First-Order Logic with Dependent Types

Florian Rabe
florian@cs.cmu.edu *

Carnegie Mellon University and International University Bremen

Abstract. We present DFOL, an extension of classical first-order logic with dependent types, i.e., as in Martin-Löf type theory, signatures may contain type-valued function symbols. A model theory for the logic is given that stays close to the established first-order model theory. The logic is presented as an institution, and the logical framework LF is used to define signatures, terms and formulas. We show that free models over Horn theories exist, which facilitates its use as an algebraic specification language, and show that the classical first-order axiomatization is complete for DFOL, too, which implies that existing first-order theorem provers can be extended. In particular, the axiomatization can be encoded in LF.

1 Introduction and Related Work

Classical first-order logic (FOL) and its variations are folklore knowledge. Lots of variations classify the elements of a model into different sorts, e.g., many-sorted or order-sorted FOL. Type theory may also be viewed as an extension of FOL, introducing function sorts. Further extensions of type theory include type operators, type polymorphism and dependent types. All these extensions have varying advantages and disadvantages and various implementations exist. Surprisingly, not much work has been undertaken to extend FOL with just dependent types.

PVS ([ORS92]) is a classical verification system that extends simply-typed higher order logic with dependent function, record and product types and other concepts. Its type system is undecidable, and a set theoretic semantics for an idealized language exists ([OS97]). Coq ([BC04]), Nuprl ([CAB+86]) and LF ([Pfe01]) implement Martin-Löf's intuitionistic type theory with dependent types ([ML74]); while the first two add further concepts and are directed at general mathematics, the main purpose of LF is the specification of logics. A semantics for Martin-Löf type theory was introduced in [Car86], where also algebraic theories are treated. It was linked with locally cartesian closed categories as analogues of FOL structures in [See84] (see also [Hof94] and [Dyb95] for related approaches). However, these concepts are mathematically very complex and not tightly connected to research on theorem proving. Neither are they easy to specialize to FOL, even if intuitionistic FOL is used.

* The author was supported by a fellowship for Ph.D. research of the German Academic Exchange Service.

We could only locate one attempt at combining FOL with just dependent types ([Mak], never published), which is mainly directed at studying equivalence of categories. It adds connectives and quantifiers to the treatment in [Car86] and gives an axiomatization, but does not allow general equality and function symbols (without which dependent types are significantly less interesting). Their type hierarchy is similar to ours, but the chosen notation is completely different from the usual one.

Our motivation in defining DFOL is to add as little as possible to FOL, keeping not only notation and intuition but also the results and applications. Thus, both researchers and implementations can use DFOL more easily. Therefore, we deliberately dispense with one feature of dependent types, namely circular dependencies, which greatly simplifies the model theory while hardly excluding interesting applications.

DFOL is presented as an institution. Institutions were introduced in [**?**] as a unifying concept for model theory. Examples for institutional definitions of logics are OBJ ([GWM$^+$93]) and Maude ([CELM96]). The syntax of DFOL is presented directly in LF (see [HST94] for other logic specifications in LF) because even in our special case the inherent complexity of dependent types makes any independent introduction inconvenient. We introduce DFOL in section 2, sections 3 to 5 treat free models, axiomatization and examples.

2 Syntax and Semantics

2.1 Preliminary Definitions

Institutions An institution is a tuple $(Sig, Sen, Mod, \models)$ where Sig is a category of signatures; signature morphisms are notation changes, usually mappings between the symbols of the signatures; $Sen : Sig \to Set$ is a functor that assigns to each signature its set of formulas and with each signature morphism the induced formula translation; $Mod : Sig \to Cat^{op}$ is a functor that assigns to every signature its category of models, and with every signature morphism the induced model reduction; and for every signature Σ, the satisfaction relation $\models_\Sigma \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$ between models and sentences determines truth. Institutions must satisfy the satisfaction condition which can be paraphrased as "Truth is invariant under change of notation.". We refer to [**?**] for a thorough introduction.

LF The logical framework LF and its implementation Twelf ([Pfe01], [PS99]) implement Martin-Löf type theory ([ML74]). LF will be used as a meta-language to define the syntax of DFOL. An LF signature[1] consists of a sequence $c : T$ of declarations, where c is a new symbol, and T is its type, which may depend on the previously declared symbols. T is of the form $\Pi\, x_1 : T_1.\ \ldots . \Pi\, x_n : T_n.\ T_{n+1}$, which means that c is a function symbol taking arguments of the types $T_1, \ldots, T_n$, called $x_1, \ldots, x_n$, and returning an argument of the type T_{n+1}; dependent types

[1] Readers familiar with LF will notice that our introduction is highly simplified.

means that x_i may occur in $T_{i+1}, \ldots, T_{n+1}$. If $T_{n+1} = \texttt{type}$, c is not a function symbol but returns a new dependent type for every argument tuple. If x does not occur in B, $\Pi\, x{:}A.\ B$ is abbreviated by $A \to B$.

To illustrate this, look at the signature Σ_B which represents our meta-language:

$\mathbf{S} : \texttt{type}. \quad \mathit{Univ} : \mathbf{S} \to \texttt{type}. \quad o : \texttt{type}.$
$\mathit{true}, \mathit{false} : o. \quad \wedge, \vee, \Rightarrow : o \to o \to o. \quad \neg : o \to o.$
$\forall, \exists : \Pi\, S{:}\mathbf{S}.\ (\mathit{Univ}\ S \to o) \to o. \quad \doteq : \Pi\, S{:}\mathbf{S}.\ \mathit{Univ}\ S \to \mathit{Univ}\ S \to o.$

In this signature, $\mathbf{S}$ is a type, the type of sorts declared in a DFOL signature. *Univ* is a dependent type family that returns a new type for each sort S, namely the type of terms of sort S; models will interpret the type $\mathit{Univ}\ S$ as the universe for the sort S. o is the type of formulas. The remainder of the signature encodes the usual grammar for FOL formulas. Higher-order abstract syntax is used, i.e., λ is used to bind the free variables in a formula, and quantifiers are operators taking a λ expression as an argument.[2] Quantifiers and the equality symbol take the sort they operate on as their first argument; we will omit this argument if no ambiguities arise. When we refer to sorts, terms or formulas, we always mean objects with the respective type that do not contain any lambda abstractions except for those preceded by quantifiers.

A context for a signature Σ is a sequence of typed variables $x : \mathit{Univ}\ S$, where previously declared variables and symbols declared in Σ may occur in S. Sorts, terms and formulas in context C may contain the variables declared in C.

We introduce abbreviations to make the LF syntax more familiar: The usual infix notation and bracket conventions of FOL are used, and conjunction binds stronger than implication; $\forall \lambda\, x : \mathit{Univ}\ S.\ F$ is abbreviated as $\forall x : S.\ F$, and we write $\forall x, y : S,\ z : S'$ instead of $\forall x : S.\ \forall y : S.\ \forall z : S'$, and similarly for $\exists$. Note that application is written as $f\ t_1\ \ldots\ t_n$ instead of the familiar $f(t_1, \ldots t_n)$ and that substitution is written as β-reduction.

We allow a harmless[3] extension of LF: Contexts and signatures need not be finite but may contain infinitely many declarations of the form $c : \mathit{Univ}\ S$. The reason for this is purely technical: It allows to have an infinite reservoir of names c for elements that occur in the universe of S, which facilitates some proofs.

2.2 Signatures

We are now ready to introduce DFOL signatures as certain LF signatures. A DFOL signature will consist of Σ_B followed by declarations of the form

$$c : \Pi\, x_1 : \mathit{Univ}\ S_1.\ \ldots\ .\Pi\, x_m : \mathit{Univ}\ S_m.\ T$$

where $m \in \mathbb{N}$, $T = \mathbf{S}$, $T = \mathit{Univ}\ S$ or $T = o$, and $S_1, \ldots, S_m, S$ are sorts. c is called a sort symbol if $T = \mathbf{S}$, a function symbol with target S if $T = \mathit{Univ}\ S$,

[2] The reflexivity of LF is used to encode the sort dependencies. Alternatively, $\mathbf{S}$ could be replaced with `type` and *Univ* omitted everywhere. But then sorts could not be used as arguments since LF does not support polymorphism.

[3] It is not harmless in general, only in our setting as explained below.

and a predicate symbol if $T = o$. The sorts S_i are called arguments of c. If we only allow sort symbols without arguments, we obtain the usual many-sorted FOL. Sort symbols with arguments construct dependent sorts.

Definition 1 (Signatures). *Let Σ_i be partial LF signatures such that $\Sigma = \Sigma_B\ \Sigma_0\ \ldots\ \Sigma_d$ is an LF signature, and let Σ^n abbreviate $\Sigma_B\ \Sigma_0\ \ldots\ \Sigma_n$ for $n \leq d$. Σ is called a DFOL signature if*

1. *only sort, function or predicate symbols are declared in $\Sigma_0\ \ldots\ \Sigma_d$,*
2. *all sort symbol declarations in Σ_n have only arguments from Σ^{n-1},*
3. *the target of a function symbol declaration in Σ_n is not an LF term over Σ^{n-1} (i.e., only over Σ_n).*

Condition 1 prevents the use of more expressive LF declarations[4]. Condition 2 establishes an acyclic sort dependency: Every sort declared in Σ_n may only depend on sorts that have been declared in Σ^{n-1}. d is called the depth of Σ. A sort, term or formula has depth n if it is defined over Σ^n (for some context) but not over Σ^{n-1}. Condition 3 is the most important one: It requires that a function symbol that takes a sort of depth n as an argument may not return an element of a smaller depth.[5]

Condition 3 is rather restrictive to ensure the existence of free models. It excludes, e.g., projection functions π from a sort $T\ n$ of n-tuples (at depth 1) over B to B (at depth 0). A weaker restriction that is still sufficient for free models could allow π if there are equality axioms that identify every term $\pi\ x$ with a term of depth 0, i.e., if π does not generate new objects.

We have the following property.

Lemma 1. *Let Σ be a DFOL signature of depth d. Then Σ^n, for $0 \leq n \leq d$, is a DFOL signature such that the sorts of Σ^n are precisely the sorts of Σ that have depth at most n; and such that the terms of a sort S of Σ^n are precisely the terms of sort S over Σ.*

Proof. Trivial but note how condition 3 is needed.

This ensures that infinitely many declarations of the form $c : \mathit{Univ}\ S$ are indeed harmless: S may depend on the previously declared symbols but these must have strictly smaller depth than S and so on; therefore, S can only depend on finitely many symbols. From now on let the word signature only refer to DFOL signatures.

In general every sort symbol in Σ_n has a type of the form

$$\Pi\, x_1 : \mathit{Univ}\ S_1.\ \ldots\ .\Pi\, x_m : \mathit{Univ}\ S_m.\ \mathbf{S} \qquad \text{(F 1)}$$

[4] In particular, function types may not occur as arguments.

[5] In the terminology of [Vir96], this corresponds to dependence relations between sort symbols given by $s \prec t$ iff s has at most the depth of t.

where $S_1, \ldots, S_m$ have depth smaller than n. Without loss of generality, we can assume that every function symbol in Σ_n has a type of the form

$$\Pi\, x_1 : \mathit{Univ}\ S_1.\ \ldots\ .\Pi\, x_r : \mathit{Univ}\ S_r.\ \mathit{Univ}\ S_{r+1} \to \ldots \to \mathit{Univ}\ S_m \to \mathit{Univ}\ S \tag{F 2}$$

where $S_1, \ldots, S_r$ have depth smaller than n and $S_{r+1}, \ldots, S_m$ have depth n. Similarly, we can assume that every predicate symbol in Σ_n has a type of the form

$$\Pi\, x_1 : \mathit{Univ}\ S_1.\ \ldots\ .\Pi\, x_r : \mathit{Univ}\ S_r.\ \mathit{Univ}\ S_{r+1} \to \ldots \to \mathit{Univ}\ S_m \to o. \tag{F 3}$$

A signature morphism $\sigma : \Sigma \to \Sigma'$ is a mapping of Σ symbols to Σ' symbols such that (i) σ is the identity for symbols declared in Σ_B; (ii) σ respects types and depths of all mapped symbols. We omit the formal definition. For example, the identity mapping is a signature morphism from $\Sigma_B\ \Sigma_0\ \ldots\ \Sigma_c$ to $\Sigma\ =\ \Sigma_B\ \Sigma_0\ \ldots\ \Sigma_d$ for $c \leq d$. These morphisms are called extensions.

Running Example We will use a running example. Consider the signature Σ

$t : \mathbf{S}.\quad i : \mathit{Univ}\ t.\quad j : \mathit{Univ}\ t.\quad d : \mathit{Univ}\ t \to \mathbf{S}.$
$f : \Pi\, x : \mathit{Univ}\ t.\ \mathit{Univ}\ d\ x.\quad c : \mathit{Univ}\ d\ i.\quad p : \Pi\, x : \mathit{Univ}\ t.\ \mathit{Univ}\ d\ x \to o$

Σ has depth 1 (Σ_0 consists of the declarations for t, i and j.) and declares three sorts, t, $d\ i$ and $d\ j$, and five terms, i and j with sort t, $f\ i$ and c with sort $d\ i$, and $f\ j$ with sort $d\ j$, and one predicate.

2.3 Sentences and Models

Definition 2 (Sentences). *$Sen(\Sigma)$, the set of formulas over Σ, is the set of LF objects $F = \lambda\, x_1 : \mathit{Univ}\ S_1.\ \ldots\ldots \lambda\, x_n : \mathit{Univ}\ S_n.\ G$ where G is of type o and all λ's in G are preceded by $\forall$ or $\exists$. For a signature morphism σ, $Sen(\sigma)$ is the mapping between formulas induced by σ.*

Taking the above lambda closure over the free variables in G is not needed in FOL. In DFOL, however, it is helpful to keep track of the sorts of the free variables in G because x_i may occur in $S_{i+1}, \ldots, S_n$. For simplicity, we identify G and F if it does not cause confusion. We do not distinguish between, e.g., F and $\lambda\, x : \mathit{Univ}\ S.\ F$ if x does not occur in F. Closed and atomic formulas are defined in the obvious way.

Running Example Examples for closed formulas are $E_1 = \forall x : t.\ (i \doteq x \ \Rightarrow\ p\ x\ f\ x)$ and $E_2 = \forall x : t.\ \exists y : d\ x.\ (p\ x\ y)$.

Definition 3 (Models, Assignments). *Let Σ have depth d. We define Σ-models and assignments by induction on d. If $d = 0$, $Mod(\Sigma)$ is the category of many-sorted FOL models of Σ, i.e., interpretation functions M given by*

- *a universe s^M for every sort symbol $s : \mathbf{S} \in \Sigma$,*

– *a function* $f^M : s_1^M \times \ldots \times s_m^M \to s^M$ *for every function symbol* f *with* m *arguments,*
– *a mapping* $p^M : s_1^M \times \ldots \times s_m^M \to \{0,1\}$ *for every predicate symbol* p *with* m *arguments.*

If $d > 0$ *and* Mod *is defined for signatures of depth* $d-1$*, the objects* $M \in Mod(\Sigma)$ *are interpretation functions* $\cdot^M$ *that interpret the sort, function and predicate symbols of* Σ *in the following way.*

1. $\cdot^M$ *restricted to* Σ^{d-1} *(which has depth* $d-1$*) is a* Σ^{d-1}*-model. We denote this restriction by* N*.*
2. *Let* $C = (x_i : \ Univ\ S_i)_{i=1,\ldots,n}$ *be a context over* Σ^{d-1}*. We define assignments and model extensions through two entwined recursions:*
   – *An assignment from* C *into* N *is a mapping* u *that assigns to every* $x_i : Univ\ S_i$ *in* C *an element* $u(x_i) \in S_i^N(u)$*. Here,* $S_i^N(u)$ *is the extension of* $\cdot^N$ *to sorts induced by the assignment* u*.*
   – *The extension of* $\cdot^N$ *to sorts and terms for an assignment* u *from the respective free variables into* N*, is defined recursively by* $x_i^N(u) = u(x_i)$ *and*
   $$(c\ t_1\ \ldots\ t_m)^N(u) = c^N(t_1^N(u), \ldots, t_m^N(u))$$
   for a sort or function symbol c *and terms* t_i*.*
3. *For every sort symbol of the form (F 1) declared in* Σ_d *and every assignment* u *from* $(x_i : \ Univ\ S_i)_{i=1,\ldots,m}$ *into* N*,*
   $$s^M(u(x_1), \ldots, u(x_m)) \text{ is a set.}$$
4. *Note that at this point,* $\cdot^M$ *is defined for all sort symbols in* Σ_d *and all terms of depth at most* $d-1$*. As in step 2, we define assignments into* M *and the extension* $\cdot^M$ *to sorts of depth* d*.*
5. *For every function symbol* f *of the form (F 2) declared in* Σ_d *and every assignment* u *from* $(x_i : \ Univ\ S_i)_{i=1,\ldots,m}$ *into* M*,*
   $$f^M(u(x_1), \ldots, u(x_m)) \in S^M.$$
6. *As in step 2, we define* $\cdot^M$ *for all terms of depth* n*.*
7. *For every predicate symbol* p *of the form (F 3) declared in* Σ_d *and every assignment* u *from* $(x_i : \ Univ\ S_i)_{i=1,\ldots,m}$ *into* M*,*
   $$p^M(u(x_1), \ldots, u(x_m)) \in \{0,1\}.$$

For a sort S, S^M is called the universe of M. A model morphism $\varphi : M \to N$ for Σ-models M and N maps from each universe of M to some universe of N as follows.[6] For every assignment u from $(x_i : \ Univ\ S_i)_{i=1,\ldots,m}$ into M, we put $\boldsymbol{u} = (u(x_1), \ldots, u(x_m))$; then $\varphi(\boldsymbol{u}) = (\varphi(u(x_1)), \ldots, \varphi(u(x_m))$ is an assignment into N. We require that for every d and every appropriate assignment u into M

[6] Since the universes of M are not required to be pairwise disjoint, φ should be indexed with the universes to distinguish these mappings. We omit these indexes and rely on the context.

1. for every sort symbol s declared in Σ_d, φ is a mapping from $s^M(\boldsymbol{u})$ to $s^N(\varphi(\boldsymbol{u}))$,
2. for every function symbol f declared in Σ_d, $\varphi(f^M(\boldsymbol{u})) = f^N(\varphi(\boldsymbol{u}))$,
3. for every predicate symbol p declared in Σ_d, $p^M(\boldsymbol{u}) \leq p^N(\varphi(\boldsymbol{u}))$. [7]

Note that for $d = 0$, this reduces to the usual first-order definition of homomorphisms.

For a signature morphism $\sigma : \Sigma \to \Sigma'$, $Mod(\sigma)$ is the usual model reduction functor from $Mod(\Sigma')$ to $Mod(\Sigma)$, i.e., $Mod(\sigma)$ maps a Σ'-model N to a Σ-model M defined by $c^M = \sigma(c)^N$ for every symbol c of Σ (Thus every universe of M is also a universe of N.); and $Mod(\sigma)$ maps a Σ'-model morphism from N to N' to its restriction to the universes of $Mod(\sigma)(N)$. We omit the formal proof that Mod is indeed a functor.

In particular for an extension $\varphi : \Sigma^n \to \Sigma^d$ for $n \leq d$, $Mod(\varphi)$ maps every Σ^d-model M to its restriction M^n over Σ^n.

Running Example A model M for the example signature is given by $t^M = \mathbb{N}\backslash\{0\}$, $i^M = 2$, $j^M = 3$, $d^M(n) = \mathbb{N}^n$, $f(n) = (1, \ldots, n)$, and $p(n, (m_1, \ldots, m_n)) = 1 \Leftrightarrow m_1 = n$. Note how all interpretations must be defined for all elements regardless of whether they can be named by terms.

2.4 Satisfaction

Satisfaction $M \models_\Sigma F$ is defined in the usual way: $M \models_\Sigma F \Leftrightarrow F^M(u) = 1$ for all assignments u from the free variables in F into M, where $\cdot^M$ is extended to all formulas by

- if $F = p\ t_1\ \ldots t_m$ for a predicate symbol p, then $F^M(u) = p^M(t_1^M(u), \ldots, t_m^M(u))$,
- if $F = t_1 \doteq t_2$, then $F^M(u) = 1 \Leftrightarrow t_1^M(u) = t_2^M(u)$,
- the usual definition for propositional connectives,
- if $F = \forall x : S.\ G$, then $F^M(u) = \inf\{G^M(u\{x \mapsto v\}) \mid v \in S^M(u)\}$,
- if $F = \exists x : S.\ G$, then $F^M(u) = \sup\{G^M(u\{x \mapsto v\}) \mid v \in S^M(u)\}$,

where $u\{x \mapsto v\}$ is as u but with $u(x) = v$. We omit the formal proof of the satisfaction condition.

Running Example We have $M \not\models_\Sigma E_1$ since for $u(x) = 2$, we have $(i \doteq x)^M(u) = 1$ but $(f\ x)^M(u) = (1, 2)$ and therefore, $(p\ x\ f\ x)^M(u) = 0$. And we have $M \models_\Sigma E_2$ since for every $n \in t^M$ there is an $m \in d^M(n)$ such that $p^M(n, m) = 1$, for example $m = (n, 1, \ldots, 1)$.

[7] Using $\leq$ means that truth of atomic formulas is preserved along homomorphisms.

3 Free Models

A theory $K = (\Sigma, T)$ consists of a signature Σ and a set T of closed Σ-formulas. Sen and Mod are extended to theories by putting $Sen(\Sigma, T) = Sen(\Sigma)$ and $Mod(\Sigma, T) = \{M \in Mod(\Sigma) \mid M \models F \text{ for all } F \in T\}$.

For many-sorted first-order logic there is the standard result (see for example [**?**]) that for a Horn theory (Σ, T) and sets A_s of generators for all sort symbols s, there is a Σ-model $Free_\Sigma(A)/T$ of T such that for every $M \in Mod(\Sigma, T)$ and every family of mappings $\varphi_s : A_s \to s^M$ there is a unique Σ-morphism $\overline{\varphi} : Free_\Sigma(A)/T \to M$ that extends φ.

The purpose of this section is to establish the corresponding result for DFOL. Horn formulas over a DFOL signature Σ are defined in the usual way (i.e., a universally closed implication $T \Rightarrow H$ in which the tail T is a conjunction of atomic formulas and the head H is an atomic formula except for $false$). A Horn formula is hierarchic if the depth of its head is at least as big as the depth of its tail. We can allow only hierarchic Horn formulas because, informally, otherwise axioms of a greater depth could influence the interpretation of symbols of a lower depth. As generators, we might use a family A_S where S runs over all sorts, but a stronger result is possible that also allows generators for sorts that depend on other generators. The most elegant way to describe such generators is by using signatures that contain arbitrarily many constant declarations of the form $a :$ *Univ* S.[8] Then DFOL has free models over hierarchic Horn theories in the following sense.

Lemma 2. *For a hierarchic Horn theory $K = (\Sigma, T)$, there is a K-model $Free_\Sigma/T$ such that for every K-model M there is a unique Σ-morphism from $Free_\Sigma/T$ to M.*

Proof. Let Σ and T be as stated. Let T^n be the restriction of T to depth at most n. We define $Free_\Sigma/T$ by induction on the depth of Σ, say d. If $d = 0$, $F^0 = Free_{\Sigma^0}/T^0$ is the classical result; note that the universes of F^0 arise by taking equivalence classes of terms.

By the induction hypothesis, we assume

$$F^{d-1} = Free_{\Sigma^{d-1}}/T^{d-1} \in Mod(\Sigma^{d-1}, T^{d-1})$$

all universes of which consist of equivalence classes of terms, let $[\cdot]$ denote these equivalence classes. We define Herbrand models to be (Σ^d, T^d)-models M that satisfy the following conditions:

- M agrees with F^{d-1} for all symbols in Σ^{d-1},
- for all sort symbols s of the form (F 1) in Σ_d: $s^M([t_1], \ldots, [t_m]) = U/\equiv$ where U contains those terms that have any of the sorts $s\ b_1\ \ldots\ b_m$ for $b_i \in [t_i]$, and $\equiv$ is some equivalence relation[9]; let $\langle\cdot\rangle$ denote the equivalence classes of $\equiv$; clearly, all universes are disjoint, so that we can use the symbols $\equiv$ and $\langle\cdot\rangle$ for all universes,

[8] This is why we allow infinite signatures.

– for a function symbol f of the form (F 2) in Σ_d:

$$f^M([t_1], \ldots, [t_r], \langle t_{r+1}\rangle, \ldots, \langle t_m\rangle) = \langle f\ t_1\ \ldots\ t_m\rangle,$$

– for a predicate symbol p of the form (F 3) in Σ_d:

$$p^M([t_1], \ldots, [t_r], \langle t_{r+1}\rangle, \ldots, \langle t_m\rangle) = 1 \Leftrightarrow M \models_{\Sigma^d} p\ t_1\ \ldots\ t_m.$$

We put F^d to be the Herbrand model that satisfies the least atomic formulas. By definition, it is a (Σ^d, T^d)-model in which all universes arise by taking equivalence classes of terms.

We have to prove that the above F^d exists. This is done in essentially the same way as for the FOL case. Informally, a Herbrand model is uniquely determined by the equivalence $\equiv$ and the set P of atomic formulas of depth d that it satisfies. Let $(\equiv_i, P_i)_{i\in I}$ be all Herbrand models. This family is not empty: It contains the model in which all atomic formulas of depth d are true. Then it is easy to show that this family also contains the model determined by $\bigcap_{i\in I} \equiv_i$ and $\bigcap_{i\in I} P_i$, which we can put to be F^d. This completes the induction.

The unique morphism into M simply maps the equivalence class of a term t to t^M. This is well-defined because M satisfies T and by the definition of F^d.

Running Example The free model F for the example signature with the axioms $\forall x : t.\ (p\ x\ f\ x)$ and $c \doteq f\ i$ is given by $t^F = \{\{i\}, \{j\}\}$, $d^F(\{i\}) = \{\{c, f\ i\}\}$, $d^F(\{j\}) = \{\{f\ j\}\}$ and $(p\ x\ y)^F(u) = 1$ for both possible assignments u.

Generators Since we allow infinite signatures, Lem. 2 contains the result where there are arbitrarily many generators. We make this more precise: Let $\Sigma(A)$ be the signature Σ enriched with the declarations from a context A over Σ. Then every Σ-model M and every assignment u from A into M induce a $\Sigma(A)$-model, which we call (M, u).

Theorem 1. *For a signature Σ, a context A over Σ, and a set T of hierarchic Horn formulas over $\Sigma(A)$, there is a $\Sigma(A)$-model $Free_\Sigma(A)/T$ of T such that for every Σ-model M and every assignment u from A into M such that (M, u) models T, there is a unique $\Sigma(A)$-morphism $\overline{u} : Free_\Sigma(A)/T \to (M, u)$ that extends u.*

Proof. Simply put $Free_\Sigma(A)/T$ to be $Free_{\Sigma(A)}/T$.

Of particular interest is the case where T does not depend on A. Then $Free_\Sigma(A)/T$ is a (Σ, T)-model and every assignment u from A into a (Σ, T)-model M has a unique extension to a Σ-morphism, namely the extension of the interpretation function M under the assignment u.

We abstain from a categorical interpretation in terms of adjoint functors and simply remark that $Free_\Sigma(\varnothing)/T$ is initial in the category $Mod(\Sigma, T)$.

[9] Note that $\equiv$ identifies more terms than the relation $(t \doteq t')^M = 1$: Term identification in F^{d-1} may lead to sort identification at depth d, thus causing terms of different sorts to become equal.

Running Example The free model F for the example signature with the generators $A = a_1 : t,\ a_2 : f\ a_1$ with the axioms E_1 (from the running example above) and $i \doteq a_1$ is given by: $t^F = \{\{i, a_1\}, \{j\}\}$, $d^F(\{i, a_1\}) = \{\{f\ i, f\ a_1\}, \{c\}, \{a_2\}\}$, $d^F(\{j\}) = \{\{f\ j\}\}$ and $(p\ x\ y)^F(u) = 1$ precisely for $u(x) = \{i, a_1\}$, $u(y) = \{f\ i, f\ a_1\}$.

4 Axiomatization

$$\frac{}{\vdash A}\ \text{for every } A \in T \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$$

$$\frac{}{\Gamma \vdash \mathit{true}, \Delta} \qquad \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

$$\frac{\Gamma, A \vdash \Delta}{\Gamma, \exists x : S.\ A \vdash \Delta} \qquad \frac{\Gamma \vdash (\lambda x{:}S.\ A)\ t, \Delta}{\Gamma \vdash \exists x : S.\ A, \Delta}$$

$$\frac{}{\begin{array}{c}\vdash \forall x_1 : S_1, \ldots\ x_r : S_r,\ x_{r+1}, x'_{r+1} : S_{r+1}, \ldots,\ x_m, x'_m : S_m. \\ x_{r+1} \doteq x'_{r+1} \wedge \ldots \wedge x_m \doteq x'_m \Rightarrow \\ f\ x_1\ \ldots\ x_r\ x_{r+1}\ \ldots\ x_m \doteq f\ x_1\ \ldots\ x_r\ x'_{r+1}\ \ldots\ x'_m \\ \text{for all } f \text{ of the form (F 2)}\end{array}}$$

In the $\exists$left rule, we assume that x does not occur free in Γ or Δ. In the $\exists$right rule, t is a term of sort S in which the free variables from Γ, S, A and Δ may occur. We omit the structural rules (axioms, cut, weakening, contraction and exchange), the remaining equality rules (reflexivity, symmetry and transitivity) and the rules for definable connectives and quantifiers.

Fig. 1. Axiomatization for $SC(\Sigma, T)$

Completeness To enhance readability, we omit the operator *Univ* completely from now on. We show that the classical axiomatizations are sufficient for DFOL. We use the common Gentzen style notation for sequents and rules. For simplicity, we only give the completeness result for the case that empty universes are forbidden. The general case is similar.

Theorem 2. *Let $K = (\Sigma, T)$ be a finite theory (i.e., Σ and T are finite), and let the rules of $SC(K)$ be as in classical sequent style axiomatizations for FOL with equality (see [Gal86]) with slight modifications as given in Fig. 1. Then $SC(K)$ is sound and complete for $Mod(K)$.*

The modifications mainly serve to account for free variables. Only the congruence axiom has an unfamiliar form and may even look incomplete, but note that the putatively underivable formulas are not well-typed in the first place.

Proof. We only sketch the completeness proof since it is almost the same as the classical Henkin style proof (see [Hen49]). There are only a few minor technical differences in the notation of free variables and quantifiers. Firstly, a set of formulas Γ such that $\Gamma \cup T$ is consistent is extended to a maximal consistent set $\overline{\Gamma}$ with witnesses. To do that, we use a context A containing infinitely many declarations for each sort over Σ and A.

Then $M = Free_\Sigma(A)/T$, where T is the set of atomic formulas in $\overline{\Gamma}$, is a $(\Sigma(A), T)$-model of $\overline{\Gamma}$. The proof proceeds by induction on the number of occurrences of logical symbols in $F \in \overline{\Gamma}$.

Then the $\Sigma(A)$-model M yields a Σ-model M' by forgetting the interpretations of the symbols from A (formally $M' = Mod(\sigma)(M)$ where $\sigma : \Sigma \to \Sigma(A)$ is the injection). M' is a model of $\Gamma \cup T$ because no symbols from A occur in $\Gamma \cup T$. From this model existence result, the theorem follows as in the classical case.

As in the classical case, Thm. 4 yields the compactness theorem and the Löwenheim-Skolem result that any consistent set of sentences has a countable model.

Running Example Let an axiom for Σ be given by $i \doteq j$. This implies that the sorts $d\ i$ and $d\ j$ are equal, it also implies $(f\ i)^M = (f\ j)^M$ in every model M. Note, however, that $f\ i \doteq f\ j$ cannot be derived because it is not a well-formed formula. It cannot be well-formed because it only makes sense in the presence of the axiom $i \doteq j$. If equations between terms of the same depth but different sorts were allowed, and if equations in this broader sense were forbidden to occur in the axioms of a theory, the completeness result would still hold.

Implementation It follows that a Gentzen style theorem prover of FOL can be turned into one for DFOL, if its syntax is extended to support DFOL signatures. In the case of the existing implementation of FOL in LF, which is part of the Twelf distribution, only rules for equality need to be added.

Completeness results for resolution based proving as in Vampire ([RV02]) require additional work. The transformation into conjunctive normal form is as for FOL with the exception of skolemization, which transforms

$$\lambda\, x_1 : S_1.\ \ldots . \lambda\, x_n : S_n.\ \exists x : S.\ G \text{ to } \lambda\, x_1 : S_1.\ \ldots . \lambda\, x_n : S_n.\ (\lambda\, x : S.\ G)(f x_1 \ldots x_n).$$

Here $x_1, \ldots, x_n$ also contain the free variables of S, and the substitution must also operate on S. If paramodulation (see for example [NR01]) is used to replace a subterm s of F with t, both the sorts of s and t as well as s and t themselves must be unified (see [PP03] for unification with dependent types in LF).

5 Examples

In the examples, we use infix notation for some symbols without giving the corresponding Twelf declarations. And we use the implicit arguments notation of Twelf, i.e., if a free variable X which, due to the context, must have type T occurs in a declaration, it is assumed that the type is prefixed with $\Pi\, X : T$. If such a free variable occurs in an axiom, we assume an implicit universal quantification.

Categories Let Cat be the following theory of depth 1 (where we leave a blank line between signature and axioms)

$Ob:\ \mathbf{S}.$
$Mor:\ \Pi\, A, B{:}Ob.\ \mathbf{S}.$
$id:\ \Pi\, A{:}Ob.\ Mor\ A\ A.$
$\circ:\ Mor\ A\ B\ \to\ Mor\ B\ C\ \to\ Mor\ A\ C.$

$\forall f : Mor\ A\ B.\ f \doteq f \circ id\ B$
$\forall f : Mor\ A\ B.\ f \doteq id\ A \circ f$
$\forall f : Mor\ A\ B,\ g : Mor\ B\ C,\ h : Mor\ C\ D.\ (f \circ g) \circ h \doteq f \circ (g \circ h)$

Then $Mod(Cat)$ is the category of small categories. For example the formula $\lambda X{:}Ob.\ \forall A : Ob.\ \exists f : Mor\ A\ X.\ \forall g : Mor\ A\ X.\ f \doteq g$ expresses the property that $X : Ob$ is a terminal element.

The free category over a generating graph G can be obtained by applying Thm. 3 to Cat where A contains declarations $N : Ob$ for every node N and $E : Mor\ N\ N'$ for every edge $E = (N, N')$ of G. Note that the theorem also allows to impose specific equalities, e.g., an axiom $E = E_1 \circ E_2$ if the composition of E_1 and E_2 is already part of G.

A theory extension consists of additional declarations and additional axioms; it is simple, if it does not add sort declarations. Let $2Cat$ be the following extension of Cat

$2cell:\ \Pi\, f, g{:}Mor\ A\ B.\ \mathbf{S}.$
$Id_2:\ \Pi\, f{:}Mor\ A\ B.\ 2cell\ f\ f.$
$\circ_{vert}:\ \Pi\, f, g, h{:}Mor\ A\ B.\ 2cell\ f\ g\ \to\ 2cell\ g\ h\ \to\ 2cell\ f\ h.$
$\circ_{hor}:\ \Pi\, f, g{:}Mor\ A\ B.\ \Pi\, f', g'{:}Mor\ B\ C.$
$\quad 2cell\ f\ g\ \to\ 2cell\ f'\ g'\ \to\ 2cell\ f \circ f'\ g \circ g'.$

If we also add appropriate axioms, $Mod(2Cat)$ becomes the category of small 2-categories. Bi-categories can be specified similarly, e.g., using the axiom
$\forall f : Mor\ A\ B,\ g : Mor\ B\ C\ ,\ h : Mor\ C\ D,\ k, l : Mor\ A\ D.$
$\quad k \doteq (f \circ g) \circ h\ \wedge\ l \doteq f \circ (g \circ h)\ \Rightarrow\ \exists \alpha : 2cell\ k\ l.\ \exists \beta : 2cell\ l\ k.$
$\qquad \alpha \circ_{vert} \beta \doteq Id_2\ k\ \wedge\ \beta \circ_{vert} \alpha \doteq Id_2\ l.$

Under the Curry-Howard-Tait correspondence, a 2-category corresponds to a logic with formulas, proofs and rewrites. If a simple extension of $2Cat$ declares function symbols for connectives, proof rules and conditional rewrite rules of a logic, Thm. 3 yields its free 2-category of proofs.

Let $OCat$ be the extension of Cat with

$\rightsquigarrow:\ Mor\ A\ B\ \to\ Mor\ A\ B\ \to\ o.$

$\forall f : Mor\ A\ B.\ f \rightsquigarrow f$
$\forall f, g, h : Mor\ A\ B.\ (f \rightsquigarrow g\ \wedge\ g \rightsquigarrow h\ \Rightarrow\ f \rightsquigarrow h)$

where we interpret $\rightsquigarrow$ as rewritability between morphisms. Let K be a simple extension of $OCat$, and let K' be as K but with the additional axioms
$\forall X_1 : S_1 \ldots X'_m : S_m.$

$X_1 \rightsquigarrow X_1' \wedge \ldots \wedge X_m \rightsquigarrow X_m' \Rightarrow f\ X_1\ \ldots\ X_m \rightsquigarrow f\ X_1'\ \ldots\ X_m'$
for every function symbol f of depth 1. We call $Mod(K')$ the category of small order-enriched K-categories. The added axioms in K' are simply the congruence conditions for all function symbols with respect to rewriting. The arising axiomatization of rewriting is the same as in rewriting logic (see [BM03], which also allows frozen arguments).

In particular, this yields free order-enriched K-categories over Horn theories, and a complete axiomatization of this category. This allows a succinct view of logics under the Curry-Howard-Tait correspondence.[10]

Linear Algebra Let Σ be the following signature of depth 1

$N : \mathbf{S}.$
$one : N$
$succ : N \to N.$
$Mat : N \to N \to \mathbf{S}.$
$0 : R.$
$1 : R.$
$RowAppend : Mat\ m\ one \to R \to Mat\ succ\ m\ one.$
$ColAppend : Mat\ m\ n \to Mat\ m\ one \to Mat\ m\ succ\ n.$
$E : Mat\ m\ m.$
$+ : Mat\ m\ n \to Mat\ m\ n \to Mat\ m\ n.$
$- : Mat\ m\ n \to Mat\ m\ n \to Mat\ m\ n.$
$\cdot : Mat\ l\ m \to Mat\ m\ n \to Mat\ l\ n.$
$det : Mat\ m\ m \to R.$
$inv : Mat\ m\ m \to o.$
$eigenvalue : Mat\ m\ m \to R \to o.$

(where we use R to abbreviate $Mat\ one\ one$). Clearly, Σ can be used to axiomatize linear algebra over any ring that can be axiomatized in first-order logic. Examples for axioms are
$\forall M : Mat\ m\ m.\ (inv\ M \Leftrightarrow \neg\ det\ M \doteq 0)$ and
$\forall M : Mat\ m\ n,\ r : R.\ (eigenvalue\ M\ r \Leftrightarrow \exists v : Mat\ n\ one.\ M \cdot v \doteq v \cdot r)$
(with the usual abbreviation $\Leftrightarrow$).

6 Conclusion

We have introduced an extension of FOL with dependent types such that the generalization of definitions, results and implementations is very natural. The formulation of DFOL as an institution allows to apply the established institution-independent results (see the book [**?**]). The formulation of the syntax in LF

[10] Having a simple meta-language that supports dependent types while allowing axiomatization and free models was the original motivation to introduce DFOL. The idea for these specifications and the introduction of DFOL is due to Till Mossakowski and discussions with him and others.

immediately yields implementations of type checking, proof checking and simple theorem proving.

Of course, DFOL does not permit anything that has not been possible before. For example, category theory has been specified in Coq ([HS98]) or simply using FOL.[11] However, the simple model theory and the performance of automated theorem provers provide good arguments to stick to FOL if possible. The research presented here is targeted at those situations where specifications in (partial) FOL are desirable but awkward. In the examples, we demonstrated that only one or two dependent sort constructors can allow an elegant specification of a mathematical theory that would be awkward in FOL, but for which tools like Coq are far more powerful than necessary.

It can be argued that signatures of depth greater than 1 or 2 are not interesting. And in fact, the general case was not our original goal. But it turned out that the step from depth 0 to depth 1 is already almost as complex as the induction step for the general case so that no simplifications are to be expected from restricting the depth.

Although further work is needed (e.g., on resolution or Craig interpolation), it turned out that crucial classical results can be extended to DFOL. Free models make DFOL valuable as an algebraic specification language, and we plan to integrate it into CASL ([BM04]). And the axiomatization indicates that existing provers can be extended for DFOL. The FOL encoding in Twelf can be adapted easily so that both pure LF and the Twelf meta-theorem prover ([SP96]) can be applied.

References

[BC04] Y. Bertot and P. Castéran. *Coq'Art: The Calculus of Inductive Constructions*. Springer, 2004.

[BM03] R. Bruni and J. Meseguer. Generalized rewrite theories. In *Proceedings of ICALP '03*. Springer, 2003.

[BM04] Michel Bidoit and Peter D. Mosses. *Casl User Manual*. LNCS 2900 (IFIP Series). Springer, 2004.

[CAB+86] R. Constable, S. Allen, H. Bromley, W. Cleaveland, J. Cremer, R. Harper, D. Howe, T. Knoblock, N. Mendler, P. Panangaden, J. Sasaki, and S. Smith. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, 1986.

[Car86] J. Cartmell. Generalized algebraic theories and contextual category. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[CELM96] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of Maude. In J. Meseguer, editor, *Proceedings of the First International Workshop on Rewriting Logic*, volume 4, pages 65–89, 1996.

[Dyb95] P. Dybjer. Internal type theory. In *TYPES*, pages 120–134, 1995.

[11] DFOL cannot in general be encoded in partial many-sorted FOL since there may be more universes in a DFOL model then there are closed sort terms in the language. An encoding in FOL is straightforward and can provide an alternative completeness result but is very awkward.

[Gal86] J. Gallier. *Foundations of Automatic Theorem Proving*. Wiley, 1986.

[GWM+93] J. Goguen, Timothy Winkler, J. Meseguer, K. Futatsugi, and J. Jouannaud. Introducing OBJ. In Joseph Goguen, editor, *Applications of Algebraic Specification using OBJ*. Cambridge, 1993.

[Hen49] L. Henkin. The completeness of the first-order functional calculus. *Journal of Symbolic Logic*, 14:159–166, 1949.

[Hof94] M. Hofmann. On the Interpretation of Type Theory in Locally Cartesian Closed Categories. In *CSL*, pages 427–441. Springer, 1994.

[HS98] G. Huet and A. Saïbi. Constructive category theory. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1998.

[HST94] R. Harper, D. Sannella, and A. Tarlecki. Structured presentations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160, 1994.

[Mak] M. Makkai. First order logic with dependent sorts (FOLDS). Unpublished.

[ML74] P. Martin-Löf. An intuitionistic theory of types: Predicative part. In *Proceedings of the '73 Logic Colloquium*. North-Holland, 1974.

[NR01] R. Nieuwenhuis and A. Rubio. Paramodulation-Based theorem proving. In *Handbook of Automated Reasoning*, pages 371–443. Elsevier Science Publishers, 2001.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, 1992. Springer.

[OS97] S. Owre and N. Shankar. The formal semantics of PVS. Technical Report SRI-CSL-97-2, SRI International, 1997.

[Pfe01] F. Pfenning. Logical frameworks. In *Handbook of automated reasoning*, pages 1063–1147. Elsevier, 2001.

[PP03] B. Pientka and F. Pfenning. Optimizing higher-order pattern unification. In *19th International Conference on Automated Deduction*, pages 473–487. Springer, 2003.

[PS99] F. Pfenning and C. Schürmann. System description: Twelf - a meta-logical framework for deductive systems. *Lecture Notes in Computer Science*, 1632:202–206, 1999.

[RV02] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AI Communications*, 15:91–110, 2002.

[See84] R. Seely. Locally cartesian closed categories and type theory. *Math. Proc. Cambridge Philos. Soc.*, 95:33–48, 1984.

[SP96] C. Schürmann and F. Pfenning. Automated theorem proving in a simple meta-logic for LF. In C. Kirchner and H. Kirchner, editors, *Proceedings of the 15th International Conference on Automated Deduction*, pages 286–300. Springer, 1996.

[Vir96] R. Virga. Higher-order superposition for dependent types. In H. Ganzinger, editor, *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, pages 123–137. Springer, 1996.