Notes on Logical Frameworks

Robert Harper
IAS

November 29, 2012

1 Introduction

This is a brief summary of a lecture on logical frameworks given at the IAS on November 26, 2012. See the references for technical details.

2 Logic of Judgments

The idea of a logical framework is to codify the *logic of judgments* in which one may *represent* a logical system as a *theory*. The theory determines the basic forms of judgment, the syntactic categories, and the objects that inhabit them.

The sentences of LJ are called *judgments*. The basic forms are those of the represented logic (for example, $\phi\,\mathsf{true}$ expressing that a proposition ϕ is true). There are two non-basic forms that turn out to be closely related:

1. The *hypothetical judgment* $J_1, \ldots, J_n \vdash J$, expressing *entailment*, or *logical consequence*, of J by $J_1, \ldots, J_n$.

2. The *general judgment* $x_1 \in \chi_1, \ldots, x_n \in \chi_n \mid J$, expressing *generality* of J in parameters $x_1, \ldots, x_n$ ranging over *syntactic categories* $\chi_1, \ldots, \chi_n$.

We shall also have need of a *formation judgment* $x_1 \in \chi_1, \ldots, x_n \in \chi_n \mid o \in \chi$ expressing that o is an object of syntactic category χ.

The behavior of the hypothetical judgment is governed by these two fundamental principles:

Reflexivity $J \vdash J$

Transitivity if $J_1, \ldots, J_m \vdash J$ and $J'_1, \ldots, J, \ldots, J'_n \vdash J'$, then $J'_1, \ldots, J_1, \ldots, J_m, \ldots, J'_n \vdash J'$.

Structural entailment includes also these principles:

Weakening If $J_1, \ldots, J_n \vdash J$, then $J_1, \ldots, J_n, , J_{n+1} \vdash J$.

Exchange If $J_1, \ldots, J, , J', \ldots, J_n \vdash J''$, then $J_1, \ldots, J', J, \ldots, J_n \vdash J''$.

Contraction If $J_1, \ldots, , J, J, \ldots, J_n \vdash J'$, then $J_1, \ldots, J, \ldots, J_n \vdash J'$.

Substructural entailment denies one or more of these structural principles.

The behavior of the general judgment is governed by these two fundamental principles:

Variable $x \in \chi \mid x \in \chi$.

Substitution If $x_1 \in \chi_1, \ldots, x \in \chi, \ldots x_n \in \chi \vdash J$ and $x_1 \in \chi_1, \ldots, x_n \in \chi_n \mid o \in \chi$, then $x_1 \in \chi_1, \ldots, x_n \in \chi_n \vdash [o/x]J$. Similarly, if $x_1 \in \chi_1, \ldots, x \in \chi, \ldots x_n \in \chi \mid o' \in \chi'$ and $x_1 \in \chi_1, \ldots, x_n \in \chi_n \mid o \in \chi$, then $x_1 \in \chi_1, \ldots, x_n \in \chi \mid [o/x]o' \in \chi'$.

Here $[o/x]J$ indicates replacement of x in J by o. *Structural* generality includes these additional principles:

Proliferation If $x_1 \in \chi_1, \ldots, x_n \in \chi_n \mid J$, then $x_1 \in \chi_1, \ldots, x_n \in \chi_n, x \in \chi \mid J$.

Permutation If $x_1 \in \chi_1, \ldots, x \in \chi, x' \in \chi', \ldots, x_n \in \chi \mid J$, then $x_1 \in \chi_1, \ldots, x' \in \chi', x \in \chi, \ldots, x_n \in \chi \mid J$.

Duplication If $x_1 \in \chi_1, \ldots, x \in \chi, x \in \chi, \ldots, x_n \in \chi \mid J$, then $x_1 \in \chi_1, \ldots, x \in \chi, \ldots, x_n \in \chi \mid J$.

The similarities between the hypothetical and general judgments are not accidental. They can be consolidated into a single form by introducing *evidence* for judgments and corresponding variables ranging over such evidence so that the entailment $J_1, \ldots, J_n \vdash J$ becomes $\xi_1 \in J_1, \ldots, \xi_n \in J_n \vdash \varepsilon \in J$. With this notation we see that reflexivity corresponds to variable, transitivity to substitution, weakening to proliferation, exchange to permutation, and contraction to duplication.

This informal account of the logic of judgments is vague in a number of respects, most importantly in the definition of substitution (of objects for object variables and of evidence for evidence variables). It is also not quite clear what would constitute a representation of a logic as a theory in the logic of judgments.

3 LF Language

The **LF** language is an attempt to clarify and formalize a logic of judgments in such a way that there is a clear notion of how to present a logical system as a theory, and a clear notion of what it means for such a presentation to be correct.

The syntax of **LF** involves three forms of expressions, *kinds*, K, *families*, A, and *objects*, M. In addition there are *contexts*, Γ, declaring the types of variables and *signatures*, Σ, declaring the types and kinds of constants. These

are specified by a type system with the following forms of judgment:

$$\begin{array}{ll} \Sigma \vdash & \Sigma \equiv \Sigma' \vdash \\ \Gamma \vdash_\Sigma & \Gamma \equiv \Gamma' \vdash_\Sigma \\ \Gamma \vdash_\Sigma K & \Gamma \vdash_\Sigma K \equiv K' \\ \Gamma \vdash_\Sigma A : K & \Gamma \vdash_\Sigma A \equiv A' : K \\ \Gamma \vdash_\Sigma M : A & \Gamma \vdash_\Sigma M \equiv M' : A \end{array}$$

(The exact formulation varies in different presentations.) The left column are the *formation* judgments, the right are the *definitional equivalence* judgments.

There is a basic kind, `type`, classifying the types. The signature defines the *basic type families*, including the *basic types*. The types are closed under dependent function space, $\Pi x{:}A.A'$, and the kinds are closed under dependent function space over a type, $\Pi x{:}A.K$. Definitional equivalence includes the expected β, and usually η, principles.

It is critically important that all judgment forms of **LF** be *analytic*, which means *self-evident*, rather than *synthetic*, which means *requiring evidence*. The purpose of **LF** is to formalize the forms of evidence for a judgment in some logic, and we wish to avoid the infinite regress of requiring evidence for the judgments of the framework itself. The represented logic may have synthetic or analytic judgment forms, and this influences the means by which a logic may be represented in **LF**.

Please see the references for the precise definition of the **LF** type theory (and bear in mind that there are various forms that differ slightly in technical detail for various reasons).

4 Synthetic Presentations

One method for representing a logic in **LF** is to treat all judgments of the object logic as synthetic, uninterpreted assertions. The goal of the encoding is simply to capture the rules of the logic, whatever they may be, without regard to their meaning, if any. So, for example, it is perfectly sensible to represent an inconsistent logic in **LF** without any difficulties.

The organizing principle for the synthetic representation is the *judgments as types* principle. The idea is that each judgment of the represented logic is encoded as a type family in **LF**, and the rules for deriving those judgments are encoded as constants of appropriate type. The correctness of the encoding is expressed by an *adequacy theorem* which says, roughly, that there is a compositional (aka natural, or commuting with substitution) bijection between the deductive apparatus of the object logic and the canonical forms (that is, the long $\beta\eta$-normal forms) of specified types in **LF**, relative to a signature defining these constants and a context specifying the variables that may be involved in the encoding.

An example is the synthetic encoding of Gödel's **T**, which is given as follows.[1]

[1] As a TEXnical convenience I will use the notation of the Twelf implementation of **LF** in

First, the abstract syntax, using the technique of *higher-order abstract syntax* to encode variable binding:

```
tp : type.
nat : tp.
arr : tp -> tp -> tp.

tm : type.
zero : tm.
succ : tm -> tm.
rec : tm -> (tm -> tm) -> tm -> tm.
app : tm -> tm -> tm.
lam : (tm -> tm) -> tm.
```

The adequacy is expressed as follows. First, there is a bijection $\ulcorner \tau \urcorner$ between types τ of **T** and canonical forms of **LF** of type `tp`. Second, there is a bijection $\ulcorner e \urcorner$ between terms e of **T** with free variables $x_1, \ldots, x_n$ and canonical forms of **LF** of type `tm` in context x_1 : `tm`, $\ldots, x_n$: `tm`. Moreover, this bijection is *compositional* in that it commutes with substitution for the free variables, which is to say that $\ulcorner [e/x]e' \urcorner = [\ulcorner e \urcorner / x] \ulcorner e' \urcorner$. (Compositionality does not arise with the adequacy of the representation of types for the simple reason that we do not consider variable types in **T**, whereas we do, of course, consider variable terms.)

Second, the typing judgment of **T**, using the judgments as types principle:

```
of : tm -> tp -> type.
of/zero : of zero nat.
of/succ : e:nat of (succ e) nat.
of/rec :
  {t:tp} {e0:tm} {e1:tm->tm} {e:tm}
  of e0 t -> ({x:tm} of x t -> of (e1 x) t) -> of e nat ->
  of (rec e0  e1 e) t.
of/lam :
  {t1:tp} {t2:tp} {e:tm->tm}
  ({x:tm} of x t1 -> of (e x) t2) -> of (lam e) (arr t1 t2).
of/app :
  {t1:tp} {t2:tp} {e1:tm} {e2:tm}
  of e1 (arr t1 t2) -> of e1 t1 -> of (app e1 e2) t2.
```

The adequacy of the encoding is expressed by giving a compositional bijection $\ulcorner \nabla \urcorner$ between derivations ∇ of $x_1 ; \tau_1, \ldots, x_n : \tau_n \vdash e : \tau$ of **T** and canonical forms of **LF** with typings of the form

$$x_1 : \mathtt{tm}, \xi_1 : \mathtt{of}\ x_1\ \ulcorner \tau_1 \urcorner, \ldots, x_n : \mathtt{tm}, \xi_n : \mathtt{of}\ x_n\ \ulcorner \tau_n \urcorner \vdash \ulcorner \nabla \urcorner : \mathtt{of}\ \ulcorner e \urcorner\ \ulcorner \tau \urcorner.$$

which Π's are represented using curly braces, and λ's using square brackets. For the sake of clarity I will eschew the many conveniences provided by Twelf that would allow me to omit many quantifiers.

As before, compositionality means that $\ulcorner[\nabla/\xi]\nabla'\urcorner = [\ulcorner\nabla\urcorner/\xi]\ulcorner\nabla'\urcorner$, where the notation on the left denotes the obvious composition of derivations of typing judgments in **T**, which are not ordinarily made explicit in informal accounts of the system.

Third, the conversion judgment of **T** is encoded as follows:[2]

```
conv : tm -> tm -> type.
refl : {e:tm} conv e e.
sym : {e1:tm} {e2:tm} conv e1 e2 -> conv e2 e1.
beta : {f:tm->tm} {e:tm} conv (app (lam f) e) (f e).
```

The advantage of higher-order abstract syntax becomes apparent in the `beta` rule, which simply applies the body of the λ to an argument to effect the substitution. The adequacy of the encoding of conversion is stated similarly to the adequacy of the encoding of typing.

5 Analytic Presentations

The synthetic representation of **T** is not faithful to the intended meaning of its judgments. In particular the typing and definitional equivalence judgments of **T** are analytic, and should be represented as such in **LF**. Representing them synthetically misses the intended meaning, and is to this extent inaccurate.

The alternative is to use an *analytic* representation in which typing **T** is represented directly as typing of **LF**, and conversion of **T** is represented directly by definitional equivalence of **LF**. This means that the presentation of **T** in this form consists of a signature of constants together with an equational theory governing the behavior of these constants.

Here is a sketch of the signature of the analytic encoding of **T**. First, the syntax, which represents *only* the well-typed terms:

```
tp : type.
nat : tp.
arr : tp->tp->tp.

el : tp -> type.
zero : el nat.
succ : el nat -> el nat.
rec : {t:tp} el t -> (el t -> el t) -> el nat -> el t.
lam : {t1:tp} {t2:tp} (el t1 -> el t2) -> el (arr t1 t2).
app : {t1:tp} {t2:tp} el (arr t1 t2) -> el t1 -> el t2.
```

Notice that the "polymorphism" of the language is handled at the framework level.

There is an obvious bijection between types and terms of **T** and canonical forms of suitable type in **LF**. For example, $x : \tau_1 \vdash e : \tau_2$ in **T** corresponds

[2]Only a few key rules are given here.

to $x : \mathtt{el}\ulcorner\tau_1\urcorner \vdash \ulcorner e\urcorner : \mathtt{el}\ulcorner\tau_2\urcorner$ in **LF**, and this correspondence commutes with (well-typed) substitution.

Second, definitional equivalence is given by equational axioms such as the following:

$$\Gamma \vdash \mathtt{app}\ulcorner\tau_1\urcorner\ulcorner\tau_2\urcorner(\mathtt{lam}\, F)\, M \equiv F\, M : \mathtt{el}\ulcorner\tau_2\urcorner$$

Similar axioms govern the recursor in the usual manner. The foregoing axiom presupposes the adequacy of the encoding by quantifying over arbitrary F and M of the appropriate type. One might also state the axiom in more explicit form by requiring M to be $\ulcorner e\urcorner$ for some e of the domain type τ_1, and requiring F to be $\lambda x{:}\ulcorner\tau_1\urcorner.\ulcorner e\urcorner$, where $x : \tau_1 \vdash e : \tau_2$ in **T**. The axiom then states directly the β principle for **T**.

It is a matter of semantics to understand that this encoding of **T** in **LF** is adequate. In particular we must ensure that definitional equivalence of **LF** remains analytic under this representation to ensure that we have faithfully encoded the object logic. From an implementation point of view this means that we must develop methods for checking conversion of terms in **T** via their representation in **LF**, and this process must be repeated for each represented logical system.

References

[1] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum, editors. *Handbook of Logic in Computer Science*, volume 5. Oxford University Press, 2001.

[2] Robert Harper. *Practical Foundations for Programming Languages.* Cambridge University Press, 2012.

[3] Robert Harper, Furio Honsell, and Gordon D. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1993.

[4] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the lf type theory. *ACM Trans. Comput. Log.*, 6(1):61–101, 2005.

[5] Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Notre Dame Journal of Philosophical Logic*, 1(1):11–60, 1996.

[6] Bengt Nordström. *Martin-Löf's Type Theory*, chapter 1. Volume 5 of Abramsky et al. [1], 2001.