A Proof-Theoretic Approach to the HOL/Nuprl Connection with Applications to Theory and Proof Translation*

Mark-Oliver Stehr[1] Pavel Naumov[2] José Meseguer[1]

[1] Computer Science Laboratory, SRI International,
Menlo Park, CA, 94025, USA, {stehr,meseguer}@csl.sri.com

[2] Department of Mathematical and Computer Sciences,
Pennsylvania State University, Middletown, PA, 17057, USA, naumov@psu.edu

Abstract

The connection between the HOL and Nuprl proof assistants earlier introduced by Howe is reexamined from a proof-theoretic point of view as a translation between the sentences of HOL and a classical variant of Nuprl followed by a theory interpretation stage. Using standard terminology from the theory of general logics we establish soundness as a proof-theoretic correctness result of the translation which complements Howe's semantics-based argument. Soundness is obtained in a constructive way by giving a Nuprl proof for each proof of a translated HOL theorem. As an application of this approach we have developed an executable formal specification in Maude of an HOL to Nuprl theory translator, as well as a proof translator on top of the Nuprl system that converts HOL proofs into their Nuprl counterparts.

1 Introduction

We investigate Doug Howe's connection [8, 14] between the classical logic of the HOL system [11] and a classical variant of the Nuprl type theory [7] from the viewpoint of general logics [16]. The main idea behind this connection is to rediscover the classical logic present in the boolean data type of an expressive type theory like Nuprl using a hybrid computational/set-theoretic semantics [13, 12]. This semantics justifies a classical variant of Nuprl (see [12] for details) which makes use of the axiom of the excluded middle. In the remainder of this paper we will simply refer to this variant as Nuprl. The benefit of this approach is that an extensional and an intensional logic can coexist in a single framework and that we can take advantange of both. The former is well suited to deal directly with external classical knowledge, and for the latter theorem-proving is well supported in the Nuprl system and, moreover, it allows extraction of programs from proofs.

*We gratefully acknowledge support for the work conducted at SRI by DARPA and NASA (Contract NAS2-98073), by Office of Naval Research (Contract N00014-96-C-0114), by NSF Grant (CCR-9633363), and by a DAAD grant in the scope of HSP-III. The second author was supported by DARPA grant F30602-98-2-0198 on the initial stage of this work that was done at Cornell University, Ithaca, NY.

The HOL/Nuprl connection as it has been implemented in [15] can be understood in the framework of general logics as a composition of two stages:

1. The first stage is a *translation* of an *axiomatic HOL theory* into an *axiomatic Nuprl theory*. The use of the term "axiomatic" emphasizes the fact that the theories are not necessarily only definitional extensions of the basic logic. The translation is, by its very nature, meta-logical, in the sense that, by relating two different logics, it is beyond the scope of each of them. It is therefore a critical stage whose correctness cannot be reduced to that of the two theorem provers involved and requires careful analysis.

2. The second stage is the *interpretation* of an axiomatic Nuprl theory inside Nuprl. In this way we can often obtain a *computationally meaningful* theory, which is closer to the spirit of Nuprl, that favors definitional extensions. As in Howe's extension of Nuprl, this interpretation stage can take place inside the Nuprl system in a formally rigorous way. However, since there are many possible choices for the interpretation, and certain proofs to be carried out, user interaction is required.

The present paper clarifies the translation in terms of maps between entailment systems, which provide an abstract meta-logical setting for formal systems within the theory of general logics [16]. The questions we address are: What are the entailment systems involved? How can the translation be explained at an abstract level that makes its correctness easy to verify? The interpretation stage on the other hand does not take place between different logics but within a single logic, so that an actual interpretation can be formally verified in a rigorous way. The critial inter-logic translation stage should be kept as simple as possible, so that we can achive a high confidence that it is indeed correct. Furthermore, the map of entailment systems defines a theory translator, that we have formally specified in Maude in an executable way, and we have used to translate large libraries of HOL theories into Nuprl, as explained in the last section. As another benefit of the proof-theoretic approach, the soundness result in terms of entailment systems immediately suggests extending the HOL/Nuprl connection to a translator that does not only translate theorems but also their proofs. In the last section we also report on some practical experience with an implementation of such a proof translator on top of the Nuprl system.

Ideally, there is a close connection between the soundness of the mapping between the two formal systems and proof translation: If $\mathcal{L}$ and $\mathcal{L}'$ are the source and target logics, respectively, and α is the mapping of $\mathcal{L}$ into $\mathcal{L}'$ then *soundness*[2] is the property that $\Gamma \vdash_{\mathcal{L}} P$ implies $\alpha(\Gamma) \vdash_{\mathcal{L}'} \alpha(P)$ for any set Γ of axioms and any formula P. Hence, a proof of soundness would demonstrate that for each proof of P from Γ in $\mathcal{L}$ there is a corresponding proof of $\alpha(P)$ from $\alpha(\Gamma)$ in $\mathcal{L}'$. Furthermore, if the soundness proof is conducted in a constructive way as

[2]Soundness is the central property of a map of entailments systems, which represent the notion of derivability in general logics [16]. For the purpose of the introduction we adopt a simplified viewpoint disregarding the issue of signature translation.

in this paper, it implicitly contains an algorithm for proof translation. In fact, in our case the proof translator from HOL into NuPRL is an algorithm, which is essentially extracted from a mathematical proof of soundness.

Obviously, soundness of the underlying mapping is the main theoretical requirement for the correctness of the translator. Notice, however, that soundness can always be achieved by extending the target system by additional axioms and inference rules, as it is often necessary in practice. Of course, such an extension could make the target system inconsistent, which is why the soundness proof is meaningful only in the presence of a consistency proof for the target system extension. Typically, such a consistency proof is done using an abstraction of the target system, e.g. by construction of a semantic model. In a more general setting, where we work relative to arbitrary theories, a suitably general *model preservation* property[3] property is that each model M of $\mathcal{L}$ can be obtained from a model M' of $\mathcal{L}'$ by a mapping β such that $\beta(M') \models_{\mathcal{L}} P$ iff $M' \models_{\mathcal{L}'} \alpha(P)$. Although the model preservation property, which in the case of HOL and Nuprl follows from Howe's hybrid computatonal/set-theoretic semantics for Nuprl [13, 12], is generally important for the translation mapping to be semantically meaningful, it is of little use for the implementation of a proof translator which typically relies on proof-theoretic constructions.

2 Preliminaries

In the sequel we often make use of finite lists of objects. We use [] for the empty list and a comma to denote list concatenation. Sometimes we employ a suggestive notation, like σ^m which makes explicit the length m of the list. If σ is a meta-variable ranging over certain objects, σ^m is a meta-variable ranging over lists of length m of such objects. The elements of the list can be accessed by σ_i^m for $i \in \{1 \ldots m\}$. Sometimes objects have a certain structure, for instance, if they are of the form $x : \sigma$ we also write $x^n : \sigma^n$ for a list of length n of such objects. So we assume notational conventions such as $x^n : \sigma^m = x_1^n : \sigma_1^n, \ldots, x_n^n : \sigma_n^n$.

3 HOL

HOL [11] is a proof development system based on higher-order logic. It uses an ML-style polymorphic lambda-calculus together with an axiomatization of the logic using polymorphic equality, implication and Hilbert's choice operator as basic ingredients. As most higher-order logics it is a logic of total functions. The HOL system favors conservative theory extensions (to introduce new constants and/or new data types) but axiomatic extensions are also supported.

[3]This property is usually expressed in the context of a map of institutions [10], which constitute the model-theoretic component of general logics [16].

The following presentation of HOL is close to [11] but we have cast it into a categorical setting which is suggested by the framework of general logics [16].

3.1 Syntactic Categories

We assume countably infinite disjoint sets of *constants*, *variables*, *type constants*, and *type variables*. Meta-variables for type constants and constants are ν and c, respectively. We use the meta-variables α, β, and γ for arbitrary but distinct type variables and meta-variables x, y, and z for arbitrary but distinct variables.

A *type constant declaration* takes the form $\nu : \mathsf{Type}^n \to \mathsf{Type}$, where ν is the *type constant* and n is its *arity*.[4] It is called an *atomic type declaration* if $n = 0$ (in this case we write $\nu : \mathsf{Type}$) and a *type operator declaration* otherwise. A *type signature* Ω is a finite set of type constant declarations with distinct type constants. We furthermore assume that a type signature Ω contains a *standard type constant declaration* $\triangleright\, \mathsf{o} : \mathsf{Type}$ and we refer to o as the *type of propositions*. The category of type signatures **HolTypeSign** has type signatures as objects, and its morphisms are functions mapping type constants in one signature to type constants in the other while preserving the standard type constants and the arity of each type constant.

A *type context* takes the form $\alpha^n : \mathsf{Type}^n$ with all the type variables α_i^n distinct. The category of type contexts is denoted by **HolTypeCtxt**. Its morphisms, called *type context inclusions*, are inclusions between the sets of declared type context variables. A *full type context* has the form $\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n$ and the category of full type contexts is **HolFullTypeCtxt** $=$ **HolTypeSign** $\times$ **HolTypeCtxt**.

The set $\mathcal{T}^{Hol}$ of *types* over a type signature Ω is inductively defined as follows:

1. $\alpha^n : \mathsf{Type}^n \;\triangleright\; \alpha_i^n$ is in $\mathcal{T}^{Hol}$ for each $i \in \{1...n\}$.
2. If $\alpha^n : \mathsf{Type}^n \;\triangleright\; \sigma_i^m$ is in $\mathcal{T}^{Hol}$ for each $i \in \{1...m\}$ and
   $\nu : \mathsf{Type}^m \to \mathsf{Type}$ is a type constant declaration in Ω, then
   $\alpha^n : \mathsf{Type}^n \;\triangleright\; \nu(\sigma^m)$ is in $\mathcal{T}^{Hol}$.
   We write ν instead of $\nu(\sigma^m)$ if $m = 0$.
3. If $\alpha^n : \mathsf{Type}^n \;\triangleright\; \sigma$ and $\alpha^n : \mathsf{Type}^n \;\triangleright\; \tau$ are in $\mathcal{T}^{Hol}$, then
   $\alpha^n : \mathsf{Type}^n \;\triangleright\; \sigma \to \tau$ is in $\mathcal{T}^{Hol}$.

An expression σ is said to be a *type* over a full type context $\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n$ if $\alpha^n : \mathsf{Type}^n \;\triangleright\; \sigma$ is a type over Ω. The set of types over $\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n$ is denoted by $\mathit{HolType}(\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n)$. Each full type context morphism is lifted to a function between sets of types in the natural way. This defines a functor $\mathit{HolType} : \mathbf{HolFullTypeCtxt} \to \mathbf{Set}$. A *full type* has the form $\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n \;\triangleright\; \sigma$, where σ is a type over $\Omega \;\triangleright\; \alpha^n : \mathsf{Type}^n$.

[4] Type is part of our suggestive notation, but it does not have a formal status in HOL.

Let Ω be a type signature. A *constant declaration* over Ω takes the form $\alpha^n : \mathsf{Type}^n \triangleright c : \sigma$, where σ is a type over $\Omega \triangleright \alpha^n : \mathsf{Type}^n$ and where each type variable α_i^n occurs in σ. Here c is a *constant* and σ is called the *type* of c. A *signature* Δ over Ω is a set of constant declarations over Ω with distinct constants. We assume that Δ contains *standard constant declarations* $\triangleright \Rightarrow : \mathsf{o} \to \mathsf{o} \to \mathsf{o}$ and $\alpha : \mathsf{Type} \triangleright = : \alpha \to \alpha \to \mathsf{o}$ for *logical implication* and *polymorphic equality*, respectively. A *signature morphism* between signatures Δ and Δ' over Ω is a function from Δ to Δ' which preserves standard constants and the type of each constant. The category of signatures over Ω is denoted by $\mathit{HolSign}(\Omega)$. Actually, we have a functor $\mathit{HolSign} : \mathbf{HolTypeSign} \to \mathbf{Cat}$ which lifts each type signature morphism $f : \Omega \to \Omega'$ to a functor $\mathit{HolSign}(f) : \mathit{HolSign}(\Omega) \to \mathit{HolSign}(\Omega')$. A *full signature* $\Omega \triangleright \Delta$ consists of a type signature Ω and a signature Δ over Ω. The category of full signatures is $\sum \Omega : \mathbf{HolTypeSign}\ .\ \mathit{HolSign}(\Omega)$ and is denoted by **HolFullSign**.[5]

Given a full type context $\Omega \triangleright \alpha^n : \mathsf{Type}^n$, a *context* over $\Omega \triangleright \alpha^n : \mathsf{Type}^n$ has the form $x^m : \sigma^m$, with distinct variables x_i^m and types σ_i^m over $\Omega \triangleright \alpha^n : \mathsf{Type}^n$. The category of contexts over $\Omega \triangleright \alpha^n : \mathsf{Type}^n$ is denoted by $\mathit{HolCtxt}(\Omega \triangleright \alpha^n : \mathsf{Type}^n)$ and its morphisms, called *context inclusions*, are inclusions between the sets of declared context variables such that the type of each variable is preserved.[6] Actually, we have a functor $\mathit{HolCtxt} : \mathbf{HolFullTypeCtxt} \to \mathbf{Cat}$ lifting full type context morphisms to functors. A *full context* takes the form $\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m$, where Δ is a signature over Ω and where $x^m : \sigma^m$ is a context over $\Omega \triangleright \alpha^n : \mathsf{Type}^n$. The category of *full contexts* is $\mathbf{HolFullCtxt} = \sum \Omega : \mathbf{HolTypeSign}\ .\ \mathit{HolSign}(\Omega) \times \sum \omega : \mathbf{HolTypeCtxt}\ .\ \mathit{HolCtxt}(\Omega \triangleright \omega)$.

Let $\Omega \triangleright \Delta$ be a full signature. The set $\mathcal{TT}^{Hol}$ of *typed terms* over $\Omega \triangleright \Delta$ is a subset of expressions of the form $\alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m \triangleright M : \tau$ such that $\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m$ is a full context. It is inductively defined as follows:[7]

1. $\alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m \triangleright x_i^m : \sigma_i^m$ is in $\mathcal{TT}^{Hol}$ for each $i \in \{1 \ldots m\}$.
2. If $\beta^k : \mathsf{Type}^k \triangleright c : \tau$ is a constant declaration in Δ and $\alpha^n : \mathsf{Type}^n \triangleright \tau_i^k$ is a type over Ω for each $i \in \{1 \ldots k\}$, then $\alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m \triangleright c_{\tau[\tau^k/\beta^k]} : \tau[\tau^k/\beta^k]$ is in $\mathcal{TT}^{Hol}$.
3. If $\alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m, y : \rho \triangleright M : \tau$ is in $\mathcal{TT}^{Hol}$, then $\alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m \triangleright (\lambda\, y : \rho\ .\ M) : \rho \to \tau$ is in $\mathcal{TT}^{Hol}$.

[5] For a functor $F : \mathbf{I} \to \mathbf{Cat}$ the category ΣF, also written $\sum x : \mathbf{I}\ .\ F(x)$, is defined by the general Grothendiek construction [2] which generalizes the set-theoretic sum.

[6] More powerful morphisms that allow renaming of context variables are possible if the notion of term is generalized so that *accidental hiding* of context variables by λ-abstractions is avoided. A possible framework to formalize this is provided by CINNI, the Calculus of Indexed Names and Named Indices [19].

[7] To avoid any technicalities connected with the treatment of bound variables we specialize the notion of terms in HOL by enforcing the convention that all bound and free variables are distinct.

4. If $\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd M : \rho \to \tau$ and $\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd N : \rho$ are in $\mathcal{TT}^{Hol}$, then $\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd (M\ N) : \tau$ is in $\mathcal{TT}^{Hol}$.

If $\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd M : \tau$ is a typed term over $\Omega \rhd \Delta$ we say that M is a *term* over $\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m$ and τ is its *type*. We will make use of the property of the HOL type system that M has a unique type. The set of *terms* over a full context $\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m$ is denoted by $\mathit{HolTerm}(\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m)$. Each full context morphism is lifted to a function between sets of terms. This defines a functor $\mathit{HolTerm} : \mathbf{HolFullCtxt} \to \mathbf{Set}$. On terms we assume the usual notion of α-conversion $\equiv_\alpha$, but we do *not* identify terms that are α-convertible.

A *formula* over a full context $\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m$ is a term over $\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m$ of type o. A *sentence* over $\Omega \rhd \Delta$ has the form $\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd A^k \Vdash A$ with formulae A_i^k and A over $\Omega \rhd \Delta \rhd \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m$. The set of sentences over $\Omega \rhd \Delta$ is denoted by $\mathit{HolSen}(\Omega \rhd \Delta)$. Each signature morphism can be lifted to a function between sets of sentences. So we have a functor $\mathit{HolSen} : \mathbf{HolFullSign} \to \mathbf{Set}$.

3.2 Deduction and Entailment

Given a full signature $\Sigma = (\Omega \rhd \Delta)$ the *deductive system of HOL* over Σ is the binary relation $(\vdash_1^{Hol}) \subseteq \mathcal{P}_{\mathrm{fin}}(\mathit{HolSen}(\Sigma)) \times \mathit{HolSen}(\Sigma)$ specified by the *inference rules of HOL*, which are the following:

$$\frac{}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd A \Vdash A} \quad (\mathsf{ASSUME})$$

$$\frac{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd G \Vdash A \qquad \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H \Vdash A \Rightarrow B}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H, G \Vdash B} \quad (\mathsf{MP})$$

$$\frac{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H \Vdash B}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd G \Vdash A \Rightarrow B} \quad (\mathsf{DISCH})$$

where G is obtained from H by removing all formulae α-equivalent to A.

$$\frac{}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd \ \Vdash M =^\rho M} \quad (\mathsf{REFL})$$

$$\frac{}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd \ \Vdash ((\lambda\, y : \rho\, .\, M)\, N) =^\tau M[N/y]} \quad (\mathsf{BETA\text{-}CONV})$$

$$\frac{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H_i^k \Vdash M_i^k =^{\rho_i^k} N_i^k \text{ for all } i \in \{1 \dots k\} \qquad \alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd G \Vdash A[M^k/z^k]}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H^k, G \Vdash A[N^k/z^k]} \quad (\mathsf{SUBST})$$

$$\frac{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m, y : \rho \rhd H \Vdash M =^\tau N}{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H \Vdash (\lambda\, y : \rho\, .\, M) =^{\rho \to \tau} (\lambda\, y : \rho\, .\, N)} \quad (\mathsf{ABS})$$

where y is not free in H.

$$\frac{\alpha^n : \mathsf{Type}^n \rhd x^m : \sigma^m \rhd H \Vdash A}{\beta^k : \mathsf{Type}^k \rhd x^m : \sigma^m[\tau^n/\alpha^n] \rhd H \Vdash A[\tau^n/\alpha^n]} \quad (\mathsf{INST\text{-}TYPE})$$

where $\beta^k : \mathsf{Type}^k \rhd \tau_i^n$ are types and α_i^n does not occur in H for $i \in \{1 \ldots n\}$.

For better readability we have written $M =^\rho N$ instead of $(((=_{\rho\to\rho\to\mathsf{o}})\ M)\ N)$.

As in the HOL system the assumptions on the lefthand side of $\Vdash$ are lists instead of sets, which is justified by the fact that HOL inference rules do not depend on their order. Another noteworthy point is that we have not identified α-equivalent terms. The α-conversion is, however, used implicitly in the HOL implementation of the rules MP and SUBST, which makes their implementation slighly more liberal than the rules given above. To compensate for this, we assume in addition to the rules above a rule for renaming of bound variables, thereby avoiding a commitment to the technicalities of the HOL implementation.

Derivability of a sentence ϕ from a set of sentences Γ in the deductive system of HOL is defined in the conventional way employing sequential proofs: A sentence ϕ can be *derived* from a set of sentences Γ iff there exists a sequence $\phi_1, \ldots, \phi_n$ such that **(1)** $\phi = \phi_n$ and **(2)** for all i either $\phi_i \in \Gamma$ or $\Gamma' \vdash_1^{Hol} \phi_i$ for some Γ' such that $\Gamma' \subseteq \Gamma \cup \{\phi_1, \ldots, \phi_{i-1}\}$. The sequence $\phi_1, \ldots, \phi_n$ is called a *sequential proof* of ϕ from Γ.

Given a full signature Σ, the deductive system of HOL defines the *entailment relation* $(\vdash_\Sigma^{Hol}) \subseteq \mathcal{P}_{\mathrm{fin}}(\mathit{HolSen}(\Sigma)) \times \mathit{HolSen}(\Sigma)$ of HOL as follows: $\Gamma \vdash_\Sigma^{Hol} \phi$ holds iff ϕ can be derived from Γ in the deductive system of HOL. According to the terminology of [16] $(\mathbf{HolFullSign}, \mathit{HolSen}, \vdash^{Hol})$ constitutes an entailment system. We call it the *entailment system of HOL*.

3.3 Theories

An *(axiomatic) HOL theory* (Σ, Γ) consists of a full signature Σ together with a set Γ of sentences over Σ called *axioms*. A full signature morphism $H : \Sigma \to \Sigma'$ is said to be a *theory morphism* $H : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ iff $\Gamma' \vdash_{\Sigma'}^{Hol} \mathit{HolSen}(H)(\phi)$ for all $\phi \in \Gamma$. This gives a category of theories that will be denoted by **HolTh**.

All mathematical developments in HOL take place in *standard theories* extending the *logical theory* `bool`. Therefore, for the remainder of this paper we define `bool` as `o`, and we use `bool` to emphasize that we are working with classical extensional logic. Indeed, it is the logical theory `bool` which makes HOL a logic of this kind. The logical theory `bool` $= (\Sigma, \Gamma)$ has a signature Σ which contains the standard type constant `bool` (i.e. `o`) and the standard constants $=$ and $\Rightarrow$. The remaining constants and axioms of `bool` are the following.

Σ contains the constants $\mathtt{T} : \mathtt{bool}$, $\mathtt{F} : \mathtt{bool}$, $\neg : \mathtt{bool} \to \mathtt{bool}$, $\wedge : \mathtt{bool} \to \mathtt{bool} \to \mathtt{bool}$, $\vee : \mathtt{bool} \to \mathtt{bool} \to \mathtt{bool}$, $\forall : (\alpha \to \mathtt{bool}) \to \mathtt{bool}$, $\exists : (\alpha \to \mathtt{bool}) \to \mathtt{bool}$, and Hilbert's choice operator $\epsilon : (\alpha \to \mathtt{bool}) \to \alpha$.

Γ contains suitable definitional axioms to equip these constants with their standard (classical) meaning. Indeed all the constants above are defined in terms of (higher-order) equality $=$, implication $\Rightarrow$ and Hilbert's choice operator ϵ. Moreover, there are some non-definitional axioms in Γ, namely

$\Vdash\ \forall x : \texttt{bool} \,.\, (x =^{\texttt{bool}} \texttt{T}) \vee (x =^{\texttt{bool}} \texttt{F})$ (BOOL-CASES-AX)

$\Vdash\ \forall x : \texttt{bool} \,.\, \forall y : \texttt{bool} \,.\, (x \Rightarrow y) \Rightarrow (y \Rightarrow x) \Rightarrow (x =^{\texttt{bool}} y)$ (IMP-ANTISYM-AX)

$f : \alpha \to \beta\ \Vdash\ (\lambda\, x : \alpha\ .\ (f\, x)) =^{\beta} f$ (ETA-AX)

$\Vdash\ \forall P : \alpha \to \texttt{bool} \,.\, \forall x : \alpha \,.\, (P\, x) \Rightarrow (P\, (\epsilon\, P))$ (SELECT-AX)

with the usual abbreviation $\forall x : \sigma \,.\, M$ for $\forall(\lambda\, x : \sigma\ .\ M)$.

In addition, Σ contains an atomic type constant `ind`, and there is an axiom INFINITY-AX in Γ stating that `ind` has an infinite number of elements.

4 Nuprl

Nuprl type theory [7] is a variant of Martin-Löf's 1982 polymorphic, extensional type theory (the version contained in [18] with extensional equality). Although Nuprl has very advanced features (e.g. subset types, subtyping, quotient types, recursive types, intersection types, partial functions, and direct computation, which make these type theories rather different) semantically Nuprl can be viewed as an extension of Martin-Löf's type theory, in the sense that it has a richer variety of types and more flexible rules which give rise to a richer collection of well-typed terms.

Apart from the advanced features mentioned above, there is a major difference in the meaning of sequents between these two type theories. In Martin-Löf's type theory well-formedness of the lefthand side of a sequent is presupposed, i.e., it is implied by a derivable sequent. This is not the case in Nuprl, where well-formedness is part of the assumption expressed by the lefthand side of the sequent. Details about this difference can be found in [1]. For our purposes here this difference is not important, since we consider only rules where all well-formedness conditions are made explicit. In this way we obtain a set of rules which are valid and derivable in Martin-Löf's type theory as well as in Nuprl.

In constrast to HOL, terms in Nuprl are neither explicitly nor implicitly equipped with types. Instead types are ordinary terms, and the judgement that a type can be *assigned* to a term is a sentence in the logical language which is not decidable in general. Indeed, since Nuprl is polymorphic, a term may be associated with different types.

Even though the advanced features of Nuprl provide an important motivation for the HOL/Nuprl connection, the connection itself does not rely on features that go beyond Martin-Löf's type theory as presented in [18]. Indeed, in the following we present the part of classical Nuprl that is sufficent to establish the HOL/Nuprl connection. We give a simplified presentation based on [7], but, as for HOL, we have used a categorical formulation in order to achive a uniform and systematic description of the translation.

4.1 Syntactic Categories

We assume countably infinite disjoint sets of *constants* and *variables* with the same meta-variable conventions as for HOL.

A *Nuprl signature* Σ is a set of constants. We assume that Σ contains a number of *standard constants* including $\mathbb{U}_i$ for $i \geq 1$, `app`, λ, `pi` and further constants that we do not explicitly mention here. We denote the category of signatures and bijective functions preserving standard constants by **NuprlSign**.

Given a signature Σ we inductively define the set $\mathcal{T}^{Nuprl}$ of *terms* over Σ as follows:

1. $x^m \triangleright x_i^m$ is in $\mathcal{T}^{Nuprl}$ for each $i \in \{1 \ldots m\}$.
2. If c is a constant in Σ and
   $x^m \triangleright M_i^k$ is in $\mathcal{T}^{Nuprl}$ for each $i \in \{1 \ldots k\}$, then
   $x^m \triangleright c(M^k)$ is in $\mathcal{T}^{Nuprl}$.
   We write c instead of $c(M^k)$ if $k = 0$.
3. If $x^m, y \triangleright M$ is in $\mathcal{T}^{Nuprl}$, then
   $x^m \triangleright (y.M)$ is in $\mathcal{T}^{Nuprl}$.

If $x^m \triangleright M$ is a term over Σ we say that M is a *term* over $\Sigma \triangleright x^m$. Instead of making constants c explicit by writing $c(M^k)$ we often use a more convenient notation[8] such as $(M\ N)$ for function application $\mathtt{app}(M, N)$, $\lambda\ x\ .\ M$ for λ-abstraction $\lambda(x.M)$, and $(x : S \to T)$ for dependent types $\mathtt{pi}(S, (x.T))$.

Given a signature Σ, a *context* over Σ takes the form $x^m : S^m$, where S_{i+1}^m is a term over $\Sigma \triangleright x_1^m, ..., x_i^m$ and where all the variables x_i^m are distinct. A *context inclusion* is an inclusion between the sets of declared context variables such that the type of each variable is preserved. A *full Nuprl context* is of the form $\Sigma \triangleright x^m : S^m$, where $x^m : S^m$ is a context over Σ. The category of contexts over Σ with context inclusions as morphisms is denoted by $\mathit{NuprlCtxt}(\Sigma)$. Furthermore, each signature morphism can be lifted to a morphism between contexts, so that we obtain a functor $\mathit{NuprlCtxt} : \mathbf{NuprlSign} \to \mathbf{Cat}$. The category of full contexts is $\mathbf{NuprlFullCtxt} = \sum\ \Sigma : \mathbf{NuprlSign}\ .\ \mathit{NuprlCtxt}(\Sigma)$.

A *term* over a full context $\Sigma \triangleright x^m : S^m$ is precisely a term over $\Sigma \triangleright x^m$. The set of terms over $\Sigma \triangleright x^m : S^m$ is denoted by $\mathit{NuprlTerm}(\Sigma \triangleright x^m : S^m)$. Each full context morphism is lifted to a function between sets of terms. This defines a functor $\mathit{NuprlTerm} : \mathbf{NuprlFullCtxt} \to \mathbf{Set}$.

Given a signature Σ we also define *sentences* over Σ, which have the form $x^m : S^m \Vdash T$, with a context $x^m : S^m$ over Σ and a term T over $\Sigma \triangleright x^m : S^m$. The set of sentences over Σ is denoted by $\mathit{NuprlSen}(\Sigma)$. For each fixed signature Σ this is a category if we take context inclusions lifted to sentences as

[8] Nuprl employs user-definable display forms to bridge the gap between the formal notion of term and the way terms are presented to a user of the system.

morphisms. Moreover, signature morphisms can be lifted to functors between such categories, so that we have a functor $\mathit{NuprlSen} : \mathbf{NuprlSign} \to \mathbf{Cat}$.

It is worthwhile mentioning that this notion of sentence is a proper specialization of the judgements admitted in [7], which take the form $x^m : S^m \ \Vdash\ T\,[\mathrm{ext}\,P]$, the pragmatic intention being that the extraction term P is usually hidden from the user, but it can be extracted from a completed proof. Assuming that T is intended as a proposition, we are usually not interested in the extraction term P. Therefore we will only use *abstract judgements* of the form $x^m : S^m \ \Vdash\ T$. We define such an abstract judgement to be derivable iff $x^m : S^m \ \Vdash\ T\,[\mathrm{ext}\,P]$ is derivable for some P. To make explicit an element P that inhabits a type T, we use equality types of the form $M = N \in T$, so that we can derive $x^m : S^m \ \Vdash\ T\,[\mathrm{ext}\,P]$ iff $x^m : S^m \ \Vdash\ P \in T$ is derivable, where we assume that $P \in T$ is identified with $P = P \in T$.

4.2 Derivability and Entailment

We define an *entailment relation* $(\vdash_\Sigma^{Nuprl}) \subseteq \mathcal{P}_{\mathrm{fin}}(\mathit{NuprlSen}(\Sigma)) \times \mathit{NuprlSen}(\Sigma)$ where $\Gamma \vdash_\Sigma^{Nuprl} \phi$ holds iff the sentence ϕ can be derived using the sentences in Γ as assumptions. Here, we consider derivability in the fragment of classical Nuprl given by the inference rules below, which are either basic-inference rules or trivially derivable. $(\mathbf{NuprlSign}, \mathit{NuprlSen}, \vdash^{Nuprl})$ constitutes an entailment system. We call it the *entailment system of Nuprl.*

$$\frac{H, G \ \Vdash\ T : \mathbb{U}_i}{H, x : T, G \ \Vdash\ x \in T} \qquad \text{(HYP)}$$

$$\frac{H \ \Vdash\ S : \mathbb{U}_i \quad H, G \ \Vdash\ T}{H, x : S, G \ \Vdash\ T} \ \text{if } x \text{ is not declared in } H \text{ or } G \qquad \text{(THIN)}$$

$$\frac{H, G \ \Vdash\ N \in S \quad H, x : S, G \ \Vdash\ T}{H, G \ \Vdash\ T[N/x]} \qquad \text{(CUT)}$$

$$\frac{H, x : S \ \Vdash\ T : \mathbb{U}_i \quad H \ \Vdash\ T[M/x] \quad H \ \Vdash\ M = N \in S}{H \ \Vdash\ T[N/x]} \qquad \text{(SUBST)}$$

$$\frac{}{H \ \Vdash\ \mathbb{U}_i \in \mathbb{U}_j} \ \text{if } i < j \qquad \text{(UNIV-INTRO)}$$

$$\frac{H \ \Vdash\ M \in \mathbb{U}_i}{H \ \Vdash\ M \in \mathbb{U}_j} \ \text{if } i < j \qquad \text{(CUMULATIVITY)}$$

$$\frac{H \ \Vdash\ T \in \mathbb{U}_i \quad H \ \Vdash\ M \in T \quad H \ \Vdash\ N \in T}{H \ \Vdash\ (M = N \in T) \in \mathbb{U}_i} \qquad \text{(EQUALITY-FORM)}$$

$$\frac{H \ \Vdash\ M = N \in T}{H \ \Vdash\ N = M \in T} \qquad \text{(EQUALITY-SYM)}$$

$$\frac{H \ \Vdash\ K = L \in T \quad H \ \Vdash\ L = M \in T}{H \ \Vdash\ K = M \in T} \qquad \text{(EQUALITY-TRANS)}$$

$$\frac{H \ \Vdash\ A \in \mathbb{U}_i \quad H, x : A \ \Vdash\ B \in \mathbb{U}_i}{H \ \Vdash\ (x : A \to B) \in \mathbb{U}_i} \qquad \text{(FUN-FORM)}$$

$$\frac{H \Vdash S \in \mathbb{U}_i \quad H, x : S \Vdash M \in T}{H \Vdash (\lambda\, x\,.\, M) \in (x : S \to T)} \qquad \text{(FUN-INTRO)}$$

$$\frac{H \Vdash N \in S \quad H \Vdash M \in (x : S \to T)}{H \Vdash (M\ N) \in T[N/x]} \qquad \text{(FUN-ELIM)}$$

$$\frac{H \Vdash S \in \mathbb{U}_i \quad H, x : S \Vdash M = N \in T}{H \Vdash (\lambda\, x\,.\, M) = (\lambda\, x\,.\, N) \in (x : S \to T)} \qquad \text{(FUN-XI)}$$

$$\frac{H \Vdash N \in S \quad H, x : S \Vdash M \in T}{H \Vdash ((\lambda\, x\,.\, M)\ N) = M[N/x] \in T[N/x]} \qquad \text{(FUN-BETA)}$$

$$\frac{\begin{array}{c} H \Vdash f \in (x : A \to B) \quad H \Vdash g \in (x : A \to B) \\ H, x : A \Vdash (f\ x) = (g\ x) \in B \end{array}}{H \Vdash f = g \in (x : A \to B)} \qquad \text{(FUN-EXT)}$$

In addition to these rules we assume a rule for renaming of bound variables, and we make use of standard rules for the *empty type* `Void`, the *singleton type* `Unit` with an element $\bullet$, the *boolean type* $\mathbb{B}$ with elements `tt` and `ff` and elimination operator `if` M `then` N `else` N', the *type of natural numbers* $\mathbb{N}$ with standard operations, the *dependent product types* $x : S \times T$ with a pair constructor $<M, N>$ and elimination operators $\mathtt{1of}(M)$, $\mathtt{2of}(M)$, the *disjoint union types* $S + T$ with left and right injections $\mathtt{inl}(M)$, $\mathtt{inr}(M)$ and elimination operator $\mathtt{decide}(M, (x.N), (x.N'))$, and *subset types* $\{x : T \mid A\}$. Finally, we assume a standard constant `inhabited` with the classical axiom scheme given later.

The propositions-as-types interpretation is made explicit using the following logical abbreviations: $\mathbb{P}_i = \mathbb{U}_i$, $\mathtt{False} = \mathtt{Void}$, $\mathtt{True} = \mathtt{Unit}$, $A \wedge B = A \times B$, $A \vee B = A + B$, $\forall x : A\,.\,B = x : A \to B$, and $\exists x : A\,.\,B = x : A \times B$. If x is not free in B we write $A \Rightarrow B$ and $A \to B$ instead of $\forall x : A\,.\,B$ and $x : A \to B$, respectively. Furthermore, $\neg A$ abbreviates $A \Rightarrow \mathtt{False}$.

4.3 Classical Extension

For the translation of HOL's equality we wish to define a boolean polymorphic equality using Nuprl's propositional equality, but so far we do not have any means for converting a proposition into a boolean, which amounts to deciding whether a propositional type is inhabited. So we add a standard constant `inhabited` and we assume axioms $\Vdash \mathtt{inhabited} \in x : \mathbb{U}_i \to x + (x \to \mathtt{Void})$ stating that $\mathtt{inhabited}(T)$ decides if its argument, a type T in $\mathbb{U}_i$, is inhabited, and that it returns an element of T if this is the case. Using the new axioms we can easily define an operator $\downarrow_b P = \mathtt{decide}(\mathtt{inhabited}(P), (x.\mathtt{tt}), (x.\mathtt{ff}))$ that casts a propositional type into a boolean by deciding the proposition. Here we employ `decide` to perform case analysis for elements of a disjoint union type.

4.4 Theories

An *(axiomatic) Nuprl theory* (Σ, Γ) consists of a signature Σ together with a set Γ of sentences over Σ called *axioms*. The set of all such Γ over a signature

Σ is denoted by $\mathit{NuprlAxSet}(\Sigma)$. A signature morphism $H : \Sigma \to \Sigma'$ is said to be a *theory morphism* $H : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ iff $\Gamma' \vdash^{Nuprl}_{\Sigma'} \mathit{NuprlSen}(H)(\phi)$ for all $\phi \in \Gamma$. This gives a category of theories that will be denoted by **NuprlTh**.

A theory morphism $H : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ is called *axiom-preserving* iff *NuprlSen* $(H)(\Gamma) \subseteq \Gamma'$. The subcategory of **NuprlTh** with the objects of **NuprlTh** but morphisms restricted to axiom-preserving morphisms is denoted by $\mathbf{NuprlTh_0}$. This will be the target category of the translation between HOL and Nuprl theories. **NuprlTh**, on the other hand, is used for the second stage of the HOL/Nuprl connection, namely, the interpretation of Nuprl theories.

5 Theory Translation

In accordance with the general logics terminology introduced in [16], which is used as a guideline for the following definitions, the translation from HOL theories to Nuprl theories will be specified by a functor

$$\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlTh_0}$$

which translates HOL signatures into Nuprl theories and a natural transformation

$$\alpha : \mathit{HolSen} \to \mathit{NuprlSen} \circ \Phi$$

where

$$\mathit{HolSen} : \mathbf{HolFullSign} \to \mathbf{Cat}$$

$$\mathit{NuprlSen} : \mathbf{NuprlTh_0} \to \mathbf{Cat}$$

which translates HOL sentences into Nuprl sentences.

First we define the functor

$$\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlTh_0}$$

$$\Phi(\Sigma) = (\Phi(\Sigma), \Psi(\Sigma))$$

with a functor Φ and a (dependent) function Ψ (to be defined)

$$\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$$

$$\Psi : \prod \Sigma : \mathbf{HolFullSign} \,.\, \mathit{NuprlAxSet}(\Phi(\Sigma))$$

where

$$\mathit{NuprlAxSet} : \mathbf{NuprlSign} \to \mathbf{Cat}$$

We define the functor $\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlTh_0}$ by $\Phi(\Sigma) = (\Phi(\Sigma), \Psi(\Sigma))$ with a functor $\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$ and a (dependent) function $\Psi : \prod \Sigma : \mathbf{HolFullSign} \,.\, \mathit{NuprlAxSet}(\Phi(\Sigma))$ (to be defined below).

Applying Φ to a full HOL signature $\Sigma = (\Omega \triangleright \Delta)$ yields a Nuprl theory (Σ', Γ') consisting of the signature part Σ' and the axiom part Γ'.

Here, and in the following, we do not define the behaviour of functors on morphisms explicitly. Actually, there is only one natural and straightforward way to extend our definitions to morphisms.

We deal with the two levels Ω and Δ separately. First, we deal with the type signature Ω, which contains type constants to be translated into a Nuprl signature part $\Phi(\Omega)$ and an axiom part $\Psi(\Omega)$, where Φ is a functor and Ψ is a (dependent) function:

$$\Phi : \mathbf{HolTypeSign} \to \mathbf{NuprlSign}$$

$$\Psi : \prod \Omega : \mathbf{HolTypeSign}\,.\, \mathit{NuprlAxSet}(\Phi(\Omega))$$

where

$$\mathit{NuprlAxSet} : \mathbf{NuprlSign} \to \mathbf{Cat}$$

1. $\Phi(\{\nu : \mathsf{Type}^n \to \mathsf{Type}\}) = \{\nu\}$,
2. $\Phi(\emptyset) = \emptyset$,
3. $\Phi(\Omega_1 \cup \Omega_2) = \Phi(\Omega_1) \cup \Phi(\Omega_2)$
4. $\Psi(\{\nu : \mathsf{Type}^n \to \mathsf{Type}\}) = \{\alpha^n : \mathsf{S} \Vdash \nu(\alpha^n) \in \mathsf{S}\}$,
5. $\Psi(\emptyset) = \emptyset$,
6. $\Psi(\Omega_1 \cup \Omega_2) = \Psi(\Omega_1) \cup \Psi(\Omega_2)$

Since semantically all HOL types are assumed to be nonempty, we have employed $x : \mathsf{S}$ as a suggestive notation for $x : \mathbb{U}_1,\ inh_x : \mathtt{Inh}(x)$ where inh_x is a reserved variable that is associated to x only and $\mathtt{Inh}(x)$ is defined as $\exists z : x\,.\,\mathtt{True}$. Likewise, $x \in \mathsf{S}$ is a suggestive notation for $x \in \mathbb{U}_1 \wedge \mathtt{Inh}(x)$.

We also we need to translate HOL type contexts $\alpha^n : \mathsf{Type}^n$ into Nuprl contexts giving a functor:

$$\Phi : \mathbf{HolTypeCtxt} \to \mathbf{NuprlCtxt}$$

$$\Phi(\alpha^n : \mathsf{Type}^n) = \alpha^n : \mathsf{S}^n$$

This functor can be lifted to full type contexts:

$$\Phi : \mathbf{HolFullTypeCtxt} \to \mathbf{NuprlFullCtxt}$$

$$\Phi(\Omega \triangleright \alpha^n : \mathsf{Type}^n) = \Phi(\Omega) \triangleright \Phi(\alpha^n : \mathsf{Type}^n)$$

To specify the translation Ψ of the signature Δ over Ω which contains constant declarations over Ω we need the following natural transformation

$$\alpha : \mathit{HolType} \to \mathit{NuprlTerm} \circ \Phi,$$

where:

$$\textit{HolType} : \mathbf{HolFullTypeCtxt} \to \mathbf{Set}$$

$$\Phi : \mathbf{HolFullTypeCtxt} \to \mathbf{NuprlFullCtxt}$$

$$\textit{NuprlTerm} : \mathbf{NuprlFullCtxt} \to \mathbf{Set}$$

translating HOL types into Nuprl types, which are terms in Nuprl:

1. $\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\alpha_i^n) = \alpha_i^n$,
2. $\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\mathsf{o}) = \mathbb{B}$,
3. $\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma \to \tau) =$
   $\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma) \to \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\tau)$,
4. $\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\nu(\sigma_1, \ldots, \sigma_n)) =$
   $\nu(\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma_1), \ldots, \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma_n))$

Next, we give the translation of constants Δ over a type signature Ω. Again, we have a signature part $\alpha(\Omega)(\Delta)$ and an axiom part $\beta(\Omega)(\Delta)$, where α is a natural transformation and β is a type-signature-indexed family of (dependent) functions:

$$\alpha : \textit{HolSign} \to \mathbf{NuprlSign}$$

$$\beta(\Omega) : \prod \Delta : \textit{HolSign}(\Omega) \ . \ \textit{NuprlAxSet}(\alpha(\Omega)(\Delta))$$

where

$$\textit{HolSign} : \mathbf{HolTypeSign} \to \mathbf{Cat}$$

$$\mathbf{NuprlSign} : \mathbf{HolTypeSign} \to \mathbf{Cat}$$

$$\textit{NuprlAxSet} : \mathbf{NuprlSign} \to \mathbf{Cat}$$

1. $\alpha(\Omega)(\{\alpha^n : \mathsf{Type}^n \triangleright c : \sigma\}) = \{c\}$,
2. $\alpha(\Omega)(\emptyset) = \emptyset$,
3. $\alpha(\Omega)(\Delta_1 \cup \Delta_2) = \alpha(\Omega)(\Delta_1) \cup \alpha(\Omega)(\Delta_2)$
4. $\beta(\Omega)(\{\alpha^n : \mathsf{Type}^n \triangleright c : \sigma\}) =$
   $\{\alpha^n : \mathsf{S}^n \Vdash c(\alpha^n) \in \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma)\}$,
5. $\beta(\Omega)(\emptyset) = \emptyset$,
6. $\beta(\Omega)(\Delta_1 \cup \Delta_2) = \beta(\Omega)(\Delta_1) \cup \beta(\Omega)(\Delta_2)$

The previous definitions are lifted to full signatures as follows:

$$\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$$

$$\Psi : \prod \Sigma : \mathbf{HolFullSign} \ . \ \textit{NuprlAxSet}(\Phi(\Sigma))$$

1. $\Phi(\Omega \triangleright \Delta) = \Phi(\Omega) \cup \alpha(\Omega)(\Delta)$
2. $\Psi(\Omega \triangleright \Delta) = \Psi(\Omega) \cup \beta(\Omega)(\Delta)$

Next, we define the translation of contexts $x^m : \sigma^m$, that depends on type contexts and gives therefore rise to a natural transformation:

$$\alpha : \mathit{HolCtxt} \to \mathit{NuprlCtxt} \circ \Phi$$

$$\mathit{HolCtxt} : \mathbf{HolFullTypeCtxt} \to \mathbf{Cat}$$

$$\Phi : \mathbf{HolFullTypeCtxt} \to \mathbf{NuprlFullCtxt}$$

$$\mathit{NuprlCtxt} : \mathbf{NuprlFullCtxt} \to \mathbf{Cat}$$

$$\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(x^m : \sigma^m) = x^m : \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\sigma^m)$$

We now lift the natural transformation α to a functor Φ between full contexts:

$$\Phi : \mathbf{HolFullCtxt} \to \mathbf{NuprlFullCtxt}$$

$$\Phi(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m) =$$
$$\Phi(\Omega \triangleright \Delta) \triangleright \Phi(\alpha^n : \mathsf{Type}^n), \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(x^m : \sigma^m)$$

Our goal is the translation of HOL sentences into Nuprl sentences. Since HOL sentences are constructed from terms we define their translation first:

$$\alpha : \mathit{HolTerm} \to \mathit{NuprlTerm} \circ \Phi$$

where

$$\mathit{HolTerm} : \mathbf{HolFullCtxt} \to \mathbf{Set}$$

$$\Phi : \mathbf{HolFullCtxt} \to \mathbf{NuprlFullCtxt}$$

$$\mathit{NuprlTerm} : \mathbf{NuprlFullCtxt} \to \mathbf{Set}$$

1. $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(x_i^m) = x_i^m$,
2. $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(=_{\tau \to \tau \to \mathsf{o}}) = (=_{\mathsf{b}}^{\alpha(\Omega)(\tau)})$,
3. $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(\Rightarrow_{\mathsf{o} \to \mathsf{o} \to \mathsf{o}}) = (\Rightarrow_{\mathsf{b}})$,
4. $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(c_{\tau[\tau^k/\beta^k]}) =$
   $c(\alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\tau_1^k), \ldots, \alpha(\Omega \triangleright \alpha^n : \mathsf{Type}^n)(\tau_k^k))$,
   where $(\beta^k : \mathsf{Type}^k \triangleright c : \tau) \in \Delta$ and $\alpha^n : \mathsf{Type}^n \triangleright \tau_i^k$ are types over Ω,
5. $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(M\ N)) =$
   $(\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(M)$
   $\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(N))$,

6. $\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)(\lambda\, x : \sigma \,.\, M) =$
$(\lambda\, x \,.\, \alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m, x : \sigma)(M))$

where we have employed the abbreviations

1. $\Rightarrow_{\mathsf{b}}$ is the boolean implication defined as
$\Rightarrow_{\mathsf{b}} = \lambda\, x, y \,.\, \texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ \texttt{tt}$,
2. $=_{\mathsf{b}}^{T}$ is the polymorphic equality defined as
$=_{\mathsf{b}}^{T} = \lambda\, x, y \,.\, \downarrow_b(x = y \ \in \ T)$.

Now the natural transformation

$$\alpha : \mathit{HolSen} \to \mathit{NuprlSen} \circ \Phi$$

with

$$\mathit{HolSen} : \mathbf{HolFullSign} \to \mathbf{Cat}$$

$$\mathit{NuprlSen} : \mathbf{NuprlSign} \to \mathbf{Cat}$$

$$\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$$

of HOL sentences is given by

$$\alpha(\Omega \ \triangleright \ \Delta)(\alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m \ \triangleright \ H \ \Vdash \ A) =$$

$$\Phi(\alpha^n : \mathsf{Type}^n), \alpha(\Omega \ \triangleright \ \alpha^n : \mathsf{Type}^n)(x^m : \sigma^m),$$

$$\alpha(\Omega \triangleright \Delta \triangleright \alpha^n : \mathsf{Type}^n \triangleright x^m : \sigma^m)(H) \ \Vdash \uparrow\!\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)(A)$$

with

1. $\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)([]) = []$
2. $\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)([A_1, \ldots, A_n]) =$
$[\ h_1 \ : \uparrow\!\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)(A_1), \ldots,$
$h_n \ : \uparrow\!\alpha(\Omega \ \triangleright \ \Delta \ \triangleright \ \alpha^n : \mathsf{Type}^n \ \triangleright \ x^m : \sigma^m)(A_n)\ ]$

Above $h_1, \ldots, h_n$ are reserved variables of Nuprl, that are distinct from HOL variables and are not used for any other purpose.

6 Correctness of the Translation

In the following we adapt some terminology from the theory of general logics [16]. To justify the translation we introduce an appropriate notion of map between entailment systems. For the sake of brevity, we will immediately instantiate the general notions to the specific categories and functors we are interested in.

First we observe that the functor $\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlTh_0}$ can be easily extended to a functor $\Phi : \mathbf{HolTh_0} \to \mathbf{NuprlTh_0}$, called the *$\alpha$-extension to* *theories*, by mapping an HOL-theory (Σ, Γ) to the Nuprl theory (Σ', Γ') with $\Sigma' = \Phi(\Sigma)$ and $\Gamma' = \Psi(\Sigma) \cup \alpha_\Sigma(\Gamma)$.

We also extend the functors $\mathit{HolSen} : \mathbf{HolFullSign} \to \mathbf{Cat}$ and $\mathit{NuprlSen} : \mathbf{NuprlSign} \to \mathbf{Cat}$ to functors $\mathit{HolSen} : \mathbf{HolTh_0} \to \mathbf{Cat}$ and $\mathit{NuprlSen} : \mathbf{NuprlTh_0} \to \mathbf{Cat}$ by just ignoring the axioms in the theories.

Correspondingly, we extend the natural transformation $\alpha : \mathit{HolSen} \to \mathit{NuprlSen} \circ \Phi$ with $\mathit{HolSen} : \mathbf{HolFullSign} \to \mathbf{Cat}$ and $\mathit{NuprlSen} : \mathbf{NuprlSign} \to \mathbf{Cat}$ to a natural transformation between $\mathit{HolSen} : \mathbf{HolTh_0} \to \mathbf{Cat}$ and $\mathit{NuprlSen} \circ \Phi : \mathbf{HolTh_0} \to \mathbf{Cat}$ by setting $\alpha_{(\Sigma,\Gamma)} = \alpha_\Sigma$.

Now $\Phi : \mathbf{HolTh_0} \to \mathbf{NuprlTh_0}$ is an α-*simple* functor [16], i.e., it is the α-extension to theories of a functor from signatures to theories. However, since the axiomatic part in the target of the translation is essential, Φ is *not* α-plain [16], i.e., it is *not* the α-extension of a functor that factors through a functor $F : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$ mapping a signature to a theory with an empty set of axioms.

Correctness Theorem

$(\Phi, \alpha) : (\mathbf{HolFullSign}, \mathit{HolSen}, \vdash^{Hol}) \to (\mathbf{NuprlSign}, \mathit{NuprlSen}, \vdash^{Nuprl})$ is a *map between entailment systems*, i.e., the following properties are satisfied:

1. $\alpha : \mathit{HolSen} \to \mathit{NuprlSen} \circ \Phi$ is a natural transformation,
2. $\Phi : \mathbf{HolTh_0} \to \mathbf{NuprlTh_0}$ is α-sensible (see below),
3. $\Gamma \vdash^{Hol}_\Sigma \phi \Rightarrow \Psi(\Sigma) \cup \alpha_\Sigma(\Gamma) \vdash^{Nuprl}_{\Phi(\Sigma)} \alpha_\Sigma(\phi)$ for each full HOL signature Σ. As before $\Psi(\Sigma)$ denotes the axiom part of $\Phi(\Sigma)$.

Proof Sketch

Concerning (1) we have not made explicit the translation of morphisms. It should, however, be clear that there is only one natural choice and α is a natural transformation, since it is compatible with all renaming morphisms.

In condition (2) $\Phi : \mathbf{HolTh_0} \to \mathbf{NuprlTh_0}$ is called α-sensible, iff: (a) there is a functor $\Phi^\diamond : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$ such that $\mathit{NuprlSign} \circ \Phi = \Phi^\diamond \circ \mathit{HolFullSign}$, where $\mathit{HolFullSign} : \mathbf{HolTh_0} \to \mathbf{HolFullSign}$ and $\mathit{NuprlSign} : \mathbf{NuprlTh_0} \to \mathbf{HolFullSign}$ are the obvious forgetful functors, and (b) $\Gamma'^\bullet = (\emptyset' \cup \alpha_\Sigma(\Gamma))^\bullet$ whenever $\Phi(\Sigma, \emptyset) = (\Sigma', \emptyset')$ and $\Phi(\Sigma, \Gamma) = (\Sigma', \Gamma')$. Here we use the *deductive closure* w.r.t. $\vdash^{Nuprl}$, i.e. $\Gamma^\bullet = \{\phi : \Gamma \vdash^{Nuprl} \phi\}$.

This is clearly satisfied in our case (chosing $\Phi : \mathbf{HolFullSign} \to \mathbf{NuprlSign}$ for $\Phi^\diamond$). Actually, the fact that Φ is simple is a stronger property (we have $\Gamma' = \emptyset' \cup \alpha_\Sigma(\Gamma)$) which shows that we do not need the full generality of maps between entailment systems here.

In order to verify (3), namely that the entailment relation is preserved, it is sufficient to verify that the translations of HOL inference rules can be derived in Nuprl assuming the axioms $\Psi(\Sigma)$. To minimize meta-logical reasoning we use of a number of simple theorems that have been verified inside Nuprl:

```
*T assert_bequal_1 ∀T:𝕌. ∀x,y:T.  x = y ⇒ ↑(x =_b y)
*T assert_bequal_2 ∀T:𝕌. ∀x,y:T.  ↑(x =_b y) ⇒ x = y
*T REFL_lemma      ∀T:𝕌. ∀t:T.  ↑(t =_b t)
*T SUBST_lemma     ∀T:𝕌. ∀a,b:T. ∀p:T → 𝕌.  ↑(a =_b b) ⇒ (p a) ⇒ (p b)
*T DISCH_lemma     ∀p,q:𝔹.  (↑p ⇒ ↑q) ⇒ ↑(p ⇒_b q)
*T MP_lemma        ∀t1,t2:𝔹.  ↑(t1 ⇒_b t2) ⇒ ↑t1 ⇒ ↑t2
```

Indeed, using these theorems, most of the translated inference rules have surprisingly short proofs in Nuprl. For details we refer to the extended version of this paper which is available via WWW at `www.csl.sri.com/~stehr`. Even though meta-logical reasoning can be minimized by using the theorems above, the meta-logical nature of these proofs is unavoidable. The proof translator to be discussed in Section 8.2 can be seen as a formalization of their computational content of these meta-logical proofs using the metalanguage of the Nuprl system, which is ML.

7 Nuprl Theory Interpretations

Typically, we are not interested in the full generality of using an axiomatic Nuprl theory which has been obtained by translation from an axiomatic HOL theory. Instead, we may prefer to *instantiate constants* with fixed meanings. One reason may be that we want them to be *computationally meaningful*.[9] This (partial) instantiation of constants leads from one axiomatic Nuprl theory to another one, typically with a more restricted class of models. Usually, some or all of the original axioms become theorems in the new theory. Ideally, we obtain a definitional theory, which means that we have constructed an *internal model* of an axiomatic Nuprl theory inside Nuprl itself.

Given Nuprl theories (Σ, Γ) and (Σ', Γ') we say that (Σ, Γ) is *interpreted* in (Σ', Γ') by I iff $I : (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ is a theory morphism in the category **NuprlTh**. Notice that these morphisms are not necessarily axiom-preserving, since it is typically the point of such an interpretation to get rid of axioms.

Since we have to verify $\Gamma' \vdash^{Nuprl}_{\Sigma'} \mathit{NuprlSen}(I)(\phi)$ for all $\phi \in \Gamma$, the sentences $\mathit{NuprlSen}(I)(\phi)$ are called *proof obligations*. As explained in the beginning, the activity of setting up a theory morphism and verifying the proof obligations characterizes the second stage of the HOL/Nuprl connection which requires user interaction in general. Subsequently, we interpret the translation of the logical theory of HOL. In fact, we will see that this can be done, thanks to the classical axiom scheme for `inhabited`, in a purely definitional way.

[9] For instance, the fact that a new datatype is introduced in HOL by a conservative theory extensions that characterizes the datatype only up to isomorphism gives us some freedom for its interpretation in Nuprl.

7.1 Interpreting the Logical Theory

The logical HOL theory `bool` is a theory like every other theory, and does not need any particular consideration. It is however important to notice that `bool` is not a pure definitional theory in HOL. Still, it does *not* require any external meta-logical treatment, since all its axioms can be be verified in a rigorous formal way *inside* the classical Nuprl system. We follow [15] where the proof obligations have been verified using the interpretation given next.

Each HOL constant c will be interpreted by a Nuprl constant $[\![c]\!]$. For the logical constants we assume the following definitional axioms: **(1)** $[\![\neg]\!] = \lambda\ x\ .\ \texttt{if}\ x\ \texttt{then ff else tt}$, **(2)** $[\![\wedge]\!] = \lambda\ x, y\ .\ \texttt{if}\ x\ \texttt{then}\ y\ \texttt{else ff}$, **(3)** $[\![\vee]\!] = \lambda\ x, y\ .\ \texttt{if}\ x\ \texttt{then tt else}\ y$, **(4)** $[\![\forall]\!] = \lambda\ T\ .\ \lambda\ P\ .\ \downarrow_b(\forall x : T\ .\ \uparrow P)$, **(5)** $[\![\exists]\!] = \lambda\ T\ .\ \lambda\ P\ .\ \downarrow_b(\exists x : T\ .\ \uparrow P)$.

As an abbreviation we use $\texttt{arb}(T) = \texttt{decide}(\texttt{inhabited}(T), x.x, x.\bullet)$ which picks an arbitrary element inhabiting a nonempty type T. Hilbert's choice operator $([\![\epsilon]\!]\ T\ P)$, where P is a boolean predicate on M, picks an element of the subset of T specified by P if this subset is nonempty, or yields an arbitrary element of T otherwise. This motivates adding a definitional axiom $[\![\epsilon]\!] = \lambda\ T\ .\ \lambda\ P\ .\ \texttt{decide}(\texttt{inhabited}(\{x : T \mid (P\ x)\}), x.x, x.\texttt{arb}(T))$.

Finally, the HOL datatype `ind` is interpreted by $[\![\texttt{ind}]\!] = \mathbb{N}$, where $\mathbb{N}$ is the Nuprl datatype of natural numbers.

Using the interpretation $[\![\_]\!]$ all the proof obligations, i.e., the translated axioms of the HOL theory `bool` can be discharged and only the definitional Nuprl axioms given above will remain. The fact that HOL types are translated to nonempty Nuprl types is essential to verify the proof obligation corresponding to the declaration of ϵ.

8 Practical Work

8.1 An Executable Formal Specification in Maude

To obtain a formal executable specification of the translation between theories descibed before we employed the specification language Maude [5]. For this purpose we have defined a more refined version of this translation which is very close to Howe's original translator [15] in *membership equational logic* [4], the equational sublogic of *rewriting logic* which is implemented in Maude. In contrast to [15] the translator is specified in a purely equational way as a single function that maps axiomatic HOL theories into axiomatic Nuprl theories, which we think is a more suitable basis for formal meta-logical analysis. The specification is more refined than the abstract presentation in this paper, since it allows renaming, it can cope with hierarchically structured HOL theories, and it has to deal with several technicalities concerning the interfaces to HOL and Nuprl. Last but not least the specification can be executed using the Maude rewriting engine with reasonable efficiency, and has indeed been used to translate the core library of HOL into Nuprl.

The meta-theory of general logics together with an executable (and reflective) meta-logical framework [3] such as rewriting logic and its implementation in Maude can be seen as a general methodology to specify logics and logic translators in a formally rigorous way. The formal specification of the HOL/Nuprl connection nicely demonstrates the use of Maude as an executable meta-logic (cf. [6]) which also supports practical necessities like parsing external languages (HOL theories in this case) in a very uniform framework. Another application of the general methodology in the context of type theory is studied in [19] where rewriting logic is employed as an executable logical framework.

In the specification of the translator we employ membership equational logic not as a logical framework but as a meta-logic, since we do not represent all aspects of the object logics but specify a map between theories of these logics. An important application of the fact that the the meta-logic is executable is that the specification can be used as an actual translator. Another important application of a meta-logic that supports symbolic execution is formal theorem proving with partial automation. Ideally formal theorem proving and efficient execution should be unified in a single framework and recent results [3, 19] indicate that a reflective framework such as Maude is well-suited for such purposes.

8.2 Towards a Proof Translator

The translation descibed in this paper is a translation between theories, i.e., we translate a HOL sentence A over Σ into a Nuprl sentence $\alpha(A)$ over $\Phi(\Sigma)$. The proof-theoretic correctness result ensures that if A is provable from Γ in HOL then $\alpha(A)$ is provable from $\Psi(\Sigma) \cup \alpha_\Sigma(\Gamma)$ in Nuprl. The proof of the correctness theorem is carried out by constructing a Nuprl proof of $\alpha(A)$ for each HOL proof of A. This suggests extending of the translation between theories to a translation between proof calculi [16], so that this proof translation is a computable function between the datatypes of proofs. Indeed, we have found that this idea is feasible in practice as demonstrated by the implementation of an HOL/Nuprl proof translator [17] that will be briefly described in this section.

The proof translator works together with HOL90.10 and has been implemented in ML on top of the Nuprl system. This facilitates the translation of HOL proofs into Nuprl proofs considerably, since the powerful collection of Nuprl tactics can be employed. Most of the inference rules of the HOL logic can be handled by a simple combination of Nuprl tactics and the theorems given earlier in a very direct way. Routine proof obligations such as well-typedness conditions are discharged automatically by Nuprl.

In spite of the clear theoretical picture from which we started we had to overcome some difficulties in the implementation that are worth mentioning:

1. The first difficulty is of a technical nature, namely, that the HOL system is designed in such a way that proof objects are not explicitly present. Consequently, it was necessary to equip the system with an an explict notion of proof object that can be extracted from each theorem.[10]

2. Another problem is caused by the somewhat unsatisfactory situation that the HOL system does not only use the proof rules of the HOL logic as presented in the HOL documentation, but it uses in addition a considerable number of derived rules that are not reduced to basic proof rules. We guess that efficiency considerations are the reason for this "optimization", but for us this means that either:

   (a) the derived rules have to be eliminated *inside* HOL by reducing them to basic inference rules, or

   (b) that the derived rules must be given the same status as basic inference rules, which increases the number of rules that have to be treated *outside* HOL by the proof translator.

   In the long term we plan to realize alternative (b), since it turned out that, as in the case of the basic inference rules, most of the derived rules mentioned above can be implemented in Nuprl with little effort. At present, we support nearly all basic proof rules and a number of derived rules.

3. Finally, the proof translator can also deal with another particularity: The HOL implementation of the MP rule also applies to a premise $\neg M$ presupposing that $\neg M$ is equal to $M \Rightarrow \mathtt{F}$. In other words, the definitional axiom of the logical theory is built into the inference rules.

So far the proof translator has been applied to a number of simple HOL theorems of logical nature, namely, to all theorems contained in `taut_thms.sml`. Altough these theorems are quite simple, this is definitely not true for the extracted HOL proofs. The translation of the proof of `DE_MORGAN_THM`, which was automatically generated by HOL, revealed that the proof objects generated by HOL can be surprisingly large and confusing. In this case the HOL proof contained 4733 HOL inference rules. We conjecture that this problem occurs only if proofs are generated by certain automatic brute-force tactics, and that it can be circumvented by optimizing these tactics in HOL or by proving such theorems using more efficient tactics in Nuprl. Nevertheless these proofs were good test cases for the capabilities of the translator and also for the capabilities of Nuprl to handle large proof objects. For a detailed description of the proof translator and further experiments we refer to [17]. This reference also compares our approach with the implementation of an HOL/Coq proof translator presented in [9].

[10] Technically this has be done on top of a proof recording mechanism developed in [20] for the purpose of proof validation.

9 Concluding Remarks

We have complemented Howe's semantics-based justification of the HOL/Nuprl connection with a proof-theoretic counterpart. Our result showing that the translation can be seen as a map between the entailment systems of HOL and Nuprl does not only provide a simple proof-theoretic justification for *translating theories*, but it does in addition suggest *proof translation* as an interesting application that has not been considered in the context of the HOL/Nuprl connection so far. To explore the feasibility of this approach we have formally specified a theory translator in Maude and implemented a proof translator on top of Nuprl. Our experiments with a number of simple theorems indicate that proof translation is practical even though the proof objects can be rather complex. As a consequence we think that scaling up the approach to complex libraries of formal mathematics is a realistic goal, provided that proof objects are not flat proofs but *structured proofs* that reflect the modularity of the development in terms of intermediate lemmas.

Acknowledgements We would like to thank Robert Constable and Stuart Allen for several discussions on the particularities of Nuprl and the issue of logic translation. We also would like to thank Doug Howe, since this project draws on his earlier work and would not have been possible without his support. He made not only the source code of the HOL/Nuprl connection available to us, but he also gave us the opportunity to discuss with him our initial ideas on a proof-theoretic correctness result.

References

[1] S. Allen. *A Non-Type-Theoretic Semantics for Type-Theoretic Language*. PhD thesis, Cornell University, September 1987.

[2] M. Barr and C. Wells. *Category Theory for Computing Science*. International Series in Computer Science. Prentice Hall, 1990.

[3] D. Basin, M. Clavel, and J. Meseguer. Reflective metalogical frameworks. In *Proc. of LFM'99: Workshop on Logical Frameworks and Meta-languages, Paris, France, September 28, 1999.*

[4] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. In M. Bidoit and M. Dauchet, editors, *TAPSOFT'97: Theory and Practice of Software Development, 7th Int. Joint Conf. CAAP/FASE, Lille, Frace, April 1997, Proceedings.*, volume 1214 of *LNCS*. Springer-Verlag, 1997.

[5] M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. Quesada. *Maude: Specification and Programming in Rewriting Logic*. SRI International, January 1999.

[6] M. Clavel, F. Duran, S. Eker, J. Meseguer, and M.-O. Stehr. Maude as a Formal Meta-Tool. In *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 1999. Proceedings, Volume II*, LNCS 1709. Springer-Verlag, 1999.

[7] R. L. Constable, S. Allen, H. Bromely, W. Cleveland, et al. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1986.

[8] D. J. Howe. Importing mathematics from HOL into Nuprl. In J. Von Wright, J. Grundy, and J. Harrison, editors, *Ninth International Conference on Theorem Proving in Higher Order Logics TPHOL*, volume 1125 of *LNCS*, pages 267–282, Turku, Finland, August 1996. Springer Verlag.

[9] E. Denney. A Prototype Proof Translator from HOL to Coq. In M. Aagaard and J. Harrison, editors, *The 13th International Conference on Theorem Proving in Higher Order Logics*, volume 1869 of *Lecture Notes in Computer Science*, pages 108–125, Portland, OR, August 2000. Springer-Verlag.

[10] J. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[11] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[12] D. J. Howe. A classical set-theoretic model of polymorphic extensional type theory. Manuscript.

[13] D. J. Howe. Semantical foundations for embedding HOL in Nuprl. In Martin Wirsing and Maurice Nivat, editors, *Algebraic Methodology and Software Technology*, volume 1101 of *LNCS*, pages 85–101, Berlin, 1996. Springer-Verlag.

[14] D. J. Howe. Toward sharing libraries of mathematics between theorem provers. In *Frontiers of Combining Systems 1998*, 1998.

[15] D. J. Howe. Source Code of HOL-Nuprl Translator including Extensions to Nuprl, January 1999.

[16] J. Meseguer. General logics. In H.-D. Ebbinghaus et al., editors, *Proceedings, Logic Colloquium, 1987*, pages 275–329. North-Holland, 1989.

[17] P. Naumov, M.-O. Stehr, and J. Meseguer. The HOL/NuPRL proof translator — A practical approach to formal interoperability. In *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs'2001, Edinburgh, Scottland, UK , September 3–6, 2001, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 2001. To appear. See also `http://www.csl.sri.com/~stehr/fi_eng.html`.

[18] K. Petersson, J. Smith, and B. Nordstroem. *Programming in Martin-Löf's Type Theory. An Introduction.* International Series of Monographs on Computer Science. Oxford: Clarendon Press, 1990.

[19] M.-O. Stehr and J. Meseguer. Pure type systems in rewriting logic. In *Proc. of LFM'99: Workshop on Logical Frameworks and Meta-languages, Paris, France, September 28, 1999.*

[20] W. Wong. Validation of HOL proofs by proof checking. *Formal Methods in System Design: An International Journal*, 14(2):193–212, 1999.