Syntactic LF in Canonical Form

Robert Harper

January 13, 2022

1 Introduction

The definition of the LF logical framework given in Harper et al. (1993) is in the familiar *declarative* form that defines simultaneously the valid kinds, families, and objects,[1] and equivalent kinds, families, and objects. This formulation is declarative in that the classification of a family by a kind or an object by a type is posited to be invariant under equivalence of the classifier. This requirement has two consequences:

1. The rules are no longer syntax-directed, there being two rules for each construct, its principal rule and the invariance rule.

2. A type checker must include an equivalence checker so as to ensure invariance.

It is possible to implement LF in this manner, without significant difficulty. The main idea is to integrate equivalence checking into the typing rules, thereby recovering syntax-directedness, and to implement an equivalence checker for families and objects.

The adequacy of an LF encoding relates the canonical forms of certain LF types to the syntactic objects of the formalism in question. Stating this precisely requires a precise definition of the canonical forms of a type, which are, informally, the long $\beta\eta$-normal forms of its elements. It is a small matter to extend the equivalence checking algorithm mentioned above to compute a witness to each equivalence, a canonical form equivalent to the compared terms. And that suffices for adequacy.

But then, if the encodings are all given by canonical forms, why are non-canonical forms considered at all? If the canonical forms of LF could be defined directly, then such a definition would suffice for adequacy, and, moreover, simplify implementation by eliminating the need for equivalence checking entirely. The snag in this argument is that defining the canonical forms of a type or kind requires substitution of the argument of an application into the codomain of a dependent function type. Unfortunately, canonical forms are not stable under substitution, at least not as it is normally defined. For example, an object of the form $x\,M$ is canonical, as long as M is, and so is $\lambda x.N$, if N is, but $(\lambda x.N)\,M$ is certainly not canonical.

But all is not lost. The trick is to replace the familiar definition of substitution, which works on raw terms, by *hereditary*, or *canonizing*, *substution*, which makes use of types (Harper and Licata, 2007). Roughly speaking, hereditary substitution re-canonizes on the fly when faced with the situation just described: the result of the substitution is not $(\lambda x.N)\,M$, but rather (the canonization of) $[M/x]N$, which is in canonical form. To make this work requires additional information beyond

[1] Classes, sorts, and objects in the terminology of Harper (2021)

the mere terms themselves, namely the type of the substituting term, M, which is also the type of x in N. Using this information it is then possible to show that hereditary substitution is well-defined, and thus to justify the formulation of LF in canonical form.

2 LF in Canonical Form

Given an appropriate definition of substitution, it is easy to give syntax-directed rules defining the following judgments:

- Canonical kinds: $\Gamma \vdash_\Sigma K \downarrow$
- Canonical families of canonical kind K: $\Gamma \vdash_\Sigma A \downarrow K$
- Neutral families of canonical kind K: $\Gamma \vdash_\Sigma A \uparrow K$
- Canonical objects of canonical type A: $\Gamma \vdash_\Sigma M \downarrow A$
- Neutral objects of canonical type A: $\Gamma \vdash_\Sigma M \uparrow A$

The context Γ consists of neutral assumptions $x \uparrow A$, where A is a canonical type, and, similarly, the signature Σ associates canonical kinds and types to constants.

These judgments are inductively defined by the rules in Figures 1, 2, and 3. These rules make use of hereditary substitution, which is considered in the next section, indexed by the *spine* of a type, written A°. These will both be defined in the next section.

3 Hereditary Substitution

The rules defining canonical LF given in the preceding section make use of hereditary substitution of a canonical object for a variable in a canonical kind or type. Because objects may occur in kinds and types, this necessitates also defining hereditary substitution of a canonical object for a variable in another canonical object. Because canonical types can be neutral families, and canonical objects can be neutral objects, it is also necessary to define substitution of neutral objects into canonical kinds, neutral families and neutral objects.

Thus, altogether we have the following forms of hereditary substitution, differentiated by whether the target is canonical or neutral:

- $[M/x]^*_{\alpha^\circ} K$, object into kind;
- $[M/x]^*_{\alpha^\circ} A$, object into family;
- $[M/x]^*_{\alpha^\circ} N$, object into object.

The superscript asterisk can be either c or n. For kinds, only c is possible, there being no neutral kinds. For families and objects, both c and n are possible, for a total of five forms of substitution.

The subscript α is the *spine*, or *dependency erasure*, of the type of M, defined as follows:

$$a^\circ \triangleq a$$
$$\mathsf{Ap}(A, M)^\circ \triangleq A^\circ$$
$$(x : A_1 \to A_2)^\circ \triangleq A_1^\circ \to A_2^\circ$$

In effect all instances of a family are regarded as base types differentiated only by the head constant, and not the arguments. Having removed objects from types, the dependent function type degenerates to a simple function type.

Hereditary substitution is inductively defined by the rules given in Figures 4, 5, and 6.

Lemma 1 (Substitution is Well-Defined)**.** *Suppose that* $\Gamma \vdash_\Sigma M \downarrow A$.

1. *If* $\Gamma, x \uparrow A \vdash_\Sigma N \downarrow B$, *then there exists* N' *such that* $[M/x]^{\mathsf{c}}_{A^\circ} N = N'$.
2. *If* $\Gamma, x \uparrow A \vdash_\Sigma B \downarrow K$, *then there exists* A' *such that* $[M/x]^{\mathsf{c}}_{A^\circ} B = B'$.
3. *If* $\Gamma, x \uparrow A \vdash_\Sigma K \downarrow$, *then there exists* K' *such that* $[M/x]^{\mathsf{c}}_{A^\circ} K = K'$.

Proof Sketch. For the induction these statements must be augmented with corresponding statements governing neutral target families, and neutral source and target objects.

The critical case is rule ApOC, in which substitution into the neutral principal argument results in a canonical form, a λ-abstraction. The third premise of the rule invokes hereditary substitution on the body of that function, at the spine of its domain type. The critical point is that A_1° must be structurally smaller than A°, even though P can be arbitrarily larger than $\mathsf{ap}(N, M_1)$.

Thus, a lexicographic induction with principal index being the spine of the type of the object and secondary induction being the structure of the target, suffices to ensure that hereditary substitution is well-defined. □

Exercise 1. *Complete the proof of the well-definedness of hereditary substitution.*

Exercise 2. *Extend the canonical LF to account for unit and dependent product types, including the definition of hereditary substitution.*

$$\textsc{TyK}\ \frac{}{\Gamma \vdash_\Sigma \mathsf{sort} \downarrow} \qquad \textsc{PiK}\ \frac{\Gamma \vdash_\Sigma A_1 \downarrow \mathsf{sort} \qquad \Gamma, x_1 \uparrow A_1 \vdash_\Sigma K_2 \downarrow}{\Gamma \vdash_\Sigma x_1 : A_1 \to K_2 \downarrow}$$

Figure 1: Canonical Kinds

$$\textsc{CoF}\ \frac{}{\Gamma \vdash_{\Sigma, a \uparrow K} a \uparrow K} \qquad \textsc{ApF}\ \frac{\Gamma \vdash_\Sigma A \uparrow x_1 : A_1 \to K_2 \qquad \Gamma \vdash_\Sigma M_1 \downarrow A_1}{\Gamma \vdash_\Sigma \mathsf{Ap}(A, M_1) \uparrow [M_1/x_1]^{\mathsf{c}}_{A_1^\circ} K_2}$$

$$\textsc{NeF}\ \frac{\Gamma \vdash_\Sigma A \uparrow \mathsf{sort}}{\Gamma \vdash_\Sigma A \downarrow \mathsf{sort}} \qquad \textsc{PiF}\ \frac{\Gamma \vdash_\Sigma A_1 \downarrow \mathsf{sort} \qquad \Gamma, x_1 \uparrow A_1 \vdash_\Sigma A_2 \downarrow \mathsf{sort}}{\Gamma \vdash_\Sigma x_1 : A_1 \to A_2 \downarrow \mathsf{sort}}$$

Figure 2: Neutral and Canonical Families

$$\textsc{VaO}\ \frac{}{\Gamma, x \uparrow A \vdash_\Sigma x \uparrow A} \qquad \textsc{CoO}\ \frac{}{\Gamma \vdash_{\Sigma, c \uparrow A} c \uparrow K}$$

$$\textsc{ApO}\ \frac{\Gamma \vdash_\Sigma M \uparrow x_1 : A_1 \to A_2 \qquad \Gamma \vdash_\Sigma M_1 \downarrow A_1}{\Gamma \vdash_\Sigma \mathsf{ap}(M, M_1) \uparrow [M_1/x_1]^{\mathsf{c}}_{A_1^\circ} A_2} \qquad \textsc{NeO}\ \frac{\Gamma \vdash_\Sigma M \uparrow A \qquad (A \neq x : A_1 \to A_2)}{\Gamma \vdash_\Sigma M \downarrow A}$$

$$\textsc{LaO}\ \frac{\Gamma \vdash_\Sigma A_1 \downarrow \mathsf{sort} \qquad \Gamma, x_1 \uparrow A_1 \vdash_\Sigma M_2 \downarrow A_2}{\Gamma \vdash_\Sigma \lambda_{A_1}(x_1.M_2) \downarrow x_1 : A_1 \to A_2}$$

Figure 3: Neutral and Canonical Objects

References

Robert Harper. An equational logical framework for type theories. (Unpublished manuscript.), March 2021. URL `https://www.cs.cmu.edu/~rwh/courses/chtt/notes/slf.pdf`.

Robert Harper and Daniel R. Licata. Mechanizing metatheory in a logical framework. *Journal of Functional Programming*, 17(4-5):613–673, 2007. doi: 10.1017/S0956796807006430.

Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40:194–204, 1993.

$$\textsc{TyK}\ \frac{}{[M/x]^{\mathsf{c}}_{A^\circ}\mathsf{sort} = \mathsf{sort}} \qquad \textsc{PiK}\ \frac{[M/x]^{\mathsf{c}}_{A^\circ}A_1 = A_1' \qquad [M/x]^{\mathsf{c}}_{A^\circ}K_2 = K_2'}{[M/x]^{\mathsf{c}}_{A^\circ}x_1 : A_1 \to K_2 = x_1 : A_1' \to K_2'}$$

Figure 4: Hereditary Substitution into Kinds

$$\textsc{CoF}\ \frac{}{[M/x]^{\mathsf{n}}_{A^\circ}a = a} \qquad \textsc{ApF}\ \frac{[M/x]^{\mathsf{n}}_{A^\circ}B = B' \qquad [M/x]^{\mathsf{c}}_{A^\circ}M_1 = M_1'}{[M/x]^{\mathsf{n}}_{A^\circ}\mathsf{Ap}(B, M_1) = \mathsf{Ap}(B', M_1')}$$

$$\textsc{NeF}\ \frac{[M/x]^{\mathsf{n}}_{A^\circ}B = B'}{[M/x]^{\mathsf{c}}_{A^\circ}B = B'} \qquad \textsc{PiF}\ \frac{[M/x]^{\mathsf{c}}_{A^\circ}A_1 = A_1' \qquad [M/x]^{\mathsf{c}}_{A^\circ}A_2 = A_2'}{[M/x]^{\mathsf{c}}_{A^\circ}x_1 : A_1 \to A_2 = x_1 : A_1' \to A_2'}$$

Figure 5: Hereditary Substitution into Families

$$\textsc{VaOY}\ \frac{}{[M/x]^{\mathsf{n}}_{A^\circ}x = M} \qquad \textsc{VaON}\ \frac{(x \neq y)}{[M/x]^{\mathsf{n}}_{A^\circ}y = y} \qquad \textsc{CoO}\ \frac{}{[M/x]^{\mathsf{n}}_{A^\circ}c = c}$$

$$\textsc{NeO}\ \frac{[M/x]^{\mathsf{n}}_{A^\circ}N = N'}{[M/x]^{\mathsf{c}}_{A^\circ}N = N'} \qquad \textsc{LaO}\ \frac{[M/x]^{\mathsf{c}}_{A^\circ}A_1 = A_1' \qquad [M/x]^{\mathsf{c}}_{A^\circ}M_2 = M_2'}{[M/x]^{\mathsf{c}}_{A^\circ}\lambda_{A_1}(x_1.M_2) = \lambda_{A_1'}(x_1.M_2')}$$

$$\textsc{ApOC}\ \frac{[M/x]^{\mathsf{n}}_{A^\circ}N = \lambda_{A_1}(x_1.P) \qquad [M/x]^{\mathsf{c}}_{A^\circ}M_1 = M_1' \qquad [M_1/x_1]^{\mathsf{c}}_{A_1^\circ}P = P'}{[M/x]^{\mathsf{n}}_{A^\circ}\mathsf{ap}(N, M_1) = P'}$$

$$\textsc{ApON}\ \frac{[M/x]^{\mathsf{n}}_{A^\circ}N = N' \neq \lambda \qquad [M/x]^{\mathsf{c}}_{A^\circ}M_1 = M_1'}{[M/x]^{\mathsf{n}}_{A^\circ}\mathsf{ap}(N, M_1) = \mathsf{ap}(N', M_1')}$$

Figure 6: Hereditary Substitution into Objects