Hybridizing a Logical Framework

Jason Reed [1,2]

Computer Science Department
Carnegie Mellon University

Abstract

Logical connectives familiar from the study of hybrid logic can be added to the logical framework LF, a constructive type theory of dependent functions. This extension turns out to be an attractively simple one, and maintains all the usual theoretical and algorithmic properties, for example decidability of type-checking. Moreover it results in a rich metalanguage for encoding and reasoning about a range of resource-sensitive substructural logics, analagous to the use of LF as a metalanguage for more ordinary logics.

This family of applications of the language, contrary perhaps to expectations of how hybridized systems are typically used, does not require the usual modal connectives box and diamond, nor any internalization of a Kripke accessibility relation. It does, however, make essential use of distinctively hybrid connectives: universal quantifiation over worlds, truth of a proposition at a named world, and local binding of the current world. This supports the claim that the innovations of hybrid logic have independent value even apart from their traditional relationship to temporal and alethic modal logics.

Key words: Hybrid logic, logical frameworks, substructural logics

1 Introduction

The notion of hybrid logic has emerged as an intermediate point in the space between modal logics and the first-order logics in which they can be soundly embedded. By providing, within the language of propositions itself, ways of explicitly referring to and manipulating modal worlds, hybridization can recover much of the expressiveness of first-order logic without entirely sacrificing the simplicity and metatheoretic felicities of a more basic modal language.

[1] jcreed@cs.cmu.edu

[2] This work has been supported by the National Science Foundation under grant CCR-0306313.

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

The story of the present piece of work has essentially the same form: by taking the logical framework LF and judiciously adding *hybrid* logical connectives, which explicitly name, bind and make use of a certain notion of 'world,' we achieve significant gains in expressiveness without making the resulting language so strong as to cause failure of familiar desirable properties of the system as a whole. A key difference is that the starting point in our case is *not* a modal logic, but rather a logical system that lacks any obvious intrinsic notion of world[3]. Thus we add such a notion, and furthermore endow it with an algebraic operation inspired by the semantics for bunched logic given by Galmiche and Méry [13]. Also worth noting is that the present system is, as is typical for logical frameworks but not for hybrid logics, a *constructive* logic, locating it within a small but growing body of work on intuitionistic and constructive hybrid logics [18,20,5,6].

The contribution of hybridization in this case is the set of connectives $\forall$, $@_p$, $\downarrow$, which respectively quantify over variables that stand for worlds, shift the perspective of a proposition to a new world p, and bind the current world to a variable. This choice of connectives is sufficient for a surprisingly large set of applications. Even without adding the typically *modal* structure of the connectives $\Box$, $\Diamond$, and an internalization of the Kripke accessibility, we are able to easily build a system that

- generalizes the linear logical framework LLF [7]
- faithfully encodes the family of 'usage analysis' function types $\rightarrow_n$ [32] that require exactly n uses of their argument.
- faithfully encodes a significant fragment of the logic of Bunched Implications [22] (the connectives $-\!\!*, \rightarrow, \wedge, \top$)
- provides a language in which one can state *linear metatheorems*, thus achieving an 'internal' and simplified version of the system described by McCreight and Schürmann [19].

This last application, discussed in Section 3.3, was in fact the principal motivation that led us to design the present system. We did not set out in the beginning to find out what could be expressed when hybrid connectives were added to LF; rather, we searched for a system in which linear metatheorems could be formulated. Hybridization was, in effect, then forced upon us, as we discovered it to be by far the most natural way of achieving our goal, and furthermore it also supports the other applications mentioned.

As another argument for the naturalness of the extension, all three hybrid connectives mentioned are related in a way that arises from considerations of *focused proofs* [1] in that they are all *right asynchronous*. In particular, the natural deduction introduction rules for each connective are invertible, i.e.

[3] If one looks to LLF rather than LF, arguably the former does have *some* notion of 'world' already, inasmuch as it is resource-sensitive at all, but one which is considerably more buried than in modal logics.

inference from the conclusion of the rule to its premise is also sound. This is why, for instance, $\downarrow$ is taken as a primitive and not the related connective $\exists$, which falls outside the right asynchronous fragment. A more thorough history of the role of $\downarrow$ as primitive vs. defined in terms of $\exists$ in various systems is given by Blackburn [4].

The remainder of the paper is structured as follows. In section 2 we sketch the definition of the hybrid logical framework HLF, and discuss substitution, normalization, and decidability properties of it. Section 3 gives some example encodings and applications. Section 4 describes related work, Section 5 outlines future possibilities, and Section 6 provides concluding comments.

2 The Hybrid Logical Framework HLF

2.1 Motivation

Before we begin a more formal treatment of the system, let us briefly consider some examples from linear logic that give insight into how hybridization arose naturally out of other concerns.

Linear logic [14] is a logic of *resources*. It has a context that does not admit contraction or weakening, and so enforces that every assumption obeys the resource discipline of being consumed exactly once. For example, the sequent $A \multimap B, A \vdash B$ (where $\multimap$ is *linear implication*) is provable, because the implication $A \multimap B$ and its argument A get consumed in producing B. Neither $A \multimap (A \multimap B), A \vdash B$ nor $A \multimap B, A, A \vdash B$ are provable, the former because there are too few copies of A in the context (we cannot use contraction to duplicate it as we can in the ordinary natural deduction proof of $A \supset A \supset B, A \vdash B$) and the latter because there is an extra copy of A (which we cannot use weakening to elide).

Trying to generalize this resource discipline in certain directions (similar to Wright's usage analysis [32]) led us to consider existing work on reducing substructural properties of the context (e.g. contraction, weakening) to algebraic properties of *labels* that are attached to deductions, i.e. labelled deduction [12]. Such systems constitute an entirely sensible reconsideration of the meaning of resource use in their own right.

Along these lines one replaces each *linear* hypothesis with an ordinary hypothesis, but labelled by a unique identifier that acts as a name for its role as a consumable resource; instead of the context $A \multimap B, A, A$ we might say something like $A \multimap B[\alpha], A[\beta], A[\gamma]$. Now the two occurrences of A in the context are distinguished as different resources that can (and must) be consumed independently. Now conclusions from these hypotheses are also labelled, by an expression that lists resources that they actually consume. In this way we could express the failure of the linear sequent $A \supset A \supset B, A \vdash B$ as the failure of the labelled sequent $A \supset B[\alpha], A[\beta], A[\gamma] \vdash B[\alpha \cdot \beta \cdot \gamma]$: given resource α of type $A \supset B$, resources β, γ both of type A, we cannot achieve

B by consuming all three resources α, β, γ.

One simple advantage of this approach is it lets us return to having contexts that *do* satisfy weakening and contraction. Adding further copies of, e.g. the hypothesis $A[\beta]$ to the context doesn't harm anything, for it is the *labels* such as β that enforce the substructural nature of the logic.

The major advantage, however, and the one that is pertinent to this paper, is that by exposing the labelled side of substructural logics, we are but a small distance away from hybridization. We need only consider that these resource labels are something like modal worlds, and the meaning of the implication $\multimap$ emerges clearly as a statement in the hybrid language that refers to them: $A \multimap B$ is achievable by consuming resources p ('at world p') if, for *any* hypothetical resource named α that has type A, we can produce B by consuming all the resources p plus the resource α ('at the local current world combined with α'). We will see in Section 3 precisely how to phrase this in terms of $\forall, \downarrow, @$.

2.2 Definition

Understanding our presentation of HLF obviously benefits from some familiarity with LF, [15,16] the system it is based on, but we provide a little background here nonetheless. LF is a λ-calculus in the Automath family, which defines a decidable typing judgment for dependently typed terms. Since dependent types are present, i.e. equality of types depends on equality of terms, equality on terms (usually defined up to $\beta-$ and optionally $\eta-$ conversion) also must be (and in fact is) decidable. The basic syntax of the language of HLF includes, as LF does, a notion of terms and type families, and furthermore a category of expressions that describe worlds:

$$
\begin{array}{rl}
\text{Worlds} & p, q, r ::= \alpha \mid p_1 \cdot p_2 \mid \epsilon \\
\text{Terms} & M ::= \lambda x.M \mid M\ N \mid x \mid c \\
\text{Type Families} & A ::= \Pi x{:}A.B \mid A\ M \mid a \mid \forall \alpha.A \mid \downarrow \alpha.A \mid @_p A \\
\text{Contexts} & \Gamma ::= \cdot \mid \Gamma, x : A \mid \Gamma, \alpha : \mathsf{world}
\end{array}
$$

World expressions, recall, are meant to describe resource use. They can be world variables α, the 'concatenation' of two other world expressions, (i.e. the simultaneous use of two sets of resources) or else the empty resource ϵ, which as a unit for concatenation makes it a monoid. The language of terms is in fact identical to that of LF. It includes function expressions, function application, variables x, and primitive constants c. The notion of type is where HLF diverges from LF. Familiar from LF are dependent function types $\Pi x{:}A.B$, application of dependent type families to arguments $A\ M$, and primitive constant type families a. What HLF adds is the remaining three type constructors, $\forall, \downarrow, @_p$. The contexts Γ in which terms are typechecked are

built up from the empty context by adding hypotheses of term variables x assumed to be at some type A, and world variables α.

The central typing judgment of HLF is

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

pronounced as 'in the context Γ, the term M has type A at world p'. This differs from LF's by the addition of the world expression p.

In order to define this judgment, make the following definitions: Let $\{M/x\}$ and $\{p/\alpha\}$ denote (capture-avoiding) substitution of terms for variables and worlds for world-variables respectively. Let $\Gamma \vdash A \equiv B$: type denote *type-directed definitional equality* (as in [16]) and $p \equiv_{ACU} q$ denote equality of world expressions up to **A**ssociativity and **C**ommutativity of $\cdot$, and **U**nit laws for ϵ with respect to $\cdot$. Let Σ be a signature of declarations of typed constants.

Then the definition of the judgment $\Gamma \vdash M : A[p]$ may be given by a set of typing rules, as follows:

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A[\epsilon]}\,\mathsf{var} \qquad \frac{c : A \in \Sigma}{\Gamma \vdash c : A[\epsilon]}\,\mathsf{const}$$

$$\frac{\Gamma \vdash A \equiv B : \text{type} \qquad \Gamma \vdash p \equiv_{ACU} q \qquad \Gamma \vdash M : A[p]}{\Gamma \vdash M : B[q]}\,\mathsf{tconv}$$

$$\frac{\Gamma, x : A \vdash M : B[p]}{\Gamma \vdash \lambda x.M : \Pi x{:}A.B[p]}\,\Pi\mathsf{I} \qquad \frac{\Gamma \vdash M : \Pi x{:}A.B[p] \qquad \Gamma \vdash N : A[\epsilon]}{\Gamma \vdash M\ N : (\{N/x\}B)[p]}\,\Pi\mathsf{E}$$

$$\frac{\Gamma, \alpha : \mathsf{world} \vdash M : B[p]}{\Gamma \vdash M : \forall\alpha.B[p]}\,\forall\mathsf{I} \qquad \frac{\Gamma \vdash M : \forall\alpha.B[p] \qquad \Gamma \vdash q : \mathsf{world}}{\Gamma \vdash M : (\{q/\alpha\}B)[p]}\,\forall\mathsf{E}$$

$$\frac{\Gamma \vdash M : (\{p/\alpha\}B)[p]}{\Gamma \vdash M : \downarrow\alpha.B[p]}\,\downarrow\mathsf{I} \qquad \frac{\Gamma \vdash M : \downarrow\alpha.B[p]}{\Gamma \vdash M : (\{p/\alpha\}B)[p]}\,\downarrow\mathsf{E}$$

$$\frac{\Gamma \vdash M : A[p]}{\Gamma \vdash M : @_p A[q]}\,@\mathsf{I} \qquad \frac{\Gamma \vdash M : @_p A[q]}{\Gamma \vdash M : A[p]}\,@\mathsf{E}$$

The auxiliary judgment $\Gamma \vdash p : \mathsf{world}$ simply establishes that all variables in p in fact occur in the context Γ. This description of the system is considerably simplified from the full version being developed [27] but it contains most the essential features. There are standard side conditions on variable occurrences in rules: introduction of quantifiers such as $\Pi, \forall$ require that the bound variables they introduce are fresh, etc. We have also omitted the additive connectives $\&, \top$.

Since the term language is the same as LF's, the role of the hybrid connectives is merely to allow description of *refinements* (that is to say, subsets of the terms) of existing types. To establish that a term M has the type $\forall\alpha.A$, we simply hypothesize a fresh α, and continue checking that M has type A,

which may refer to α. The way that we can use the knowledge that M has type $\forall\alpha.A$ is by instantiating the universal quantifier with any other valid world. The binder $\downarrow$ operates similarly, except that it fixes the instantiation to be the whatever the current world is as it is being typechecked. Finally, the type operator $@_p$ 'transfers' to the world p, in a manner similar to the appearance of @ in other hybrid logics.

The *type conversion rule* tconv usually serves to realize the action of $\beta\eta$-conversion within types. If two types are convertible, then any term that has one type has the other; thus they are the same type. Here we also allow 'world conversion,' doing the same thing for the monoid equations on worlds. If a term is well-typed at one world, it is also well-typed at any ACU-equivalent world.

We ought also to comment on the use of the 'empty' world ϵ in several rules. In var and const, it means that directly using a variable from the context or using a constant from the signature does not consume any resources. That is, the single context of the judgment is a context of *unrestricted hypotheses* in the sense of [11]. To express hypotheses that *are* restricted, e.g. that a variable of type A is only available via the consumption of a resources p, one uses the $@_p$ type operator, putting in the context a hypothesis $x : @_pA$.

In ΠE, the use of ϵ signifies that Π is still the *unrestricted* dependent function type from [11,7]. The arguments of such functions must typecheck at world ϵ. It is important for our intended encodings that we retain such a type constructor that corresponds exactly to the Π of the source of the encoding. It may well be that adding other dependent type constructors to the system that don't refer explicitly to ϵ could work, but we have not yet investigated this possibility.

However, we can already use Π and $@_p$ together to express function types that do use resources. In particular, the primitive implication in Braüner and de Paiva's hybridization of intuitionistic logic [5,6] differs from ours (in that it requires the argument of an implication to be checked at the same world as the conclusion, rather than at any distinguished world such as our ϵ) but we can easily encode their $A \supset B$ in our language as $\downarrow\alpha.\Pi x{:}(@_\alpha A).B$, or equivalently (using the standard abbreviation $A \to B$ for $\Pi x{:}A.B$ when x doesn't occur in B) as $\downarrow\alpha.(@_\alpha A \to B)$.

2.3 Metatheory

To actually prove all the results that follow, we made use of techniques developed relatively recently by Watkins et al. [8,9] For space reasons, we do not give full details here, but refer to [27] for at least a formulation of the system that makes these proofs much easier than they would be in traditional form. The general idea of such techniques is to avoid defining first the set of *all* the terms of the language (i.e. including those that have β-redices) and subsequently simply showing (typically by a logical relations argument such

as [16]) that every term has a canonical form that it reduces to.

Instead one defines the language in the first place from the perspective that *what really matters is the set of canonical forms*. By giving a syntax that only admits canonical forms, and by defining substitution in such a way that it carries out normalization in a hereditary and terminating fashion, the normalization theorem for all well-typed terms follows as a simple inductive corollary.

The reason this can be done is that the substitution function is annotated with the type of the variable being substituted for, and so induction proceeds principally on the structure of the types, even though during the course of carrying out reductions the terms involved may become larger.

In this formulation, it is extremely important (and not at all difficult) to show that substitution is correct vis-a-vis the typing judgment:

Lemma 2.1 (Substitution of Terms) *If* $\Gamma \vdash M : A[\epsilon]$ *and* $\Gamma, x : A, \Gamma' \vdash N : B[q]$*, then*

$$\Gamma, \sigma\Gamma' \vdash \sigma N : \sigma B[q]$$

where σ *abbreviates the substitution* $\{M/x\}$*.*

Lemma 2.2 (Substitution of Worlds) *Suppose* p *is a valid world, i.e.* $\Gamma \vdash p : \mathsf{world}$*.*

- *If* $\Gamma, \alpha : \mathsf{world}, \Gamma' \vdash N : B[q]$*, then* $\Gamma, \sigma\Gamma' \vdash \sigma N : \sigma B[\sigma q]$
- $\Gamma, \alpha : \mathsf{world}, \Gamma' \vdash q : \mathsf{world}$*, then* $\Gamma, \sigma\Gamma' \vdash \sigma q : \mathsf{world}$

where σ *abbreviates the substitution* $\{p/\alpha\}$*.*

Once this is done, a result that is usually somewhat more involved becomes almost trivial:

Proposition 2.3 (Normalization) *Every well-typed term normalizes to a canonical (*β*-normal,* η*-long) form.*

The most important remaining issue is the decidability of type-checking. Although there are several places in the above presentation where this proposition is not clearly true, there is one that is most essential: the rule $\forall\mathsf{E}$. Since q does not appear anywhere in the term language, it seems we must 'guess' from the term and the type in front of us what it is.

In fact, however, we have formulated and partially implemented an algorithm that tracks *residual constraints* on these unknown worlds, postponing them as unification variables until such time as they are constrained by type-checking other parts of the term. This approach is rather standard, and is much like how type inference typically works in programming languages. In particular the type conversion rule tconv is the source of unification equations (up to ACU) on worlds.

In other words, one can reduce type-checking in this system to the problem of equational unification over associativity, commutativity, and unit laws, well-known in the literature as ACU-unification, and achieve the following result:

Proposition 2.4 *(Decidability of Type-Checking) Given a well-formed context Γ, term M, type A, and world p, it is decidable whether there is a derivation of $\Gamma \vdash M : A[p]$.*

In fact, since we have no other function, constant, or relation symbols besides $\cdot$ and ϵ, we require only a fragment of ACU-unification that is easily seen to be decidable by reduction from the problem of determining solvability of systems of linear diophantine equations. This means that other algorithms of great practical importance to the front-end of systems such as Twelf [24] that depend on unification, such as type reconstruction, are very likely to admit extensions to be compatible with worlds as well. In general, extending our approach to other substructural logics requires understanding related equational unification problems, which have been widely studied.

One might reasonably ask *why* it is that $\forall$ doesn't take an argument and thereby behave more like a function type. In this case its introduction and elimination rules might look like

$$\frac{\Gamma, \alpha : \mathsf{world} \vdash M : B[p]}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.B[p]}\,\forall\mathsf{I} \qquad \frac{\Gamma \vdash M : \forall\alpha.B[p] \qquad \Gamma \vdash q : \mathsf{world}}{\Gamma \vdash M \bullet q : (\{q/\alpha\}B)[p]}\,\forall\mathsf{E}$$

for new term constructors $\Lambda, \bullet$ that create and eliminate world-abstracting functions. This would establish the decidability type-checking problem far more neatly, since there wouldn't be important information patently missing from terms, which we then go to some pains to prove that we can recover. This is a valid alternative, though we haven't pursued it very far as of yet. There don't seem to be any major difficulties, and indeed it seems to be more compatible with categorical semantics, but its fatal flaw is that it is not compatible with the vast majority of applications we care about: if worlds appear in terms, then the system can distinguish more terms than it could before. In this proposed alternative, the reason that terms are type-correct is suddenly more fully manifest in the terms themselves, and so different reasons amount to different terms, leading to a system that no longer is, for instance, a proper extension of LLF. It is possible that explicitly quotienting out this extra information via the idea of proof irrelevance [23] could solve this issue, but we leave this to future work.

3 Applications

As mentioned, the application that motivated our work was trying to achieve a clean encoding into a logical framework of the statement of theorems about substructural logics. Ideally this we would be able to adapt of the body of metatheoretic algorithms developed for LF[25,28,30] and we would then be able to formally state and mechanically check in software properties of substructural logics and languages in a far more natural way. We can presently do this, but only with much greater effort, in existing software systems such

as Twelf, which are more suited to languages without substructural features.

This goal is not a recent innovation; previous systems such as LLF [7], RLF [17], and CLF [8,9] are significant milestones in the development of substructural logical frameworks. They are already languages in which one can write *proofs* (and, being constructive proofs, these are essentially programs) but what is missing is a language for *stating* with sufficient precision what theorems these proofs actually prove, and tools for checking such claims. We claim that HLF is such a language, and we plan to develop appropriate tools.

We begin by explaining how HLF is a generalization of the linear logical framework LLF.

3.1 Embedding LLF

The language of LLF [7] extends LF with features of linear logic. The context in LLF has *unrestricted* hypotheses $x : A$ which may be used any number of times, and *linear* hypotheses $x \mathbin{\hat{:}} A$ that express resources that must be used exactly once. The logical connectives taken from linear logic are $\multimap, \&, \top$.

The connectives $\&, \top$ can be mapped directly onto the connectives of the same name in HLF, which we have not discussed, because they are of little relevance to the hybridization of the system. What is interesting from that perspective is the connective $\multimap$. Make the following definition. If A is a type of LLF, then A^* is A with every subexpression $B \multimap C$ of it replaced by

$$\downarrow\alpha.\forall\beta.((@_\beta B) \to (@_{\alpha\cdot\beta} C))$$

(where again $A \to B$ is an abbreviation for $\Pi x{:}A.B$, and where α, β are fresh variables) This 'macro expansion' of $B \multimap C$ decomposes it as meaning: let the current resources be called α. For any extra resources β, if we can produce B with β, then we can produce C with α and β together.

With this encoding we obtain the result that the logical framework LLF embeds faithfully as a subsystem of HLF:

Proposition 3.1 *The closed LLF terms well-typed at any LLF type A are in bijective correspondence to the closed HLF terms at type A^*, at world ϵ.*

In fact this bijection is only a bureacratic detail away from being an identity. If we start with a variant of LLF that does not *syntactically* distinguish unrestricted function expressions from linear function expressions, then we would obtain the stronger result

Proposition 3.2 *Let A be an LLF type, and Γ an LLF context. The system LLF derives $\Gamma \vdash M : A$ if and only if the system HLF derives $\Gamma^* \vdash M : A^*[\alpha_\Gamma]$,*

where the operation Γ^* on contexts is defined by

$$\begin{aligned}
(\Gamma, x : A)^* &= \Gamma^*, x : A^* \\
(\Gamma, x \mathbin{\hat{:}} A)^* &= \Gamma^*, \alpha_x : \mathsf{world}, x : @_{\alpha_x}(A^*) \qquad (\alpha_x \text{ fresh})
\end{aligned}$$

and where α_Γ is a $\cdot$-delimited list of α_x for each $x \mathbin{\hat{:}} B \in \Gamma$.

3.2 Usage Analysis Types

A generalization of the previous encoding that is also easy to achieve in this language is the idea of the function type $\rightarrow_n$, which requires its argument to be used exactly n times. For it we simply make the definition

$$\beta^n \equiv \overbrace{\beta \cdot \cdots \cdot \beta}^{n \text{ times}}$$

and then define $\rightarrow_n$ as a macro for

$$A \rightarrow_n B \equiv \downarrow\alpha.\forall\beta.((@_\beta A) \rightarrow (@_{\alpha\cdot\beta^n} B))$$

3.3 Metatheory for the Linear Logical Framework

A common use for LF and its implementation Twelf is to encode logics and proofs of properties about them by writing recursive functions in *relational style* that transform derivations of the logic being studied. For example, a cut elimination proof for intuitionistic logic can be stated by making the declaration of a type family (i.e. a three-place relation)

$$\mathsf{cut\text{-}admissibility} : \Pi A : o.\Pi B : o.$$

$$\mathsf{conc}\ A \rightarrow (\mathsf{hyp}\ A \rightarrow \mathsf{conc}\ B) \rightarrow \mathsf{conc}\ B \rightarrow \text{type}$$

This declaration together with *mode declarations* to the effect that there are supposed to be input derivations of $\mathsf{conc}\ A$ and $(\mathsf{hyp}\ A \rightarrow \mathsf{conc}\ B)$ and an output of $\mathsf{conc}\ B$ constitute the specification for a program, which if written, is a constructive proof of

Fact 3.3 (Cut Admissibility) *If* $\Gamma \vdash A$ *and* $\Gamma, A \vdash B$*, then* $\Gamma \vdash B$*.*

for ordinary propositional logic. Now the linear cut admissibility theorem behaves differently:

Fact 3.4 (Linear Cut Admissibility) *If* $\Gamma \vdash A$ *and* $\Delta, A \vdash B$*, then* $\Gamma, \Delta \vdash B$*.*

It involves contexts that must be *combined* as lists of resources, and so if we cannot write specifications that take this into account, we cannot state as powerful theorems as we would like. We can only state weaker theorems whose proofs might mismanage the context and still pass muster according to the formal checker.

One can (and we did for a long time!) try to mix linearity and dependency to leverage linear connectives to express relationships between contexts in such a theorem, saying something like

$$\mathsf{cut\text{-}admissibility} : \Pi A : o.\Pi B : o.$$

$$((\mathsf{conc}\ A \otimes (\mathsf{hyp}\ A \multimap \mathsf{conc}\ B)) \,\&\, \mathsf{conc}\ B) \multimap \mathrm{type}$$

However, we were never able to make this into a completely satisfactory system. What does work elegantly is the language of hybrid connectives, as follows:

$$\mathsf{cut\text{-}admissibility} : \Pi A : o.\Pi B : o.\forall\alpha.\forall\beta.$$
$$@_\alpha(\mathsf{conc}\ A) \to @_\beta(\mathsf{hyp}\ A \multimap \mathsf{conc}\ B) \to @_{\alpha\cdot\beta}(\mathsf{conc}\ B) \to \mathrm{type}$$

Here we are able to refer explicitly to worlds that stand in place of contexts in the statement of the theorem we wish to verify. In this way the formal theorem plainly resembles the informal version, thus giving us even greater confidence that the formalization is correct.

3.4 Embedding Bunched Logic

One can also embed several connectives from the logic of bunched implications [22] in much the same way as the linear embedding. Here we merely need to say that bunched $\wedge$ maps onto HLF &, bunched $\top$ maps onto HLF $\top$, and the bunched multiplicative and additive arrow $-\!\!*, \to$ are translated by macro expansion as follows:

$$A -\!\!* B \equiv \downarrow\alpha.\forall\beta.((@_\beta A) \to (@_{\alpha\cdot\beta} B))$$
$$A \to B \equiv \downarrow\alpha.((@_\alpha A) \to B)$$

Note that the encoding of $-\!\!*$ is precisely the same as that of $\multimap$, and that the encoding of the bunched additive arrow is the same as Braüner-de Paiva implication mentioned in Section 2.2.

4 Related Work

It has long been recognized that there is an enormous potential gain in expressivity of a logic that comes with the ability to explicitly refer to some sort of world that parametrizes the judgment of truth of propositions, whether conceived modally as possible worlds in hybrid logic [2,4], or thought of as abstract labels in labelled deduction [12,31,26], or concretely in recent research into mobile programs as hosts on a network network hosts [20,18,21].

The important immediate predecessors to our work within the arena of hybrid logics are the constructive systems of hybrid logic independently devised by Braüner and de Paiva [5,6] Jia and Walker [18] and Chadha et al. [10] Our contributions relative to these consist of adapting the connectives to a dependently typed system, and to a notion of world that supports a monoidal structure (this notion of world by itself, is of course not original to the present work, see for instance the work of Galmiche and Méry [13]).

On the logical frameworks side, McCreight and Schürmann developed a metalogic $\mathcal{L}^+_\omega$ [19] for LLF, parallel to the metalogic $\mathcal{M}^+_\omega$ for LF that is the

subject of Schürmann's thesis [29]. The logic $\mathcal{L}_\omega^+$, broadly speaking, is a solution to the same metatheoretic encoding problem that we aimed to solve, but to define $\mathcal{L}_\omega^+$ it is necessary to quantify over variables that stand for linear contexts, which leads to significant complications and concerns about predicativity. Our solution avoids these by replacing quantification over *contexts* with quantification over *worlds*, which are simpler and more manageable, since they abstract away just those features that are essential for describing how contexts are manipulated. Moreover our system permits encodings of metatheorems and proofs in HLF itself, a noteworthy practical advantage already enjoyed by standard practices embedding metatheorems and proofs in LF.

The idea of more intimately blending resource use with dependent types, which in principle could also provide a non-hybrid answer to our problem, appears in the system RLF developed by Ishtiaq and Pym [17]. What they call 'linearity' in that work, however, is not the idea of resources that must be used *exactly* once, but rather *at least* once, i.e. a logic of *relevant* implication. We conjecture that their approach, which apparently fundamentally allows 'linear' variables to appear both in a term and its type, cannot be extended to accomodate linearity in the exactly-one-use sense.

5 Future Work

Our highest priority is to understand how to use this system to fully solve the problem that motivated its development in the first place: the problem of encoding and mechanically checking proofs about encodings of substructural logics in logical frameworks such as LLF. This problem has received copious attention in the case of more ordinary encodings into LF.

For us it remains to adapt several pieces of the metatheory of LF, and the theory of logic programs encoded in it. First of all, the operational semantics for logic programming over HLF needs to be understood, but this is likely to be relatively simple, since it is likely we can treat worlds much like terms 'of type world' and $\forall$ much like Π. We will need to extend specifications of input-output behavior (called *mode specifications* [28]) to apply to quantified world variables, and determine whether existing conservative termination checking algorithms [25] still apply. We expect most of the difficulty to arise in understanding the right way to adapt coverage checking [30], which is critical in the mechanization of proofs represented as programs that compute (generally partial) functions. A function that works by case analysis must cover all cases to be total, and it must be a total function to represent a correct proof.

Although in this paper we specifically did not focus on modal logics that feature an accessibility relation, there is no reason yet apparent why it should not be compatible with the given system. The main issue there the decidability of unification in the theory of worlds, concatenation, and an accessibility relation between worlds with some fixed axioms on it, for the decidability of type-checking hinges on this problem.

We would like to exhibit a categorical semantics and prove soundness and completeness with respect to the current system. Progress in this direction consists of the observation that the addition of the hybrid features described above to a dependent type system appears to be well modelled by taking a locally cartesian closed category (well-known to canonically model dependent types) and adding a monoid object. However, the behavior of the universal quantifier over worlds does not quite match up with its natural interpretation as a dependent function type whose domain is the carrier type of the monoid object, i.e. the type of worlds. The reason for this is that the system given syntactically treats worlds in a proof irrelevant fashion [23], so we may need to use the techniques developed by Awodey and Bauer [3] to express this categorically.

Finally, another important direction to pursue is trying to recover those connectives that are not right asynchronous: $\vee, \bot$ in ordinary logic, $\otimes, 1, \oplus, !$ in linear logic, and $*, 1_m$ in bunched logic. The central obstacle these pose to the usual methodologies of logical frameworks, namely the presence of commuting conversions, has already been addressed in CLF, [8,9] so we expect a solution along similar lines to be possible. Given the semantic analysis [13] of the bunched 'fuse' connective $*$, this may in any event also require treating some form of accessibility relation.

6 Conclusion

We have shown how to extend LF by a notion of world, and logical connectives that explicitly manage worlds, yielding a hybrid logical framework HLF. Although these worlds are different from the 'worlds' in modal logics of time and necessity, they react just as well to hybridization, in that the resulting system is very expressive. It already subsumes one other well-known extension to LF, and promises to generalize others. We conclude that hybric logic is useful to the study of substructural logics, just as it has been fruitful in augmenting the study of traditional modal logics.

References

[1] Andreoli, J. M., *Logic programming with focusing proofs in linear logic*, Journal of Logic and Computation **2** (1992), pp. 297–347.

[2] Areces, C., P. Blackburn and M. Marx, *Hybrid logics: Characterization, interpolation and complexity*, Journal of Symbolic Logic **66** (2001), pp. 977–1010.

[3] Awodey, S. and A. Bauer, *Propositions as [Types]*, Technical report, Institut Mittag-Leffler (2001).

[4] Blackburn, P., *Representation, reasoning, and relational structures: a hybrid logic manifesto*, Logic Journal of IGPL **8** (2000), pp. 339–365.

[5] Braüner, T. and V. de Paiva, *Towards constructive hybrid logic*, Elec. Proc. of Methods for Modalities **3** (2003).

[6] Braüner, T. and V. de Paiva., *Intuitionistic hybrid logic* (2006), to appear.

[7] Cervesato, I. and F. Pfenning, *A linear logical framework*, Inf. Comput. **179** (2002), pp. 19–75.

[8] Cervesato, I., F. Pfenning, D. Walker and K. Watkins, *A concurrent logical framework I: Judgments and properties*, Technical Report CMU-CS-02-101, Department of Computer Science (2003).

[9] Cervesato, I., F. Pfenning, D. Walker and K. Watkins, *A concurrent logical framework II: Examples and applications*, Technical Report CMU-CS-02-102, Department of Computer Science (2003).

[10] Chadha, R., D. Macedonio and V. Sassone, *A Hybrid Intuitionistic Logic: Semantics and Decidability*, Journal of Logic and Computation **16** (2006), p. 27.

[11] Chang, B.-Y. E., K. Chaudhuri and F. Pfenning, *A judgmental analysis of linear logic*, Technical Report CMU-CS-03-131, Carnegie Mellon University (2003).

[12] Gabbay, D., *Labelled deductive systems*, Technical Report 90-22, University of Munich (1990).

[13] Galmiche, D. and D. Méry, *Semantic Labelled Tableaux for Propositional BI*, Journal of Logic and Computation **13** (2003), pp. 707–753.

[14] Girard, J., *Linear logic*, Theoretical Computer Science **50** (1987), pp. 1–102.

[15] Harper, R., F. Honsell and G. Plotkin, *A framework for defining logics*, Journal of the Association for Computing Machinery **40** (1993), pp. 143–184.

[16] Harper, R. and F. Pfenning, *On Equivalence and Canonical Forms in the LF Type Theory*, ACM Transactions on Computational Logic **6** (2005), pp. 61–101.

[17] Ishtiaq, S. S. and D. J. Pym, *A relevant analysis of natural deduction*, Journal of Logic and Computation **8** (1998), pp. 809–838.

[18] Jia, L. and D. Walker, *Modal proofs as distributed programs*, Technical Report TR-671-03, Princeton University (2003).

[19] McCreight, A. and C. Schürmann, *A meta linear logical framework* (2003), draft manuscript.
URL http://www.itu.dk/people/carsten/papers/mllf.pdf

[20] Moody, J., *Modal logic as a basis for distributed computation*, Technical Report CMU-CS-03-194, Carnegie Mellon University (2003).

[21] Murphy, T., VII, K. Crary, R. Harper and F. Pfenning, *A Symmetric Modal Lambda Calculus for Distributed Computing*, Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS'04)-Volume 00 (2004), pp. 286–295.

[22] O'Hearn, P. and D. Pym, *The logic of bunched implications*, Bulletin of Symbolic Logic **5** (1999), pp. 215–244.

[23] Pfenning, F., *Intensionality, extensionality, and proof irrelevance in modal type theory*, in: J. Halpern, editor, *Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01)* (2001), pp. 221–230.

[24] Pfenning, F. and C. Schürmann, *System description: Twelf — a meta-logical framework for deductive systems*, in: H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)* (1999), pp. 202–206.

[25] Pientka, B. and F. Pfenning, *Termination and reduction checking in the logical framework*, in: *Workshop on Automation of Proofs by Mathematical Induction*, 2000.

[26] Rasga, J., A. Sernadas, C. Sernadas and L. Vigano, *Fibring labelled deduction systems*, Journal of Logic and Computation **12** (2002), pp. 443–473.

[27] Reed, J., *Canonical forms in a hybrid logical framework* (2006), draft manuscript.
URL http://www.cs.cmu.edu/~jcreed/papers/cfhlf.pdf

[28] Rohwedder, E. and F. Pfenning, *Mode and termination checking for higher-order logic programs*, in: H. R. Nielson, editor, *Proceedings of the European Symposium on Programming* (1996), pp. 296–310.

[29] Schürmann, C., *Automating the meta-theory of deductive systems*, Technical Report CMU-CS-00-146, Department of Computer Science, Carnegie Mellon University (2000).

[30] Schürmann, C. and F. Pfenning, *A coverage checking algorithm for LF*, in: D. Basin and B. Wolff, editors, *Proceedings of the Theorem Proving in Higher Order Logics 16th International Conference*, LNCS **2758** (2003), pp. 120–135.

[31] Viganò, L., "Labelled Non-Classical Logics," Kluwer Academic Publishers, 2000.

[32] Wright, D. and C. Baker-Finch, *Usage Analysis with Natural Reduction Types*, in: *Proceedings of the Third International Workshop on Static Analysis* (1993), pp. 254–266.