The Lambda-Free Core of ELF

Robin Adams

December 20, 2004

1 Introduction

The earliest logical frameworks were those of the AUTOMATH project (see [**?**] among other places for a description). The simplest of this family of systems was the language PAL (Primitive AUTOMATH Language)[1]. PAL is an example of a *lambda-free* logical framework; it contained no concepts taken from the lambda calculus. In PAL, it was possible to declare types, objects of types, and functions that map objects to objects or objects to types; but it was not possible to declare functionals (functions whose arguments are themselves functions). Thus, PAL could represent propositional connectives and Hilbert-style deduction systems, but not any grammar that involved binding variables, nor natural deduction systems, in which hypotheses can be assumed and later discharged.

The solution used in AUTOMATH, and every logical framework since, is to incorporate the mechanisms of the typed lambda calculus. But this solution brings several problems of its own.

When building a theory within a logical framework, we ideally require two properties:

Soundness

Every term of the object theory is represented by a term of the logical framework.

Adequacy

Every term of the logical framework represents a term of the object theory.

However, when using a logical framework based on the typed lambda calculus, these two desiderata have to be compromised somewhat. The framework now contains many terms, such as partially applied functions, that do not correspond to anything in the object theory. Further, every expression of the object theory is now interpreted, not by a single term in the logical framework, but by a $\beta\eta$-equivalence class of terms.

Even after we rephrase the soundness and adequacy criteria, they are usually difficult to prove; adequacy in particular. We often need to use metatheoretic

[1] In fact, the AUTOMATH family included an even simpler language: SEMIPAL, an untyped version of PAL.

results about the framework, such as normalization properties, that are difficult to establish. Much of this work needs to be repeated for each theory we build within the framework.

In this paper, we present one piece of evidence towards the following thesis: Given any logical framework, it is possible to construct a system (which we call its *lambda-free core*) that has the same power as the original framework, but does not contain the lambda calculus. This core can be produced from the original framework in a fairly methodical way. It is a simpler system than the original framework, and has a simpler metatheory. Further, it provides a more faithful representation of the object theories; every term of the object theory corresponds to a single term of the logical framework, and vice versa. Because of this close correspondence, such results as completeness and adequacy become trivial to establish.

We illustrate this thesis here by dealing with the case of the Edinburgh Logical Framework, ELF [2]. We call the lambda-free core the *rudimentary framework*, RF[2]. We demonstrate the better representation of object theories with the example of an arbitrary first-order theory.

2 The Edinburgh Logical Framework

We present here the grammar and rules of deduction of the Edinburgh Logical Framework (ELF). We stay as close as possible to the presentation in [2]; the main difference is that we assign arities to the variables, constants and terms.

2.1 Grammar

The set of *arities* is defined by the grammar

$$\text{Arity } \alpha ::= (\alpha, \ldots, \alpha)$$

Intuitively, an $(\alpha_1, \ldots, \alpha_n)$-ary function is a function that takes, as arguments, an α_1-ary function, an α_2-ary function, ..., and an α_n-ary function, and returns an object. (Thus, an object is a ()-ary function.)

Fix, for each arity α, disjoint, countably infinite sets $\mathcal{V}_\alpha$ of *α-ary variables*, $\mathcal{C}^{\mathcal{O}}_\alpha$ of *α-ary object-level constants*, and $\mathcal{C}^{\mathcal{F}}_\alpha$ of *α-ary family-level constants*.

We now define, for arities α, β, the sets:

- $\mathcal{K}_\alpha$ of *α-ary kinds*;
- $\mathcal{F}^\beta_\alpha$ of *α-ary families of β-ary types* (or *α, β-ary family*);
- $\mathcal{O}_\alpha$ of *α-ary objects*.

Intuitively:

[2]I gave the theory this name as I was originally studying it as a subsystem of TF (Type Framework), the system that I was interested in at the time. It was only later that I noticed the correspondence with ELF.

- An α-ary object is an α-ary function whose values are terms of the object theory.
- An α, β-ary family as an α-ary function whose values are types, the objects of which are β-ary objects. We shall sometimes refer to α as its *base* arity, and β as its *target* arity.
- An α-ary kind is one whose objects are α, β-ary families for some β.

The definition is as follows (see also Figure 1):

Kinds

- **type** is a $()$-ary kind.
- If x is an α_0-ary variable, A an $(), \alpha_0$-ary family, and K an $(\alpha_1, \ldots, \alpha_n)$-ary kind, then $\Pi x : A.K$ is an $(\alpha_0, \alpha_1, \ldots, \alpha_n)$-ary kind.

Families

- Every α-ary family-level constant is an $\alpha, ()$-ary family.
- If x is an α_0-ary variable, A an $(), \alpha_0$-ary family, and B a $(), (\alpha_1, \ldots, \alpha_n)$-ary family, then $\Pi x : A.B$ is a $(), (\alpha_0, \alpha_1, \ldots, \alpha_n)$-ary family.
- If x is an α_0-ary variable, A an $(), \alpha_0$-ary family, and B a $(\alpha_1, \ldots, \alpha_n), \beta$-ary family, then $\lambda x : A.B$ is a $(\alpha_0, \alpha_1, \ldots, \alpha_n), \beta$-ary family.
- If A is an $(\alpha_0, \alpha_1, \ldots, \alpha_n), \beta$-ary family, and M an α_0-ary object, then AM is a $(\alpha_1, \ldots, \alpha_n), \beta$-ary family.

Objects

- Every α-ary object-level constant is an α-ary object.
- Every α-ary variable is an α-ary object.
- If x is an α_0-ary variable, A a $(), \alpha_0$-ary family, and M an $(\alpha_1, \ldots, \alpha_n)$-ary object, then $\lambda x : A.M$ is an $(\alpha_0, \alpha_1, \ldots, \alpha_n)$-ary object.
- If M is an $(\alpha_0, \alpha_1, \ldots, \alpha_n)$-ary object, and N an α_0-ary object, then MN is an $(\alpha_1, \ldots, \alpha_n)$-ary object.

We identify all these expressions up to α-conversion. We write $E \to E'$ for $\Pi x : E.E'$ when x does not occur free in E'.

We define β-conversion on all these three classes of expression by the usual reduction rule:

$$(\lambda x : A.E)M \rightsquigarrow_\beta [M/x]E$$

Note that, if $E =_\beta E'$, then E and E' have the same arity.

Kinds

$$\frac{}{\mathbf{type} \in \mathcal{K}_{()}} \qquad \frac{x \in \mathcal{V}_{\alpha_0} \quad A \in \mathcal{F}^{\alpha_0}_{()} \quad K \in \mathcal{K}_{(\alpha_1,\ldots,\alpha_n)}}{\Pi x : A.K \in \mathcal{K}_{(\alpha_0,\alpha_1,\ldots,\alpha_n)}}$$

Families

$$\frac{a \in \mathcal{C}^{\mathcal{F}}_{\alpha}}{a \in \mathcal{F}^{()}_{\alpha}} \qquad \frac{x \in \mathcal{V}_{\alpha_0} \quad A \in \mathcal{F}^{\alpha_0}_{()} \quad B \in \mathcal{F}^{(\alpha_1,\ldots,\alpha_n)}_{()}}{\Pi x : A.B \in \mathcal{F}^{(\alpha_0,\alpha_1,\ldots,\alpha_n)}_{()}}$$

$$\frac{x \in \mathcal{V}_{\alpha_0} \quad A \in \mathcal{F}^{\alpha_0}_{()} \quad B \in \mathcal{F}^{\beta}_{(\alpha_1,\ldots,\alpha_n)}}{\lambda x : A.B \in \mathcal{F}^{\beta}_{(\alpha_0,\alpha_1,\ldots,\alpha_n)}} \qquad \frac{A \in \mathcal{F}^{\beta}_{(\alpha_0,\alpha_1,\ldots,\alpha_n)} \quad M \in \mathcal{O}_{\alpha_0}}{AM \in \mathcal{F}^{\beta}_{(\alpha_1,\ldots,\alpha_n)}}$$

Objects

$$\frac{c \in \mathcal{C}^{\mathcal{O}}_{\alpha}}{c \in \mathcal{O}_{\alpha}} \qquad \frac{x \in \mathcal{V}_{\alpha}}{x \in \mathcal{O}_{\alpha}} \qquad \frac{x \in \mathcal{V}_{\alpha_0} \quad A \in \mathcal{F}^{\alpha_0}_{()} \quad M \in \mathcal{O}_{(\alpha_1,\ldots,\alpha_n)}}{\lambda x : A.M \in \mathcal{O}_{(\alpha_0,\alpha_1,\ldots,\alpha_n)}} \qquad \frac{M \in \mathcal{O}_{(\alpha_0,\alpha_1,\ldots,\alpha_n)} \quad N \in \mathcal{O}_{\alpha_0}}{MN \in \mathcal{O}_{(\alpha_1,\ldots,\alpha_n)}}$$

Figure 1: Grammar of ELF

Definition 2.1 *1. A* signature *is a sequence whose members are either:*

- *pairs $a : K$, where $a \in \mathcal{C}^{\mathcal{F}}_{\alpha}$, $K \in \mathcal{K}_{\alpha}$ for some α; or*
- *pairs $c : A$, where $c \in \mathcal{C}^{\mathcal{O}}_{\alpha}$, $A \in \mathcal{F}^{\alpha}_{()}$ for some α.*

2. A context *is a sequence of pairs $x : A$, where $x \in \mathcal{V}_{\alpha}$, and $A \in \mathcal{F}^{\alpha}_{()}$ for some α.*

We shall refer to $x_1 : A_1, \ldots, x_n : A_n$ as an $(\alpha_1, \ldots, \alpha_n)$-*ary context*, where α_i is the arity of x_i (and the target arity of A_i). We could also define the arity of a signature in this way, but we shall not have need to.

The *domain* of the context $\Gamma \equiv x_1 : A_1, \ldots, x_n : A_n$, $\mathrm{dom}\,\Gamma$, is defined to be the sequence $x_1, \ldots, x_n$. The domain of a signature is defined similarly.

The signatures are used in ELF to specify the object theory. The contexts are then used to represent the free variables, undischarged hypotheses, etc. when working within the theory.

The judgement forms of ELF are of five kinds:

Σ sig	Σ is a valid signature
$\vdash_{\Sigma} \Gamma$	Γ is a valid context in Σ
$\Gamma \vdash_{\Sigma} K$	K is a kind in Γ and Σ
$\Gamma \vdash_{\Sigma} A : K$	A has kind K in Γ and Σ
$\Gamma \vdash_{\Sigma} M : A$	M has type A in Γ and Σ

where Σ is a signature, Γ a context, K a kind, A a family, and M an object. In the fourth judgement form, we demand that $A \in \mathcal{F}^{\beta}_{\alpha}$, $K \in \mathcal{F}_{\alpha}$ for some α, β. In the fifth judgement form, we demand that $M \in \mathcal{O}_{\alpha}$, $A \in \mathcal{F}^{\alpha}_{()}$ for some α.

$$\frac{}{\langle\rangle \text{ sig}}$$

$$\frac{\Sigma \text{ sig} \quad \vdash_\Sigma K}{\Sigma, a : K \text{ sig}} \, (a \notin \operatorname{dom}\Sigma) \qquad \frac{\Sigma \text{ sig} \quad \vdash_\Sigma A : \mathbf{type}}{\Sigma, c : A \text{ sig}} \, (c \notin \operatorname{dom}\Sigma)$$

$$\frac{\Sigma \text{ sig}}{\vdash_\Sigma \langle\rangle} \qquad \frac{\vdash_\Sigma \Gamma \quad \Gamma \vdash_\Sigma A : \mathbf{type}}{\vdash_\Sigma \Gamma, x : A} \, (x \notin \operatorname{dom}\Gamma)$$

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma \mathbf{type}} \qquad \frac{\Gamma, x : A \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi x : A.K}$$

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma a : K} \, (a : K \in \Sigma) \qquad \frac{\Gamma, x : A \vdash_\Sigma B : \mathbf{type}}{\Gamma \vdash_\Sigma \Pi x : A.B : \mathbf{type}}$$

$$\frac{\Gamma, x : A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda x : A.B : \Pi x : A.K} \qquad \frac{\Gamma \vdash_\Sigma A : \Pi x : B.K \quad \Gamma \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma AM : [M/x]K}$$

$$\frac{\Gamma \vdash_\Sigma A : K \quad \Gamma \vdash_\Sigma K'}{\Gamma \vdash_\Sigma A : K'} \, (K =_\beta K')$$

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma c : A} \, (c : A \in \Sigma) \qquad \frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma x : A} \, (x : A \in \Gamma)$$

$$\frac{\Gamma, x : A \vdash_\Sigma M : B}{\Gamma \vdash_\Sigma \lambda x : A.M : \Pi x : A.B} \qquad \frac{\Gamma \vdash_\Sigma M : \Pi x : A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma MN : [N/x]B}$$

$$\frac{\Gamma \vdash_\Sigma M : A \quad \Gamma \vdash_\Sigma A' : \mathbf{type}}{\Gamma \vdash_\Sigma M : A'} \, (A =_\beta A')$$

Figure 2: Rules of Deduction of ELF

The rules of deduction of ELF are given in Figure 2.
ELF is thus essentially the PTS with

- Sorts **type**, **kind**
- Axiom **type** : **kind**
- Rules (**type**, **type**), (**type**, **kind**).

The main differences from the usual style of presentation of PTSs are the separation of declarations into signature and context, and the formal distinction between variables and the two classes of constants.

3 Rudimentary Framework

We now perform the procedure that may be described as extracting the lambda-free core of this system. We operate in the following manner on the system:

- We remove every term except the normal forms (that is, the β-normal, η-long forms).
- We remove Π-types everywhere except in the declarations of variables and constants.
- We remove λ-abstractions everywhere except in the argument positions of function applications.

For example, consider the ELF judgement

$$\Gamma \vdash \lambda\Delta.M : \Pi\Delta'.A$$

If $\Delta =_\beta \Delta'$, then this judgement is derivable if and only if the judgement

$$\Gamma, \Delta \vdash M : A$$

is derivable; we can replace the first judgement with the second. If $\Delta \neq_\beta \Delta'$, then the judgement is never derivable; we can simply remove it from the system, and it will never be missed.

It is possible to perform these operations so that the only judgement forms remaining are

$$\Sigma \text{ sig}, \vdash_\Sigma \Gamma, \ \Gamma \vdash_\Sigma A : \mathbf{type}, \ \Gamma \vdash_\Sigma M : A$$

where all terms are in normal form, $A \in \mathcal{F}^{()}_{()}$, $M \in \mathcal{O}_{()}$. Further, the resulting system is simpler than the original ELF, and possesses several theoretical and practical advantages.

We proceed to describe this system, which we call the Rudimentary Framework, RF.

3.1 Grammar

We define the set of objects and types that RF uses. It would be possible to describe these as those ()-ary objects and $(),()$-ary families of ELF that are in normal form; however, to make RF self-contained, we present an independent definition. It is necessary to define the set of contexts in normal form simultaneously.

Definition 3.1 *Define simultaneously the sets:*

- $\mathcal{O}^*$ *of* nf objects
- $\mathcal{F}^*$ *of* nf types
- $\mathcal{D}^*_\alpha$ *of* α-ary nf contexts

as follows:

- *If a is an $(\alpha_1,\ldots,\alpha_n)$-ary family-level constant, and, for $i=1,\ldots,n$, Δ_i is an α_i-ary nf context and M_i an nf object, then*

$$a(\lambda\Delta_1.M_1)\cdots(\lambda\Delta_n.M_n)$$

  is an nf type.

- *If z is an $(\alpha_1,\ldots,\alpha_n)$-ary object-level constant or variable, and, for $i=1,\ldots,n$, Δ_i is an α_i-ary nf context and M_i an nf object, then*

$$z(\lambda\Delta_1.M_1)\cdots(\lambda\Delta_n.M_n)$$

  is an nf object.

- *If, for $i=1,\ldots,n$, x is an α_i-ary variable, Δ_i an α_i-ary nf context, and A_i an nf type, then*

$$x_1:\Pi\Delta_1.A_1,\ldots,x_n:\Pi\Delta_n.A_n$$

  is an $(\alpha_1,\ldots,\alpha_n)$-ary nf context.

See also Figure 3

Proposition 3.2 *$A\in\mathcal{F}^{()}_{()}$ is in β-normal, η-long form iff $A\in\mathcal{F}^*$.*
$M\in\mathcal{O}_{()}$ is in β-normal, η-long form iff $M\in\mathcal{O}^*$.
The α-ary context Γ is in β-normal, η-long form iff $\Gamma\in\mathcal{D}^*_\alpha$.

In *RF*, we define an *nf signature* to be a sequence whose elements are either

- pairs $c:\Pi\Delta.A$, where $c\in\mathcal{C}^{\mathcal{O}}_\alpha$, $\Delta\in\mathcal{D}^*_\alpha$ and $A\in\mathcal{F}^*$ for some α; or
- pairs $a:\Pi\Delta.\mathbf{type}$, where $a\in\mathcal{C}^{\mathcal{F}}_\alpha$, and $\Delta\in\mathcal{D}^*_\alpha$ for some α.

$$\frac{a \in \mathcal{C}^{\mathcal{F}}_{(\alpha_1,\ldots,\alpha_n)} \quad \Delta_i \in \mathcal{D}^*_{\alpha_i}\ M_i \in \mathcal{O}^*\ (i=1,\ldots,n)}{a(\lambda\Delta_1.M_1)\cdots(\lambda\Delta_n.M_n) \in \mathcal{F}^*}$$

$$\frac{c \in \mathcal{C}^{\mathcal{O}}_{(\alpha_1,\ldots,\alpha_n)} \quad \Delta_i \in \mathcal{D}^*_{\alpha_i}\ M_i \in \mathcal{O}^*\ (i=1,\ldots,n)}{c(\lambda\Delta_1.M_1)\cdots(\lambda\Delta_n.M_n) \in \mathcal{O}^*}$$

$$\frac{x \in \mathcal{V}^{\mathcal{O}}_{(\alpha_1,\ldots,\alpha_n)} \quad \Delta_i \in \mathcal{D}^*_{\alpha_i}\ M_i \in \mathcal{O}^*\ (i=1,\ldots,n)}{x(\lambda\Delta_1.M_1)\cdots(\lambda\Delta_n.M_n) \in \mathcal{O}^*}$$

$$\frac{x_i \in \mathcal{V}_{\alpha_i} \quad \Delta_i \in \mathcal{D}^*_{\alpha_i} \quad A_i \in \mathcal{F}^* \quad (i=1,\ldots,n)}{x_1 : \Pi\Delta_1.A_1, \ldots, x_n : \Pi\Delta_n.A_n \in \mathcal{D}^*_{(\alpha_1,\ldots,\alpha_n)}}$$

Figure 3: Grammar of RF

The judgement forms of RF are of the following forms:

$$\Sigma \text{ sig}$$
$$\vdash_\Sigma \Gamma$$
$$\Gamma \vdash_\Sigma A : \mathbf{type}$$
$$\Gamma \vdash_\Sigma M : A$$

where Σ is an nf signature, $\Gamma \in \mathcal{D}^*_\alpha$, $A \in \mathcal{F}^*$, and $M \in \mathcal{O}^*$.

3.2 Instantiation

The result of substitution $\lambda\Delta.M$ for a constant or variable within a normal form is not always a normal form. Thus, RF cannot use simple substitution. We define the following operation, which we name *instantiation*, to take its place.

Definition 3.3 (Instantiation) *Let $\alpha_1, \ldots, \alpha_n$ be arities. We define the nf object (type, context)*

$$\{\lambda\Delta_1.M_1/x_1, \ldots, \lambda\Delta_n.M_n/x_n\}E$$

where $\Delta_i \in \mathcal{D}^*_{\alpha_i}$, $M_i \in \mathcal{O}^*$, $x_i \in \mathcal{V}_{\alpha_i}$, and E is an nf object (type, context). The definition is as follows, writing E^* for the term above.

$$\begin{aligned}
(z(\lambda\Gamma_1.N_1)\cdots(\lambda\Gamma_m.N_m))^* &\equiv z(\lambda\Gamma^*_1.N^*_1)\cdots(\lambda\Gamma^*_m.N^*_m) \qquad (z \notin \{x_1,\ldots,x_n\}) \\
(x_i(\lambda\Gamma_1.N_1)\cdots(\lambda\Gamma_m.N_m))^* &\equiv \{\lambda\Gamma^*_1.N^*_1/y_1, \ldots, \lambda\Gamma^*_m.N^*_m/y_m\}M_i \qquad (\mathrm{dom}\,\Delta_i \equiv y_1,\ldots,y_m) \\
(x_1 : \Pi\Gamma_1.A_1, \ldots, x_n : \Pi\Gamma_m.A_m)^* &\equiv x_1 : \Pi\Gamma^*_1.A^*_1, \ldots, x_n : \Pi\Gamma^*_m.A^*_m
\end{aligned}$$

where $\mathbf{type}^* \equiv \mathbf{type}$.

Note The main reason we assign arities to variables, constants and expressions is to make this definition total, by ensuring that, in the second clause above, the number of variables in $\mathrm{dom}\,\Delta_i$ and the number of arguments that x_i is applied to are the same. An alternative would be not to use arities at all, and to leave E^* undefined whenever these two numbers do not match. We would then have to prove the result that E^* is defined whenever it is called for in the rules of deduction of RF. Either approach has its advantages and disadvantages.

The following result may make the meaning of instantiation clearer.

Proposition 3.4 *$\{\lambda\Delta_1.M_1/x_1,\ldots,\lambda\Delta_n.M_n/x_n\}E$ is the β-normal, η-long normal form of $[\lambda\Delta_1.M_1/x_1,\ldots,\lambda\Delta_n.M_n/x_n]E$.*

3.3 Satisfaction of a Context

We define a relation of *satisfaction* between a sequence of λ-abstractions and an nf context as follows:

Let

$$\Delta \equiv x_1 : \Pi\Delta_1.A_1,\ldots,x_n : \Pi\Delta_n.A_n$$

The sequence

$$\lambda\Delta_1.M_1,\ldots,\lambda\Delta_n.M_n$$

satisfies this context in Γ, Σ iff the following judgements are derivable:

$$\begin{array}{lll}
\Gamma,\Delta_1 & \vdash_\Sigma & M_1 : A_1\\
\Gamma,\Delta_2^* & \vdash_\Sigma & M_2 : A_2^*\\
 & \vdots & \\
\Gamma,\Delta_n^* & \vdash_\Sigma & M_n : A_n^*
\end{array}$$

where

$$\begin{array}{lcl}
\Delta_i^* & \equiv & \{\lambda\Delta_1.M_1/x_1,\ldots,\lambda\Delta_{i-1}.M_{i-1}/x_{i-1}\}\Delta_i\\
A_i^* & \equiv & \{\lambda\Delta_1.M_1/x_1,\ldots,\lambda\Delta_{i-1}.M_{i-1}/x_{i-1}\}A_i
\end{array}$$

We write

$$\Gamma \Vdash_\Sigma \hat{}\vec{M} :: \Delta$$

for this set of judgements. The symbol $\Vdash$ and the double colon indicate that this is a set of judgements, not a single judgement. $\hat{}\vec{M}$ denotes the sequence $\lambda\Delta_1.M_1,\ldots,\lambda\Delta_n.M_n$. The prefix $\hat{}$ emphasizes the fact that $\lambda\Delta_i.M_i$ is not a first-class object of RF.

In the case $n = 0$, we take $\Gamma \Vdash_\Sigma \hat{}\vec{M} :: \Delta$ (i.e. $\Gamma \Vdash_\Sigma \langle\rangle :: \langle\rangle$) to denote the single judgement

$$\vdash_\Sigma \Gamma$$

$$\frac{}{\langle\rangle\ \text{sig}}$$

$$\frac{\vdash_\Sigma \Delta}{\Sigma, a : \Pi\Delta.\mathbf{type}\ \text{sig}}\ (a \notin \text{dom}\,\Sigma) \qquad \frac{\Delta \vdash_\Sigma A : \mathbf{type}}{\Sigma, c : \Pi\Delta.A\ \text{sig}}\ (c \notin \text{dom}\,\Sigma)$$

$$\frac{\Sigma\ \text{sig}}{\vdash_\Sigma \langle\rangle} \qquad \frac{\Gamma, \Delta \vdash_\Sigma A : \mathbf{type}}{\vdash_\Sigma \Gamma, x : \Pi\Delta.A}\ (x \notin \text{dom}\,\Gamma)$$

$$\frac{\Gamma \Vdash_\Sigma \hat{\vec{M}} :: \Delta}{\Gamma \vdash_\Sigma a(\hat{\vec{M}}) : \mathbf{type}}\ (a : \Pi\Delta.\mathbf{type} \in \Sigma)$$

$$\frac{\Gamma \Vdash_\Sigma \hat{\vec{M}} :: \Delta}{\Gamma \vdash_\Sigma c(\hat{\vec{M}}) : \{\vec{M}/\,\text{dom}\,\Delta\}A}\ (c : \Pi\Delta.A \in \Sigma)$$

$$\frac{\Gamma \Vdash_\Sigma \hat{\vec{M}} :: \Delta}{\Gamma \vdash_\Sigma x(\hat{\vec{M}}) : \{\vec{M}/\,\text{dom}\,\Delta\}A}\ (x : \Pi\Delta.A \in \Gamma)$$

Figure 4: Rules of Deduction of RF

3.4 Rules of Deduction

The rules of deduction of RF are given in Figure 4. Compare the complexity of Figures 2 and 4.

The equivalence of RF and ELF is proven as follows. Let $NF(E)$ denote the β-normal, η-long form of the ELF expression E.

Theorem 3.5 *Every judgement derivable in RF is derivable in ELF.*

Theorem 3.6 *1. If* $\Sigma\ \text{sig}_{\text{ELF}}$*, then* $NF(\Sigma)\ \text{sig}_{\text{RF}}$*.*

2. If $\vdash^{\text{ELF}}_\Sigma \Gamma$*, then* $\vdash^{\text{RF}}_{NF(\Sigma)} NF(\Gamma)$*.*

3. If $\Gamma \vdash^{\text{ELF}}_\Sigma \Pi\Delta.\mathbf{type}$*, then*

$$\vdash^{\text{RF}}_{NF(\Sigma)} NF(\Gamma), NF(\Delta)$$

4. If $\Gamma \vdash^{\text{ELF}}_\Sigma A : \Pi\Delta.\mathbf{type}$*, then*

$$NF(A(\text{dom}\,\Delta)) \equiv \Pi\Theta.D$$

and

$$NF(\Gamma), NF(\Delta), \Theta \vdash^{\text{RF}}_{NF(\Sigma)} D : \mathbf{type}$$

5. If $\Gamma \vdash^{\text{ELF}}_\Sigma M : A$*, then*

$$NF(A) \equiv \Pi\Theta.D$$

and

$$NF(\Gamma), \Theta \vdash^{\text{RF}}_{NF(\Sigma)} NF(M(\text{dom}\,\Theta)) : D$$

Corollary 3.7 *The RF judgement J is derivable in ELF iff J is derivable in RF.*

An inspection of the proof of Theorem 3.5 reveals that the kind-level abstraction rule in ELF

$$\frac{\Gamma, x : A \vdash_\Sigma B : K}{\Gamma \vdash_\Sigma \lambda x : A.B : \Pi x : A.K}$$

is never used when deriving an RF judgement in ELF. Thus, we have a further

Corollary 3.8 *If the derivable ELF judgement J is in normal form, and not of the form*

$$\Gamma \vdash_\Sigma A : \Pi\Delta.K$$

nor

$$\Gamma \vdash_\Sigma K$$

then there is a derivation of J that does not use the kind-level abstraction rule.

Thus, the kind-level abstraction rule, and the families of the form $\lambda x : A.B$, are redundant. This result was first proven in [1], Corollary 4.3.

3.5 Decidability

Theorem 3.9 *RF is decidable.*

The proof is quite easy, as RF is syntax directed; that is, given any judgement, there is at most one rule of deduction that could possibly have produced that judgement. The decision procedure consists simply of repeatedly inverting the rules of RF. Compare this with the difficulty of proving ELF decidable (Theorem A.17 of [2]).

4 Construction of a First-Order Theory

To illustrate the relative advantages of the two frameworks, we construct an arbitrary first-order theory in each. We first remind ourselves of the following

Definition 4.1 *A* first-order language *consists of a set of* n-ary function symbols *and a set of* n-ary predicate symbols *for each natural number n. (We are taking constants to be 0-ary function symbols.) We also provide ourselves with a countably infinite set of* variables

The set of terms *is defined inductively thus:*

- *Every variable is a term.*
- *If f is an n-ary function symbol, and $t_1, \ldots, t_n$ are terms, then*

$$f(t_1, \ldots, t_n)$$

is a term.

The set of propositions *is defined inductively thus:*

- *If R is an n-ary predicate symbol, and t_1, ..., t_n are terms, then*

$$R(t_1, \ldots, t_n)$$

  is a proposition.

- *If ϕ is a proposition, so is $\neg\phi$.*

- *If ϕ and ψ are propositions, so are $\phi \vee \psi$, $\phi \wedge \psi$, and $\phi \supset \psi$.*

- *If ϕ is a proposition and x a variable, then $\forall x\phi$ and $\exists x\phi$ are propositions.*

A theory *is a set of propositions, the* axioms *of the theory. The set of* theorems *of the theory are then defined by the rules of deduction given in Figure 5.*

The rules we have given are for classical first-order logic; if one prefers to work with the rules for intuitionistic logic, nothing essential will change.

Consider a first-order language. The following signature specifies the language within either ELF or RF. Notice how closely the signature corresponds to the definition above.

$$\begin{array}{rcl}
\text{term} & : & \textbf{type} \\
\multicolumn{3}{c}{\text{For each } n\text{-ary function symbol } f,} \\
\overline{f} & : & \overbrace{\text{term} \to \cdots \to \text{term}}^{n} \to \text{term} \\
\text{prop} & : & \textbf{type} \\
\multicolumn{3}{c}{\text{For each } n\text{-ary predicate symbol } R,} \\
\overline{R} & : & \overbrace{\text{term} \to \cdots \to \text{term}}^{n} \to \text{prop} \\
\neg & : & \text{prop} \to \text{prop} \\
\vee & : & \text{prop} \to \text{prop} \to \text{prop} \\
\wedge & : & \text{prop} \to \text{prop} \to \text{prop} \\
\supset & : & \text{prop} \to \text{prop} \to \text{prop} \\
\forall & : & (\text{term} \to \text{prop}) \to \text{prop} \\
\exists & : & (\text{term} \to \text{prop}) \to \text{prop}
\end{array}$$

(Note: as we have defined a signature to be a finite sequence of pairs, we can only represent finite languages.)

We can define a translation $[\![]\!]$ from the first-order language into ELF or RF as follows. We define, for each term t or proposition ϕ, the object $[\![t]\!]$ or $[\![\phi]\!]$,

$$(\neg E)\ \frac{\begin{array}{cc}[\neg\phi] & [\neg\phi]\\ \vdots & \vdots\\ \psi & \neg\psi\end{array}}{\phi}$$

$$(\vee Il)\ \frac{\phi}{\phi\vee\psi}\qquad (\vee Ir)\ \frac{\psi}{\phi\vee\psi}\qquad (\vee E)\ \frac{\begin{array}{ccc}\phi\vee\psi & \begin{array}{c}[\phi]\\ \vdots\\ \chi\end{array} & \begin{array}{c}[\psi]\\ \vdots\\ \chi\end{array}\end{array}}{\chi}$$

$$(\wedge I)\ \frac{\phi\quad\psi}{\phi\wedge\psi}\qquad (\wedge El)\ \frac{\phi\wedge\psi}{\phi}\qquad (\wedge Er)\ \frac{\phi\wedge\psi}{\psi}$$

$$(\supset I)\ \frac{\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\phi\supset\psi}\qquad (\supset E)\ \frac{\phi\supset\psi\quad\phi}{\psi}$$

$$(\forall I)\ \frac{\phi}{\forall x\phi}\ (x \text{ not free in the hypotheses})\qquad (\forall E)\ \frac{\forall x\phi}{[t/x]\phi}\ (t \text{ free for } x \text{ in } \phi)$$

$$(\exists I)\ \frac{[t/x]\phi}{\exists x\phi}\ (t \text{ free for } x \text{ in } \phi)\qquad (\exists E)\ \frac{\exists x\phi\qquad\begin{array}{c}[\phi]\\ \vdots\\ \psi\end{array}}{\psi}\ (x \text{ not free in } \psi)$$

Figure 5: First-Order Logic

which will be an object of both ELF and RF, as follows:

$$\begin{aligned}
[\![x]\!] &\equiv x \\
[\![f(t_1,\ldots,t_n)]\!] &\equiv \overline{f}\,[\![t_1]\!]\cdots[\![t_n]\!] \\
[\![R(t_1,\ldots,t_n)]\!] &\equiv \overline{R}\,[\![t_1]\!]\cdots[\![t_n]\!] \\
[\![\neg\phi]\!] &\equiv \neg\,[\![\phi]\!] \\
[\![\phi\vee\psi]\!] &\equiv \vee\,[\![\phi]\!]\,[\![\psi]\!] \\
[\![\phi\wedge\psi]\!] &\equiv \wedge\,[\![\phi]\!]\,[\![\psi]\!] \\
[\![\phi\supset\psi]\!] &\equiv \supset\,[\![\phi]\!]\,[\![\psi]\!] \\
[\![\forall x\phi]\!] &\equiv \forall\lambda x : \mathrm{term}.\,[\![\phi]\!] \\
[\![\exists x\phi]\!] &\equiv \exists\lambda x : \mathrm{term}.\,[\![\phi]\!]
\end{aligned}$$

We can now define a signature for each theory with this language by adding the following declarations to the above signature:

$$\begin{aligned}
\mathrm{Prf} &: \mathrm{prop}\to\mathbf{type} \\
&\text{For each axiom } \alpha, \\
\overline{\alpha} &: \mathrm{Prf}\,[\![\alpha]\!] \\
\neg E &: \Pi p,q:\mathrm{prop}.(\mathrm{Prf}(\neg p)\to\mathrm{Prf}\,q)\to(\mathrm{Prf}(\neg p)\to\mathrm{Prf}(\neg q))\to\mathrm{Prf}\,p \\
\vee Il &: \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}\,p\to\mathrm{Prf}(\vee pq) \\
\vee Ir &: \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}\,q\to\mathrm{Prf}(\vee pq) \\
\vee E &: \Pi p,q,r:\mathrm{prop}.\,\mathrm{Prf}(\vee pq)\to(\mathrm{Prf}\,p\to\mathrm{Prf}\,r)\to(\mathrm{Prf}\,q\to\mathrm{Prf}\,r)\to\mathrm{Prf}\,r \\
\wedge I &: \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}\,p\to\mathrm{Prf}\,q\to\mathrm{Prf}(\wedge pq) \\
\wedge El &: \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}(\wedge pq)\to\mathrm{Prf}\,p \\
\wedge Er &; \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}(\wedge pq)\to\mathrm{Prf}\,q \\
\supset I &: \Pi p,q:\mathrm{prop}.(\mathrm{Prf}\,p\to\mathrm{Prf}\,q)\to\mathrm{Prf}(\supset pq) \\
\supset E &: \Pi p,q:\mathrm{prop}.\,\mathrm{Prf}(\supset pq)\to\mathrm{Prf}\,p\to\mathrm{Prf}\,q \\
\forall I &: \Pi P:(\mathrm{term}\to\mathrm{prop}).(\Pi x:\mathrm{term}.\,\mathrm{Prf}(Px))\to\mathrm{Prf}(\forall P) \\
\forall E &: \Pi P:(\mathrm{term}\to\mathrm{prop}).\Pi t:\mathrm{term}.\,\mathrm{Prf}(\forall P)\to\mathrm{Prf}(Pt) \\
\exists I &: \Pi P:(\mathrm{term}\to\mathrm{prop}).\Pi t:\mathrm{term}.\,\mathrm{Prf}(Pt)\to\mathrm{Prf}(\exists P) \\
\exists E &: \Pi P:(\mathrm{term}\to\mathrm{prop}).\Pi p:\mathrm{prop}.\,\mathrm{Prf}(\exists P)\to(\Pi x:\mathrm{term}.\,\mathrm{Prf}(Px)\to\mathrm{Prf}\,p)\to\mathrm{Prf}\,p
\end{aligned}$$

Let Σ denote this signature.

The correspondence between the first-order theory and the behaviour of ELF under this signature is given by the following two theorems:

Proposition 4.2 (Soundness) *In ELF:*

1. *For every term t, if $FV(t)=\{x_1,\ldots,x_n\}$, then*

$$x_1,\ldots,x_n:\mathrm{term}\vdash_\Sigma [\![t]\!]:\mathrm{term}$$

2. *For every proposition ϕ, if $FV(\phi) = \{x_1, \ldots, x_n\}$, then*

$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma [\![\phi]\!] : \text{prop}$$

3. *If $\phi_1, \ldots, \phi_n \vdash \psi$, and $FV(\phi_1) \cup \cdots \cup FV(\phi_n) \cup FV(\psi) = \{x_1, \ldots, x_m\}$, then there is a term P such that*

$$x_1, \ldots, x_m : \text{term}, p_1 : \text{Prf}\,[\![\phi_1]\!], \ldots, p_n : \text{Prf}\,[\![\phi_n]\!] \vdash_\Sigma P : \text{Prf}\,[\![\psi]\!]$$

Proposition 4.3 (Adequacy) *In ELF:*

1. *If*

$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma T : \text{term}$$

then there is a term t with $FV(t) \subseteq \{x_1, \ldots, x_n\}$ such that $T =_{\beta\eta} [\![t]\!]$.

2. *If*

$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma P : \text{prop}$$

then there is a proposition ϕ with $FV(\phi) \subseteq \{x_1, \ldots, x_n\}$ such that $P =_{\beta\eta} [\![\phi]\!]$.

3. *If*

$$x_1, \ldots, x_m : \text{term}, p_1 : \text{Prf}\,P_1, \ldots, p_n : \text{Prf}\,P_n \vdash_\Sigma D : \text{Prf}\,Q$$

then there are propositions ϕ_1, ..., ϕ_n, ψ, with $FV(\phi_i), FV(\psi) \subseteq \{x_1, \ldots, x_m\}$, such that

$$P_1 =_{\beta\eta} [\![\phi_1]\!], \ldots, P_n =_{\beta\eta} [\![\phi_n]\!], Q =_{\beta\eta} [\![\psi]\!]$$

and

$$\phi_1, \ldots, \phi_n \vdash \psi$$

Adequacy is always harder to prove than soundness for a logical framework. In the case of ELF, the proof requires weak normalization for the system ELF itself, which is fairly difficult to establish. The contrast with RF is striking:

Proposition 4.4 (Soundness) *In RF:*

1. *For every term t, if $FV(t) = \{x_1, \ldots, x_n\}$, then*

$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma [\![t]\!] : \text{term}$$

2. *For every proposition ϕ, if $FV(\phi) = \{x_1, \ldots, x_n\}$, then*

$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma [\![\phi]\!] : \text{prop}$$

3. *If $\phi_1, \ldots, \phi_n \vdash \psi$, and $FV(\phi_1) \cup \cdots \cup FV(\phi_n) \cup FV(\psi) = \{x_1, \ldots, x_m\}$, then there is a term P such that*

$$x_1, \ldots, x_m : \text{term}, p_1 : \text{Prf}\,[\![\phi_1]\!], \ldots, p_n : \text{Prf}\,[\![\phi_n]\!] \vdash_\Sigma P : \text{Prf}\,[\![\psi]\!]$$

Proposition 4.5 (Adequacy) *In RF:*

1. *If*
$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma T : \text{term}$$
then there is a term t such that $FV(t) \subseteq \{x_1, \ldots, x_n\}$ and $T \equiv [\![t]\!]$.

2. *If*
$$x_1, \ldots, x_n : \text{term} \vdash_\Sigma P : \text{prop}$$
then there is a proposition ϕ with $FV(\phi) \subseteq \{x_1, \ldots, x_n\}$ such that $P \equiv [\![\phi]\!]$.

3. *If*
$$x_1, \ldots, x_m : \text{term}, p_1 : \text{Prf}\, P_1, \ldots, p_n : \text{Prf}\, P_n \vdash_\Sigma D : \text{Prf}\, Q$$
then there are propositions $\phi_1, \ldots, \phi_n, \psi$, with $FV(\phi_i), FV(\psi) \subseteq \{x_1, \ldots, x_m\}$, such that
$$P_1 \equiv [\![\phi_1]\!], \ldots, P_n \equiv [\![\phi_n]\!], Q \equiv [\![\psi]\!]$$
and
$$\phi_1, \ldots, \phi_n \vdash \psi$$

Thus, RF provides a closer representation of the object logic than does ELF; we have syntactic identity in the statement of adequacy, rather than just $\beta\eta$-convertibilty. More importantly, the proof of adequacy for RF is simple; it is a straightforward induction on the derivation of the premise in each case, and relies on no metatheoretic results.

The use of the $\forall I$ rule provides a good illustration of how much more closely RF models the object logic than ELF. Suppose we have a proposition ϕ, possibly with free variable x, and a derivation of ϕ in which x does not occur free. In the logical frameworks, this corresponds to deriving the following two judgements:

$$\begin{array}{lll} \Gamma, x : \text{term} & \vdash_\Sigma & P : \text{prop} \\ \Gamma, x : \text{term} & \vdash_\Sigma & D : \text{Prf}\, P \end{array}$$

for appropriate Γ, P, D. We wish to use $\forall I$ to construct a proof of $\forall x.\phi$, i.e. a term in $\text{Prf}(\forall \lambda x : \text{term}.P)$. In ELF, we must perform the following derivation:

$$\begin{array}{rll} & \vdash_\Sigma & \Gamma \\ \Gamma & \vdash_\Sigma & \forall I : \Pi P : \text{term} \to \text{prop}.(\Pi x : \text{term}.\, \text{Prf}(Px)) \to \text{Prf}(\forall P) \\ \Gamma, x : \text{term} & \vdash_\Sigma & P : \text{prop} \\ \Gamma & \vdash_\Sigma & \lambda x : \text{term}.P : \text{term} \to \text{prop} \\ \Gamma & \vdash_\Sigma & \forall I \lambda x : \text{term}.P : (\Pi x : \text{term}.\, \text{Prf}((\lambda x : \text{term}.P)x)) \to \text{Prf}(\forall \lambda x : \text{term}.P) \\ \Gamma & \vdash_\Sigma & (\Pi x : \text{term}.\, \text{Prf}\, P) \to \text{Prf}(\forall \lambda x : \text{term}.P) : \textbf{type} \\ \Gamma & \vdash_\Sigma & \forall I \lambda x : \text{term}.P : (\Pi x : \text{term}.\, \text{Prf}\, P) \to \text{Prf}(\forall \lambda x : \text{term}.P) \\ \Gamma, x : \text{term} & \vdash_\Sigma & D : \text{Prf}\, P \\ \Gamma & \vdash_\Sigma & \lambda x : \text{term}.D : \Pi x : \text{term}.\, \text{Prf}\, P \\ \Gamma & \vdash_\Sigma & \forall I (\lambda x : \text{term}.P) \lambda x : \text{term}.D : \text{Prf}(\forall \lambda x : \text{term}.P) \end{array}$$

In RF, the following is an instance of one of the rules of deduction:

$$\frac{\begin{array}{c}\Gamma, x : \text{term} \vdash_\Sigma P : \text{prop} \\ \Gamma, x : \text{term} \vdash_\Sigma D : \text{Prf}\, P\end{array}}{\Gamma \vdash_\Sigma \forall I(\lambda x : \text{term}.P)\lambda x : \text{term}.D : \text{Prf}(\forall \lambda x : \text{term}.P)}$$

5 Conclusion

We have presented a system whose derivable judgements are precisely the derivable judgements of ELF in normal form whose subjects are ()-ary objects or types. These are the only judgements with which one is concerned when using the logical framework, so nothing is lost by moving to the new system RF. Much, on the other hand, is gained: RF provides a more faithful representation of the object theories, and has much simpler metatheory.

These advantages of RF over ELF strongly suggest that it would be worth extracting the lambda-free core of other logical frameworks, particularly ones where (say) proving adequacy or giving semantics to the object theory is problematic because of the complexity of the framework's machinery.

References

[1] Herman Geuvers and Erik Barendsen. Some logical and syntactical observations concerning the first-order dependent type system λP. *Math. Struct. in Comp. Science.*, 9:335–359, 1999.

[2] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993. A preliminary version appeared in the *Proceedings of the Symposium on Logic in Computer Science*, pages 194–204, June 1987.