

# An Extension of LF: Reasoning About Stateful Deductive Systems with Connectives from Hybrid Logic

Maja Frydrychowicz

19 April 2008

## Abstract

The Hybrid Logical Framework (HLF) is intended to build upon the achievements of Linear LF (LLF) in mechanizing the metatheory of stateful deductive systems. With the help of a simple translation of the linear implication operator ( $\multimap$ ), it has been shown that LLF is in fact embedded in HLF. With the intention of gaining a better understanding of the resource labels and hybrid connectives introduced in HLF, we apply this result to several examples from an LLF case study of Mini-ML with References (MLR).

## 1 Introduction

Many interesting programming language features, such as references or pattern matching, can be easily modelled with linear logic. The restricted hypothetical context of linear logic, in which contraction and weakening are forbidden, offers an elegant explanation of stateful systems: restricted hypotheses correspond to consumable resources. Although it is possible to model stateful systems in LF, the resulting encodings are quite messy. As a result, mechanizing linear metatheory is a highly desirable extension to the logical framework LF [2].

In the past, a highly promising proposal was that of Linear LF (LLF) [1]. This extension of LF uses two kinds of hypothetical contexts, linear and unrestricted, as well as several linear operators, to encode stateful systems. The authors published a case study of Mini-ML extended with references (MLR), in which the static and dynamic semantics are fully encoded in LLF. However, LLF cannot be used to faithfully formalize the statements of certain linear meta-theorems due to limitations in its reasoning about the interaction between distinct restricted hypothetical contexts.

In response to this theoretical gap, Jason Reed proposes the Hybrid Logical Framework (HLF), which extends LF with a notion of worlds, referred to as “labels”, or “resources”, and several connectives from hybrid logic [3, 2]. In the sections that follow, we review the major theoretical contributions associated with HLF and discuss the benefits afforded by hybrid logical connectives in comparison with LLF. With the help of several examples selected from the LLF case study of MLR, to which we apply a simple translation that helped prove the embedding of LLF in HLF, we build an intuitive understanding of how labels in HLF are used to model state.

$$\begin{array}{c}
\frac{\Gamma, \alpha : label \vdash M : B[p]}{\Gamma \vdash M : \forall \alpha. B[p]} \quad \frac{\Gamma \vdash M : \forall \alpha. B[p] \quad \Gamma \vdash q : label}{\Gamma \vdash : (\{q/\alpha\}B)[p]} \\
\\
\frac{\Gamma \vdash M : (\{p/\alpha\}B)[p]}{\Gamma \vdash M : \downarrow \alpha. B[p]} \quad \frac{\Gamma \vdash M : \downarrow \alpha. B[p]}{\Gamma \vdash M : (\{p/\alpha\}B)[p]} \\
\\
\frac{\Gamma \vdash M : A[p]}{\Gamma \vdash M : @_p A[q]} \quad \frac{\Gamma \vdash M : @_p A[q]}{\Gamma \vdash M : A[p]}
\end{array}$$

Figure 1: HLF typing rules for new hybrid type operators.

## 2 HLF: Selected Contributions

Hybrid LF is inspired by the reduction of substructural properties of context to the algebraic properties of resource labels attached to deductions [3]. Every hypothesis in the HLF context is unrestricted unless it is associated with a label  $p$ , in which case it is considered linear. Through the addition of connectives from hybrid logic, sets of restricted hypotheses may be manipulated directly.

Thus, HLF extends LF with a category of expressions that describes these resource labels. It also extends LF's type families with three hybrid type operators. LF, HLF and LLF have the same term language.

Labels	$p, q, r ::=$	$\alpha \mid p_1 * p_2 \mid \epsilon$
Terms	$M ::=$	$\lambda x. M \mid MN \mid x \mid c$
Type Families	$A ::=$	$\Pi x : A. B \mid AMa \mid \forall \alpha. A \mid \downarrow \alpha. A \mid @_p A$
Contexts	$\Gamma ::=$	$\cdot \mid \Gamma, x : A \mid \Gamma, \alpha : label$

Labels may be combined with  $*$ , and the label  $\epsilon$  represents empty resource use, in that it is associated with unrestricted hypotheses. HLF's typing judgment is read as 'in context  $\Gamma$ , the term  $M$  has type  $A$  at world  $p$ ':

$$\Gamma \vdash M : A[p]$$

The new typing rules are listed in Figure 1. Binding ( $\downarrow \alpha$ ) takes the current label and binds it to  $\alpha$  in the current term being checked. The transfer operator ( $@_p$ ) replaces the current label with the label  $p$ , thus assigning the consumption of resource  $p$  to the term enclosed by  $@_p$ . Therefore, a variable  $x$  of type  $A$  that is only available through the consumption of resource  $p$  appears in context  $\Gamma$  as  $x : @_p A$ . The hybrid operators introduced also include straightforward quantification over labels ( $\forall$ ).

Our upcoming discussion of hybrid type operators in HLF relies heavily on the following result. Linear implication ( $\multimap$ ), which is the only connective from LLF that HLF lacks, can be encoded with hybrid operators as follows:

$$A \multimap B \equiv \downarrow \alpha. \forall \beta. ((@_\beta A) \rightarrow (@_{\alpha * \beta} B))$$

This translation was used to prove the embedding of LLF in HLF. From this result, it follows that linear implication has the following intuitive interpretation:  $A \multimap B$  is achievable

by consuming resources  $\alpha$  if, for any resource  $\beta$  of type  $A$ , we can produce  $B$  by consuming resources  $\alpha * \beta$ .

Now, in order to translate an LLF encoding to HLF, simply take every subformula of the form  $A \multimap B$ , replace it with the above hybrid logic expression, and set the entire formula with the empty label  $\epsilon$ . For example,  $\Pi x.(A \multimap B)$  in LLF will be  $\Pi x.(\downarrow \alpha.\forall\beta.((@_{\beta}A) \rightarrow (@_{\alpha*\beta}B)))[\epsilon]$  in HLF.

### 3 Examples with Mini-ML with References

In this section we apply Reed’s embedding of LLF into HLF in order to become better acquainted with hybrid logical connectives. The MLR case study includes a full LLF encoding of continuation-based evaluation rules [1]. We consider two of these evaluation rules, shown in Figure 2, which access the memory store directly, and thus make use of linear assumptions. Using Reed’s translation of linear implication ( $\multimap$ ), as described in Section 2, we rewrite the given rules using HLF syntax and discuss the role of labelled resources in managing linear assumptions.

Since HLF is claimed to succeed where LLF fails, that is, in encoding the statements of linear metatheorems, the examples of greatest interest are of course those pertaining to the metatheory of MLR. However, it is preferable to first gain familiarity with the role played by hybrid connectives in HLF. In doing so, we highlight the benefits of being able to manipulate linear hypotheses directly. It is for this reason that we examine evaluation rules instead of metatheory.

#### Dynamic Semantics for MLR

MLR extends Mini-ML with a memory store  $S$  and imperative instructions. The store is a collection of cells  $c$  containing values. Mini-ML syntax is extended with familiar operations for manipulating individual cells:

$$\begin{array}{ll} \text{Expressions} & e ::= \dots \mid c \mid \mathbf{ref} \ e \mid !e \mid e_1 := e_2 \mid e_1; e_2 \\ \text{Stores} & S ::= \cdot \mid S, c = v \end{array}$$

The reductions occurring during the execution of an MLR program are presented as continuation-based evaluation rules with the following judgment:

$$S \triangleright K \vdash i \hookrightarrow a$$

In this judgment,  $S$  is a memory store,  $K$  is a continuation,  $i$  is an instruction and  $a$  is a final answer for the evaluation. The continuation  $K$  corresponds to a stack of expressions waiting to be evaluated, and is not of any importance in our discussion of linear data. Instructions  $i$  and answers  $a$  are part of an additional syntactic layer meant to distinguish expressions awaiting evaluation, and may be thought of as ordinary terms for our purposes.

The LLF representation of evaluation rules is based on the following type families.

```
ev          : cont -> instr -> answer -> type.
contains   : cell -> exp -> type.
```

$$\frac{S \triangleright K \vdash \mathbf{return} \ z \hookrightarrow a}{S \triangleright K \vdash \mathbf{eval} \ z \hookrightarrow a} \mathbf{ev\_z} \quad \frac{(S, c = v) \triangleright K \vdash \mathbf{return} \ c \hookrightarrow a}{S \triangleright K \vdash \mathbf{ref}^* v \hookrightarrow \mathbf{new} \ c.a} \mathbf{ev.ref}^*$$

$$\frac{(S, c = v) \triangleright K \vdash \mathbf{return} \ \langle \rangle \hookrightarrow a}{(S, c = v') \triangleright K \vdash c :=_2^* v \hookrightarrow a} \mathbf{ev.assign}_2^*$$

Figure 2: A selection of MLR evaluation rules.

The detailed definitions of these type families correspond intuitively to the unformalized theory of MLR, and are available in the report on LLF [1]. We will summarize them informally for the sake of brevity.

The LLF base type  $\mathbf{ev}$  represents the judgment  $S \triangleright K \vdash i \hookrightarrow a$ . The memory store  $S = (c_1 = v_1, \dots, c_n = v_n)$  is represented in a distributed fashion in the context of LLF. It follows from Reed’s embedding result that the memory store may be represented analogously in HLF. Cell content  $c = v$  is represented via the **contains** base type. Cells  $c$  in  $S$  are allowed to be mentioned many times during evaluation, so they are represented as intuitionistic assumptions. On the other hand, the values  $v$  contained in cells are meant to be updated destructively according to MLR semantics, so each  $v$  is represented with a linear hypothesis of the form  $\mathbf{h} \ ;_i \ \mathbf{contains} \ \mathbf{c} \ \mathbf{v}$ .

In HLF, all context variables are unrestricted unless otherwise specified. In the examples presented below, you will see that cell values are restricted with the help of the hybrid “shifting” type operator  $@_p$ , which will enforce that the variable representing a value stored in a cell is only available via the consumption of some resource  $p$ .

### Example 0: $\mathbf{ev\_z}$

As a preliminary example, we first consider the rule  $\mathbf{ev\_z}$ . This rule does not access the memory store directly, but it illustrates the basic use of resource consumption in HLF.

The LLF encoding of  $\mathbf{ev\_z}$  is as follows:

$$\mathbf{ev\_z}: \quad \mathbf{IIK.IIA.}(\mathbf{ev} \ K \ (\mathbf{return} \ z) \ A \ \multimap \ \mathbf{ev} \ K \ (\mathbf{eval} \ z) \ A)$$

We use linear implication in this declaration to ensure that both the antecedent and the consequent have access to the same linear assumption, that is, the same memory store. Intuitionistic implication would have erased all linear assumptions for the consequent. This basic pattern, intended to pass on existing linear assumptions, is repeated in all subsequent examples. Now applying Reed’s embedding translation to the above, we get the following declaration in HLF:

$$\mathbf{H.ev\_z}: \quad \mathbf{IIK.IIA.}(\downarrow \alpha. \forall \beta. (@_{\beta}(\mathbf{ev} \ K \ (\mathbf{return} \ z) \ A) \rightarrow @_{\alpha * \beta}(\mathbf{ev} \ K \ (\mathbf{eval} \ z) \ A))[\epsilon])$$

Since the whole type is unrestricted ( $\epsilon$ ),  $\alpha$  will be instantiated as  $\epsilon$ , empty. Therefore the antecedent depends on resources  $\beta$  and the consequent depends on resources  $\beta$ . This corresponds nicely to our intuition: both sides of the implication have access to the same linear assumptions (both consume the same resources  $\beta$ ), which means that no new linear assumptions are introduced.

### Example 1: $\text{ev\_ref}^*$

The evaluation rule  $\text{ev\_ref}^*$  creates a new location  $c$  in the store and initializes it with  $v$ . In the encoding, the new cell is intuitionistically assumed, whereas its contents, expressed as  $\text{contains } c \ V$ , is assumed linearly so that it may be destroyed later in evaluation.

Here's the LLF encoding:

$$\begin{aligned} \text{ev\_ref}^*: \quad & \Pi K. \Pi A. \Pi V. (\Pi c: \text{cell}. ((\text{contains } c \ V) \\ & \quad \multimap \text{ev } K \ (\text{return } (\text{rf } c)) \ A) \\ & \quad \multimap \text{ev } K \ (\text{ref}^* \ V) \ (\text{new } (\lambda c: \text{cell}. A \ c))) \end{aligned}$$

In accordance with the introduction rules for  $\downarrow$  and  $@_p$ , the corresponding HLF encoding represents the new cell value with resource  $\beta_1$ , and all previous existing values in the store with resource  $\beta$ . Therefore  $\text{ev } K \ (\text{return } (\text{rf } c)) \ A$  depends on resources  $\beta_1$  and  $\beta$ , whereas  $\text{ev } K \ (\text{ref}^* \ V) \ (\text{new } (\lambda c: \text{cell}. A \ c))$  depends only on the previous state of the store, resource  $\beta$ :

$$\begin{aligned} \text{H\_ev\_ref}^*: \quad & \Pi K. \Pi A. \Pi V \downarrow \alpha. \forall \beta. @_{\beta} (\Pi c: \text{cell}. (\downarrow \alpha_1 \forall \beta_1. (@_{\beta_1} (\text{contains } c \ V) \\ & \quad \rightarrow @_{\alpha_1, \beta_1} (\text{ev } K \ (\text{return } (\text{rf } c)) \ A))) \\ & \rightarrow @_{\alpha * \beta} (\text{ev } K \ (\text{ref}^* \ V) \ (\text{new } (\lambda c: \text{cell}. A \ c))) [\epsilon] \end{aligned}$$

### Example 2: $\text{ev\_assign}_2^*$

The rule  $\text{ev\_assign}_2^*$  takes care of the destructive update of the contents of a memory cell  $C$ . First the old value is retrieved by  $\text{contains } C \ V'$ . Value retrieval amounts to the consumption of the associated linear assumption. The remaining values in the store are accessible to the rule's premise, by virtue of the linear implication connective between premise and conclusion. The term  $\text{contains } C \ V$  assigns a new consumable value to the cell and is added to the LLF context as a new linear hypothesis.

The rule is encoded in LLF as follows:

$$\begin{aligned} \text{ev\_assign}_2^*: \quad & \Pi K. \Pi A. \Pi V. \Pi C. ((\text{contains } C \ V) \multimap \text{ev } K \ (\text{return } \text{unit}) \ A)) \\ & \quad \multimap ((\text{contains } C \ V') \multimap \text{ev } K \ \text{assign}_2^* \ (\text{rf } C) \ V) \ A) \end{aligned}$$

The corresponding HLF encoding appears below. In this case, the  $\Pi$  quantifiers are omitted to save space. Intuitively,  $c :=_2^* v \hookrightarrow a$  in the conclusion of  $\text{ev\_assign}_2^*$  depends on the resources restricting  $v'$ , the old contents of cell  $c$ , represented by  $\beta_1$ , as well as the resources restricting the contents of the rest of the memory store ( $\beta$ ). In contrast, the premise,  $\text{return } \langle \rangle \hookrightarrow a$ , depends on the new resource  $\beta_2$ , which corresponds to the updated cell value  $v$ , and on the rest of the store ( $\beta$ ).

$\text{H\_ev\_assign}_2^*$ :

$$\begin{aligned} & \downarrow \alpha. \forall \beta. (@_{\beta} ( \downarrow \alpha_2 \forall \beta_2. (@_{\beta_2} (\text{contains } \mathbf{C} \ \mathbf{V}) \rightarrow @_{\alpha_2, \beta_2} (\text{ev } \mathbf{K} \ (\text{return unit}) \ \mathbf{A})))))) \\ & \rightarrow @_{\alpha, \beta} ( \downarrow \alpha_1. \forall \beta_1. (@_{\beta_1} (((\text{contains } \mathbf{C} \ \mathbf{V}') \rightarrow @_{\alpha_1, \beta_1} (\text{ev } \mathbf{K} \ \text{assign}_2^* \ (\text{rf } \mathbf{C}) \ \mathbf{V}) \ \mathbf{A})))))) \end{aligned}$$

Note that the premise does not depend on resource  $\beta_1$ . That resource has been consumed; the associated cell contents  $v'$  has been destroyed.

## 4 Conclusion

Our choice of examples illustrates how HLF uses labelled resources to represent restricted hypotheses. The resources have a nice, intuitive correspondence to the consumable cell contents operated upon by MLR evaluation rules. The hybrid logical connectives found in HLF offer substantial flexibility for manipulating both restricted and unrestricted data.

Reed's treatment of linear metatheory provides excellent insight into related issues, namely the difficulties encountered when reasoning about interacting linear contexts in Linear LF. In addition, the paper is an abundant source of nice analogies and appeals to intuition regarding relational metatheory, stateful systems and hybrid logic, making it a pleasant, informative read. However, its contents was certainly lacking in terms of examples. My understanding of hybrid connectives and their role in manipulating restricted hypotheses was quite vague until I applied them to the MLR case study. The next step is of course to improve my understanding further by given a similar treatment to the metatheory of MLR and exploring the challenges posed by linear metatheorems.

## References

- [1] Iliano Cervesato and Frank Pfenning. A linear logical framework. *Inf. Comput.*, 179(1):19–75, 2002.
- [2] Jason Reed. A hybrid metalogical framework. Thesis Proposal, Carnegie Mellon University, January 2007.
- [3] Jason Reed. Hybridizing a logical framework. *Electron. Notes Theor. Comput. Sci.*, 174(6):135–148, 2007.