follows. Each single-sorted algebraic signature $\Omega$ is represented as the LF signature $\rho(\Omega)$ obtained by extending $\Sigma \mathcal{L}_Q^*$ with a declaration of the form $f : \iota \to \iota \to \iota \to \iota$ (with $n + 1$ occurrences of $\iota$ for each $n$-ary function symbol $f$ in $\Omega$ (we assume that the constants of the algebraic signature do not conflict with the constants of $\Sigma \mathcal{L}_Q^*$ given above.) This extends to a functor in the obvious way. There is an obvious bijection between ground $\Omega$-equations and closed LF types of the form $\text{true}([eq t t'])$ in $\rho(\Omega)$. It is easy to see that this bijection commutes with renaming of symbols, and defines a conservative CR morphism between $\mathcal{L}_Q^*(\Omega)$ and $\mathcal{LF}(\rho(\Omega)) \upharpoonright \text{Rng}_{\Omega}(\text{true})$. 

234
Since sentences of a logic are represented by closed types, it is necessary to introduce an explicit quantifier to bind the free variables and to simulate the implicit universal quantification of sentences in $\mathcal{L}Q$. Let $\Sigma \mathcal{L}Q_1$ be the signature

\[
\forall : (A \to o) \to o \\
\forall I : \Pi \phi : o \to o. (\Pi x : A. \text{true}(\phi x)) \to \text{true}(\forall \phi) \\
\forall E : \Pi \phi : o. \Pi x : A. \text{true}(\forall \phi) \to \text{true}(\phi x)
\]

and let $\mathcal{L}Q_1 = \{\text{true}\}$. We can define a surjective representation $\rho : \mathcal{L}Q_1 \to P(\Sigma \mathcal{L}Q_1, \mathcal{L}Q_1)$ similarly as in Example 7?. This yields a uniform encoding $(\Sigma \mathcal{L}Q_1, \mathcal{L}Q_1, \rho)$ of $\mathcal{L}Q_1$.

6 Putting Together Logics

In this section we consider the adaptation of the idea of presenting theories in a structured way to logic presentations. As a first step in this direction we investigate the use of pushouts to give an account of parameterization and instantiation of logic presentations. We have in mind such examples as: the parameterization of Peano arithmetic by the underlying predicate calculus, with instantiations like classical Peano arithmetic and Heyting arithmetic; the parameterization of Hoare logic by the logic of assertions; the parameterization of the calculus of synchronization trees by the synchronization algebra [21].

**Proposition 6.1** $\text{Sig}^{\mathcal{L}T}$ has pushouts along inclusions.

**Definition 6.2** A logic presentation morphism $\sigma : (\Sigma, J) \to (\Sigma', J')$ is a signature morphism $\sigma : \Sigma \to \Sigma'$ in $\text{Sig}^{\mathcal{L}T}$ such that for every $F \in J$ with

\[\downarrow \Sigma F : \Pi x_1 : A_1. \ldots x_k : A_k. \text{Type},\]

there exists $F' \in J'$ such that

\[\sigma^\dagger F = \rho_{\sigma(\text{Type})} \lambda x_1 : A_1. \ldots x_k : A_k. F'(M_1, \ldots, M_n)\]

for some $M_1, \ldots, M_n$. Identity and composition are inherited from $\text{Sig}^{\mathcal{L}T}$. LogPres is the category of logic presentations and logic presentation morphisms.

**Proposition 6.3** The assignment $(\Sigma, J) \mapsto P(\Sigma, J)$ extends to a functor $P : \text{LogPres} \to \text{Log}$.

We propose to use colimits in the category of logic presentations to build logics in the same way as colimits were used in Section 3 to build theories. Although the category of logic presentations is not finitely co-complete, it may be shown that a diagram in LogPres has a colimit iff its projection to $\text{Sig}^{\mathcal{L}T}$ has a colimit. The most pertinent case is that of pushouts along inclusions:

**Definition 6.4** A logic presentation morphism $\iota : (\Sigma, J) \hookrightarrow (\Sigma', J')$ is an inclusion if $\iota : \Sigma \hookrightarrow \Sigma'$ is an inclusion and $J \subseteq J'$.

**Proposition 6.5** LogPres has pushouts along inclusions.

A LogPres inclusion can be seen as a parameterized logic presentation where the pushout of this morphism with a “fitting” morphism amounts to instantiation, by analogy with parameterized structured theory presentations.

**Example 6.6**

\[
\begin{align*}
\Sigma \text{BASE}'' &= o : \text{Type} \\
\text{true} : o \to \text{Type} \\
\text{BASE}'' &= \{\text{true}\} \\
\Sigma \text{PROP} &= \Sigma \text{BASE}'' , \\
\rightarrow : o \to o \\
\wedge : o \to o \to o \\
\lor : o \to o \to o \\
\Rightarrow : o \to o \to o \\
\Pi \phi : o. \Pi \psi : o. (\text{true}(\phi) \to \text{true}(\psi)) \to \text{true}(\Rightarrow \phi \psi) \\
\text{PROP} &= \{\text{true}\} \\
\text{BASE}'' &= (\Sigma \text{BASE}'' , \text{BASE}'' ) \text{ presents a trivial logic containing only the type of formulae (o) and the judgement form true. PROP = (PROP, PROP) presents propositional logic; only one of the standard inference rules is given above. There is an obvious inclusion } \iota : \text{BASE}'' \hookrightarrow \text{PROP} , \text{ which may be seen as propositional logic parameterized by the type of atomic propositions.} \\
\text{Instantiation of this parameterized logic presentation to the presentation of single-sorted ground equational logic } (\Sigma \mathcal{L}Q_1 , \mathcal{L}Q_1 ) \text{ (see Example 5.9),}
\end{align*}
\]
via the inclusion of \( BASE'' \), yields a presentation \( PROP(GE_Q') \) of a propositional logic where atomic formulae are ground equations.

**Example 6.7**

\[
\begin{align*}
\Sigma BASE'' &= \iota : \text{Type} \\
o &= \text{Type} \\
\text{true} &: o \to \text{Type} \\
J BASE'' &= \{ \text{true} \} \\
\Sigma UNIV &= \Sigma BASE'', \\
\forall &:(i \to o) \to o \\
\forall I &:\forall \phi: i \to o. \ (\Pi z: i. \text{true}(\phi z)) \to \text{true}(\forall \phi) \\
\forall E &: \forall \phi: i \to o. \ (\Pi z: i. \text{true}(\phi z)) \to \text{true}(\forall \phi) \\
J UNIV &= \{ \text{true} \} \\
BASE'' = (J BASE'', BASE'', \Sigma BASE'', \Sigma UNIV) &\text{ presents a logic containing only the type of individuals (i) and formulae (o) and the judgement form true. The logic presentation } UNIV = (\Sigma UNIV, J UNIV) \text{ presents a logic of universal quantification. There is an obvious inclusion } i : BASE'' \rightarrow UNIV, \text{ which may be seen as a pure logic of universal quantification parameterized by the types of individuals and formulae.} \\
\text{We use pushouts in the category of logic presentations as a means of putting together logics. It is important to note that these constructions cannot be carried out in the category of logics. Roughly speaking, in Log the internal structure of sentences is not visible, and pushouts here are rather coarse. For example, consider two extensions to } GE_Q, \text{ one with conjunction, the other with disjunction. If we push out their presentations, we get (via } P\text{) a logical system with equations, conjunctions of disjunctions of equations, and so forth. On the other hand, if we push}
\end{align*}
\]

out in Log, we get only the union of the two logical systems, giving conjunctions and disjunctions of equations, but no other combinations.

### 7 Directions for Future Research

The definition of logical system, and especially the definition of uniform encoding, reflects the intention that sentences be "closed." The definition of logical system and uniform encoding could be generalized to admit "open" sentences, but it is important to realize that there are (at least) two different ways to construct consequence in this situation \([1]\). Under the "truth" interpretation, free variables behave essentially as constants, and hence could be handled within our notion of logical system (the situation is more complicated in free logics such as \( PX' \)[13]). Under the "validity" interpretation, free variables are implicitly universally quantified at each formula. Hilbert-type presentations of first-order logic usually take the validity interpretation, whereas natural deduction presentations take the truth interpretation. Preliminary investigation indicates that our notion of logical system may be extended to accommodate these generalizations.

The definition of basic theory presentation admits the possibility of an infinite set of axioms. In practice such sets are presented schematically since theories of interest are recursively presentable. The notion of logical system can be extended to treat axiom schemes explicitly, and the definition of uniform encoding can be correspondingly generalized to encode schemes using II-types. This extension becomes important in the case of certain truth-type logical systems lacking a universal quantifier, for there it is not possible to think of an axiom scheme as standing for all of its instances. It would be interesting to work out a treatment of schematization for both truth-type and validity-type logical systems.

The emphasis in this paper has been on provability, rather than on finding proofs. This is reflected in our decision to view logical systems as consequence relations, and in the concomitant definition of search in structured presentations. We have considered generalizing the notion of consequence relation to consequence category as a way of keeping track of proofs. It would be interesting to develop the theory of structured presentations in this setting.

The language of structured presentations may be generalized to admit translation and inverse image
along logic morphisms. This would allow for the combination of theories from several different logical systems, giving rise to an "inter-logic" search space similar to the "intra-logic" search space given by structured theory presentations. It would be interesting to develop these ideas further, and to consider their application to formal program development where there is some indication that such hybrid logics and inter-logic search will be of some use [18].

Acknowledgements: Thanks to Rod Burstall for his work on connections between LF and institutions and (from DS) for earlier collaboration on structured theories. This research has been partially supported by the U.K. Science and Engineering Research Council and Edinburgh University (RH, DS, AT), Carnegie-Mellon University (RH), the Polish Academy of Sciences and Linköping University (AT).


