Gluing together Proof Environments: Canonical extensions of LF Type Theories featuring Locks

Furio Honsell, Luigi Liquori, Petar Maksimovic, Ivan Scagnetto

Gluing together Proof Environments: Canonical extensions of LF Type Theories featuring *Locks*[*]

Furio Honsell
Department of Mathematics and Computer Science
University of Udine, Italy
furio.honsell@uniud.it

Luigi Liquori
Inria Sophia Antipolis Méditerranée, France
luigi.liquori@inria.fr

Petar Maksimović
Inria Rennes Bretagne Atlantique, France
Mathematical Institute of the Serbian Academy of Sciences and Arts, Serbia
petar.maksimovic@inria.fr

Ivan Scagnetto
Department of Mathematics and Computer Science
University of Udine, Italy
ivan.scagnetto@uniud.it

We present two extensions of the LF Constructive Type Theory featuring monadic *locks*. A lock is a monadic type construct that captures the effect of an *external call to an oracle*. Such calls are the basic tool for *gluing together* diverse Type Theories and proof development environments. The oracle can either be invoked in order to check that a constraint holds or to provide a suitable witness. The systems are presented in the *canonical style* developed by the CMU School. The first system, $\mathsf{CLLF}_{\mathscr{P}}$, is the canonical version of the system $\mathsf{LLF}_{\mathscr{P}}$, presented earlier by the authors. The second system, $\mathsf{CLLF}_{\mathscr{P}?}$, features the possibility of invoking the oracle to obtain a witness satisfying a given constraint. We discuss encodings of Fitch-Prawitz Set theory, call-by-value λ-calculi, and systems of Light Linear Logic. Finally, we show how to use Fitch-Prawitz Set Theory to define a type system that types precisely the strongly normalizing terms.

1 Introduction

In recent years, the authors have introduced in a series of papers [17, 15, 20, 19] various extensions of the Constructive Type Theory LF, with the goal of defining a simple *Universal Meta-language* that can support the effect of *gluing together*, *i.e.* interconnecting, different type systems and proof development environments.

The basic idea underpinning these logical frameworks is to allow for the user to express explicitly, in an LF type-theoretic framework the *invocation*, and uniform *recording* of the *effect*, of external tools by means of a new *monadic* type-constructor $\mathscr{L}^{\mathscr{P}}_{M,\sigma}[\cdot]$, called a *lock*. More specifically, locks permit to express the fact that, in order to obtain a term of a given type, it is necessary to *verify*, first, a constraint $\mathscr{P}(\Gamma \vdash_\Sigma M : \sigma)$, *i.e.* to *produce* suitable *evidence*. No restrictions are enforced on producing such evidence. It can be supplied by calling an *external proof search tool* or an *external oracle*, or exploiting some other epistemic source, such as diagrams, physical analogies, or explicit computations according to the *Poincaré Principle* [3]. Thus, by using lock constructors, one can *factor-out* the goal, produce pieces of evidence using different proof environments and *glue* them back together, using the *unlock operator*, which *releases* the locked term in the calling framework.

[*]The work presented in this paper was partially supported by the Serbian Ministry of Education, Science, and Technological Development, projects ON174026 and III44006.

Submitted to:
LFMTP 2015

One of the original contributions of this paper is that we show how locks can delegate to external tools not only the task of producing suitable evidence but also that of exhibiting suitable *witnesses*, to be further used in the calling environment.

Locks subsume different *proof attitudes*, such as proof-irrelevant approaches, where one is only interested in knowing that a proof witness exists, or approaches relying on powerful terminating metalanguages. Indeed, they allow for a straightforward accommodation of many different *proof cultures* within a single Logical Framework; which otherwise can be embedded only very deeply [5, 14] or axiomatically [21].

Differently from our earlier work, we focus in this paper only on systems presented in the *canonical format* introduced by the CMU school [32, 13]. This format is syntax-directed and produces a unique derivation for each derivable judgement. Terms are all in normal form and equality rules are replaced by *hereditary substitution*. Canonical systems are very handy in establishing adequacy of encodings.

First, we present the very expressive system $\mathsf{CLLF}_{\mathscr{P}}$ and discuss the relationship to its non-canonical counterpart $\mathsf{LLF}_{\mathscr{P}}$ in [19], where we introduced *lock-types* following the paradigm of Constructive Type Theory (*à la* Martin-Löf), via *introduction*, *elimination*, and *equality rules*. This paradigm needs to be rephrased for the canonical format used here. Introduction rules correspond to *type checking* rules of *canonical objects*, whereas elimination rules correspond to *type synthesis* rules of *atomic objects*. Equality rules are rendered via the rules of *hereditary substitution*. In particular, we introduce a *lock constructor* for building canonical objects $\mathscr{L}^{\mathscr{P}}_{N,\sigma}[M]$ of type $\mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]$, via the *type checking rule* $(O{\cdot}Lock)$. Correspondingly, we introduce an *unlock destructor*, $\mathscr{U}^{\mathscr{P}}_{N,\sigma}[M]$, and an *atomic rule* $(O{\cdot}Unlock)$, allowing elimination, in the hereditary substitution rules, of the lock-type constructor, under the condition that a specific predicate $\mathscr{P}$ is verified, possibly *externally*, on a judgement:

$$\frac{\Gamma \vdash_\Sigma M \Leftarrow \rho \quad \Gamma \vdash_\Sigma N \Leftarrow \sigma}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{N,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]}\ (O{\cdot}Lock) \qquad \frac{\Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho] \quad \Gamma \vdash_\Sigma N \Leftarrow \sigma \quad \mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)}{\Gamma \vdash_\Sigma \mathscr{U}^{\mathscr{P}}_{N,\sigma}[A] \Rightarrow \rho}\ (O{\cdot}Unlock)$$

Capitalizing on the monadic nature of the lock constructor, as we did for the systems in [20, 19], one can replace locked terms without necessarily establishing the predicate. This increases the expressivity of the system, and allows for reasoning under the assumption that the verification is successful, as well as for postponing verifications. The crucial rules are:

$$\frac{\Gamma, x{:}\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho]\ \mathsf{type} \quad \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\tau] \quad \rho[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^F_{(\tau)^-} = \rho'}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho']\ \mathsf{type}}\ (F{\cdot}Nested{\cdot}Unlock)$$

$$\frac{\begin{array}{ll}\Gamma, x{:}\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho] & \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\tau] \\ \rho[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^F_{(\tau)^-} = \rho' & M[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^O_{(\tau)^-} = M'\end{array}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[M'] \Leftarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho']}\ (O{\cdot}Nested{\cdot}Unlock)$$

The $(O{\cdot}Nested{\cdot}Unlock)$-rule is the counterpart of the elimination rule for monads, once we realize that $let_{T_{\mathscr{P}(\Gamma \vdash S:\sigma)}} x = A\ in\ N$ can be replaced by $N[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]$. And this holds since the $\mathscr{L}^{\mathscr{P}}_{S,\sigma}[\cdot]$-monad satisfies the property $let_{T_{\mathscr{P}}}\ x = M\ in\ N \to N$ if $x \notin \mathsf{Fv}(N)$, provided x occurs *guarded* in N, *i.e.* within subterms of the appropriate lock-type. The rule $(F{\cdot}Nested{\cdot}Unlock)$ takes care of elimination at the level of types.

We proceed then to introduce $\mathsf{CLLF}_{\mathscr{P}?}$. Syntactically, it might appear as a minor variation of $\mathsf{CLLF}_{\mathscr{P}}$, but the lock constructor is used here to express the *request* for a witness satisfying a given property, which is then *replaced* by the unlock operation. In $\mathsf{CLLF}_{\mathscr{P}?}$, the lock acts as a *binding operator* and the unlock as an *application*.

$$\begin{array}{rclrcll}
K & \in & \mathscr{K} & K & ::= & \mathsf{type} \mid \Pi x{:}\sigma.K & \textit{Kinds} \\
\alpha & \in & \mathscr{F}_a & \alpha & ::= & a \mid \alpha N & \textit{Atomic Families} \\
\sigma,\tau,\rho & \in & \mathscr{F} & \sigma & ::= & \alpha \mid \Pi x{:}\sigma.\tau \mid \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho] & \textit{Canonical Families} \\
A & \in & \mathscr{O}_a & A & ::= & c \mid x \mid A\,M \mid \mathscr{U}^{\mathscr{P}}_{N,\sigma}[A] & \textit{Atomic Objects} \\
M,N & \in & \mathscr{O} & M & ::= & A \mid \lambda x{:}\sigma.M \mid \mathscr{L}^{\mathscr{P}}_{N,\sigma}[M] & \textit{Canonical Objects} \\
\Sigma & \in & \mathscr{S} & \Sigma & ::= & \emptyset \mid \Sigma, a{:}K \mid \Sigma, c{:}\sigma & \textit{Signatures} \\
\Gamma & \in & \mathscr{C} & \Gamma & ::= & \emptyset \mid \Gamma, x{:}\sigma & \textit{Contexts}
\end{array}$$

Figure 1: Syntax of $\mathsf{CLLF}_{\mathscr{P}}$

To illustrate the expressive power of $\mathsf{CLLF}_{\mathscr{P}}$ and $\mathsf{CLLF}_{\mathscr{P}?}$ we discuss various challenging encodings of subtle logical systems, as well as some novel applications. First, we encode in $\mathsf{LLF}_{\mathscr{P}}$ Fitch-Prawitz consistent Set-Theory (FPST), as presented in [29], and to illustrate its expressive power, we show, by way of example, how it can type all strongly normalizing terms. Next, we give signatures in $\mathsf{LLF}_{\mathscr{P}}$ of a strongly normalizing λ-calculus and a system of Light Linear Logic [2]. Finally, in Section 4.5, we show how to encode functions in $\mathsf{CLLF}_{\mathscr{P}?}$.

The paper is organized as follows: in Section 2 we present the syntax, the type system and the metatheory of $\mathsf{CLLF}_{\mathscr{P}}$, whereas $\mathsf{CLLF}_{\mathscr{P}?}$ is introduced in Section 3. Section 4 is devoted to the presentation and discussion of case studies. Finally, connections with related work in the literature appear in Section 5.

2 The Canonical System $\mathsf{CLLF}_{\mathscr{P}}$

In this section, we discuss the *canonical* counterpart of $\mathsf{LLF}_{\mathscr{P}}$[19], *i.e.* $\mathsf{CLLF}_{\mathscr{P}}$, in the style of [32, 13]. This approach amounts to restricting the language only to terms in long $\beta\eta$-normal form. These are the normal forms of the original system which are normal also w.r.t. typed η-like expansion rules, namely $M \to \lambda x{:}\sigma.Mx$ and $M \to \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\mathscr{U}^{\mathscr{P}}_{N,\sigma}[M]]$. The syntax of $\mathsf{CLLF}_{\mathscr{P}}$ defines the *normal forms* of $\mathsf{LLF}_{\mathscr{P}}$, and the typing system captures all the judgements in *long* $\beta\eta$*-normal form* which are derivable in $\mathsf{LLF}_{\mathscr{P}}$. The added value of canonical systems such as $\mathsf{CLLF}_{\mathscr{P}}$ is that one can streamline results of adequacy for encoded systems. Indeed, reductions in the meta-language of non-canonical terms reflect only the history of how the proof was developed using lemmata.

2.1 Syntax and Type System for $\mathsf{CLLF}_{\mathscr{P}}$

The syntax of $\mathsf{CLLF}_{\mathscr{P}}$ is presented in Figure 1. The type system for $\mathsf{CLLF}_{\mathscr{P}}$ is shown in Figure 2. The judgements of $\mathsf{CLLF}_{\mathscr{P}}$ are the following:

$$\begin{array}{rcll}
 & \Sigma & \mathsf{sig} & \Sigma \text{ is a valid signature} \\
 & \vdash_\Sigma & \Gamma & \Gamma \text{ is a valid context in } \Sigma \\
\Gamma & \vdash_\Sigma & K & K \text{ is a kind in } \Gamma \text{ and } \Sigma \\
\Gamma & \vdash_\Sigma & \sigma \ \mathsf{type} & \sigma \text{ is a canonical family in } \Gamma \text{ and } \Sigma \\
\Gamma & \vdash_\Sigma & \alpha \Rightarrow K & K \text{ is the kind of the atomic family } \alpha \text{ in } \Gamma \text{ and } \Sigma \\
\Gamma & \vdash_\Sigma & M \Leftarrow \sigma & M \text{ is a canonical term of type } \sigma \text{ in } \Gamma \text{ and } \Sigma \\
\Gamma & \vdash_\Sigma & A \Rightarrow \sigma & \sigma \text{ is the type of the atomic term } A \text{ in } \Gamma \text{ and } \Sigma
\end{array}$$

The judgements Σ sig, and $\vdash_\Sigma \Gamma$, and $\Gamma \vdash_\Sigma K$ are as in Section 2.1 of [18], whereas the remaining ones are peculiar to the canonical style. Informally, the judgment $\Gamma \vdash_\Sigma M \Leftarrow \sigma$ uses σ to check the type of the canonical term M, while the judgment $\Gamma \vdash_\Sigma A \Rightarrow \sigma$ uses the type information contained in the atomic term A and Γ to synthesize σ. Predicates $\mathscr{P}$ in $\mathsf{CLLF}_{\mathscr{P}}$ are defined on judgements of the shape

Valid signatures

$$\frac{}{\emptyset\ \mathsf{sig}}\ (S{\cdot}Empty) \qquad \frac{\Sigma\ \mathsf{sig} \quad \vdash_\Sigma K \quad a \notin \mathsf{Dom}(\Sigma)}{\Sigma, a{:}K\ \mathsf{sig}}\ (S{\cdot}Kind) \qquad \frac{\Sigma\ \mathsf{sig} \quad \vdash_\Sigma \sigma\ \mathsf{type} \quad c \notin \mathsf{Dom}(\Sigma)}{\Sigma, c{:}\sigma\ \mathsf{sig}}\ (S{\cdot}Type)$$

Kind rules

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma \mathsf{type}}\ (K{\cdot}Type)$$

$$\frac{\Gamma, x{:}\sigma \vdash_\Sigma K}{\Gamma \vdash_\Sigma \Pi x{:}\sigma.K}\ (K{\cdot}Pi)$$

Context rules

$$\frac{\Sigma\ \mathsf{sig}}{\vdash_\Sigma \emptyset}\ (C{\cdot}Empty)$$

$$\frac{\vdash_\Sigma \Gamma \quad \Gamma \vdash_\Sigma \sigma\ \mathsf{type} \quad x \notin \mathsf{Dom}(\Gamma)}{\vdash_\Sigma \Gamma, x{:}\sigma}\ (C{\cdot}Type)$$

Atomic Family rules

$$\frac{\vdash_\Sigma \Gamma \quad a{:}K \in \Sigma}{\Gamma \vdash_\Sigma a \Rightarrow K}\ (A{\cdot}Const)$$

$$\frac{\begin{array}{l}\Gamma \vdash_\Sigma \alpha \Rightarrow \Pi x{:}\sigma.K_1 \\ \Gamma \vdash_\Sigma M \Leftarrow \sigma \\ K_1[M/x]^K_{(\sigma)^-} = K\end{array}}{\Gamma \vdash_\Sigma \alpha M \Rightarrow K}\ (A{\cdot}App)$$

Atomic Object rules

$$\frac{\vdash_\Sigma \Gamma \quad c{:}\sigma \in \Sigma}{\Gamma \vdash_\Sigma c \Rightarrow \sigma}\ (O{\cdot}Const)$$

$$\frac{\vdash_\Sigma \Gamma \quad x{:}\sigma \in \Gamma}{\Gamma \vdash_\Sigma x \Rightarrow \sigma}\ (O{\cdot}Var)$$

$$\frac{\begin{array}{l}\Gamma \vdash_\Sigma A \Rightarrow \Pi x{:}\sigma.\tau_1 \\ \Gamma \vdash_\Sigma M \Leftarrow \sigma \quad \tau_1[M/x]^F_{(\sigma)^-} = \tau\end{array}}{\Gamma \vdash_\Sigma A M \Rightarrow \tau}\ (O{\cdot}App)$$

$$\frac{\begin{array}{l}\Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho] \\ \Gamma \vdash_\Sigma N \Leftarrow \sigma \quad \mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)\end{array}}{\Gamma \vdash_\Sigma \mathscr{U}^{\mathscr{P}}_{N,\sigma}[A] \Rightarrow \rho}\ (O{\cdot}Unlock)$$

Canonical Family rules

$$\frac{\Gamma \vdash_\Sigma \alpha \Rightarrow \mathsf{type}}{\Gamma \vdash_\Sigma \alpha\ \mathsf{type}}\ (F{\cdot}Atom)$$

$$\frac{\Gamma, x{:}\sigma \vdash_\Sigma \tau\ \mathsf{type}}{\Gamma \vdash_\Sigma \Pi x{:}\sigma.\tau\ \mathsf{type}}\ (F{\cdot}Pi)$$

$$\frac{\Gamma \vdash_\Sigma \rho\ \mathsf{type} \quad \Gamma \vdash_\Sigma N \Leftarrow \sigma}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]\ \mathsf{type}}\ (F{\cdot}Lock)$$

$$\frac{\begin{array}{l}\Gamma, x:\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho]\ \mathsf{type} \\ \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\tau] \\ \rho[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^F_{(\tau)^-} = \rho'\end{array}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho']\ \mathsf{type}}\ (F{\cdot}Nested{\cdot}Unlock)$$

Canonical Object rules

$$\frac{\Gamma \vdash_\Sigma A \Rightarrow \alpha}{\Gamma \vdash_\Sigma A \Leftarrow \alpha}\ (O{\cdot}Atom)$$

$$\frac{\Gamma, x{:}\sigma \vdash_\Sigma M \Leftarrow \tau}{\Gamma \vdash_\Sigma \lambda x{:}\sigma.M \Leftarrow \Pi x{:}\sigma.\tau}\ (O{\cdot}Abs)$$

$$\frac{\Gamma \vdash_\Sigma M \Leftarrow \rho \quad \Gamma \vdash_\Sigma N \Leftarrow \sigma}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{N,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]}\ (O{\cdot}Lock)$$

$$\frac{\begin{array}{ll}\Gamma, x{:}\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho] & \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\tau] \\ \rho[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^F_{(\tau)^-} = \rho' & M[\mathscr{U}^{\mathscr{P}}_{S,\sigma}[A]/x]^O_{(\tau)^-} = M'\end{array}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{S,\sigma}[M'] \Leftarrow \mathscr{L}^{\mathscr{P}}_{S,\sigma}[\rho']}\ (O{\cdot}Nested{\cdot}Unlock)$$

Figure 2: The $\mathsf{CLLF}_{\mathscr{P}}$ Type System

$\Gamma \vdash_\Sigma M \Leftarrow \sigma$. The type system makes use, in the rules $(A{\cdot}App)$ and $(F{\cdot}App)$, of the notion of *Hereditary Substitution*, which computes the normal form resulting from the substitution of one normal form into another. The general form of the hereditary substitution judgement is $T[M/x]^t_\rho = T'$, where M is the term being substituted, x is the variable being substituted for, T is the term being substituted into, T' is the result of the substitution, ρ is the *simple-type* of M, and t denotes the syntactic class (*e.g*. atomic families/object, canonical families/objects, etc.) under consideration.

The simple-type ρ of M is obtained via the erasure function of [13] (Figure 3), mapping dependent into simple-types. The rules for Hereditary Substitution are presented in Figures 4 and 5, using Barendregt's hygiene condition.

Notice that, in the rule $(O{\cdot}Atom)$ of the type system (Figure 2), the syntactic restriction of the classifier to α atomic ensures that canonical forms are *long $\beta\eta$-normal forms* for the suitable notion of long

$$\frac{}{(a)^- = a} \qquad \frac{(\alpha)^- = \rho}{(\alpha M)^- = \rho} \qquad \frac{(\sigma)^- = \rho_1 \quad (\tau)^- = \rho_2}{(\Pi x{:}\sigma.\tau)^- = \rho_1 \to \rho_2} \qquad \frac{(\tau)^- = \rho}{(\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\tau])^- = \mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]}$$

Figure 3: Erasure to simple-types

Substitution in Kinds

$$\frac{}{\mathsf{type}[M_0/x_0]^K_{\rho_0} = \mathsf{type}}\ (\mathcal{S}\cdot K\cdot Type) \qquad \frac{\sigma[M_0/x_0]^F_{\rho_0} = \sigma' \quad K[M_0/x_0]^K_{\rho_0} = K'}{(\Pi x{:}\sigma.K)[M_0/x_0]^K_{\rho_0} = \Pi x{:}\sigma.'K'}\ (\mathcal{S}\cdot K\cdot Pi)$$

Substitution in Atomic Families

$$\frac{}{a[M_0/x_0]^f_{\rho_0} = a}\ (\mathcal{S}\cdot F\cdot Const) \qquad \frac{\alpha[M_0/x_0]^f_{\rho_0} = \alpha' \quad M[M_0/x_0]^O_{\rho_0} = M'}{(\alpha M)[M_0/x_0]^f_{\rho_0} = \alpha' M'}\ (\mathcal{S}\cdot F\cdot App)$$

Substitution in Canonical Families

$$\frac{\alpha[M_0/x_0]^f_{\rho_0} = \alpha'}{\alpha[M_0/x_0]^F_{\rho_0} = \alpha'}\ (\mathcal{S}\cdot F\cdot Atom) \qquad \frac{\sigma_1[M_0/x_0]^F_{\rho_0} = \sigma_1' \quad \sigma_2[M_0/x_0]^F_{\rho_0} = \sigma_2'}{(\Pi x{:}\sigma_1.\sigma_2)[M_0/x_0]^F_{\rho_0} = \Pi x{:}\sigma_1'.\sigma_2'}\ (\mathcal{S}\cdot F\cdot Pi)$$

$$\frac{\sigma_1[M_0/x_0]^F_{\rho_0} = \sigma_1' \quad M_1[M_0/x_0]^O_{\rho_0} = M_1' \quad \sigma_2[M_0/x_0]^F_{\rho_0} = \sigma_2'}{\mathcal{L}^{\mathcal{P}}_{M_1,\sigma_1}[\sigma_2][M_0/x_0]^F_{\rho_0} = \mathcal{L}^{\mathcal{P}}_{M_1',\sigma_1'}[\sigma_2']}\ (\mathcal{S}\cdot F\cdot Lock)$$

Figure 4: Hereditary substitution, kinds and families of $\mathsf{CLLF}_{\mathcal{P}}$

$\beta\eta$-normal form, which extends the standard one for lock-types. For one, the judgement $x{:}\Pi z{:}a.a \vdash_\Sigma x \Leftarrow \Pi z{:}a.a$ is not derivable, as $\Pi z{:}a.a$ is not atomic, hence $\vdash_\Sigma \lambda x{:}(\Pi z{:}a.a).x \Leftarrow \Pi x{:}(\Pi z{:}a.a).\Pi z{:}a.a$ is not derivable. On the other hand, $\vdash_\Sigma \lambda x{:}(\Pi z{:}a.a).\lambda y{:}a.xy \Leftarrow \Pi x{:}(\Pi z{:}a.a).\Pi z{:}a.a$, where a is a family constant of kind *Type*, is derivable. Analogously, for lock-types, the judgement $x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho] \vdash_\Sigma x \Leftarrow \mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]$ is not derivable, since $\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]$ is not atomic. As a consequence, we have that $\vdash_\Sigma \lambda x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho].x \Leftarrow \Pi x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho].\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]$ is not derivable. However, $x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho] \vdash_\Sigma \mathcal{L}^{\mathcal{P}}_{N,\sigma}[\mathcal{U}^{\mathcal{P}}_{N,\sigma}[x]] \Leftarrow \mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]$ is derivable, if ρ is atomic. Hence, the judgment $\vdash_\Sigma \lambda x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho].\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\mathcal{U}^{\mathcal{P}}_{N,\sigma}[x]] \Leftarrow \Pi x{:}\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho].\mathcal{L}^{\mathcal{P}}_{N,\sigma}[\rho]$ is derivable. Note that the unlock constructor takes an *atomic* term as its main argument, thus avoiding the creation of possible $\mathcal{L}$-redexes under substitution. Moreover, since unlocks can only receive locked terms in their body, no abstractions can ever arise. In Definition 2.3, we formalize the notion of η-expansion of a judgement, together with correspondence theorems between $\mathsf{LLF}_{\mathcal{P}}$ and $\mathsf{CLLF}_{\mathcal{P}}$.

We present $\mathsf{CLLF}_{\mathcal{P}}$ in a fully-typed style, *i.e. à la* Church, but we could also follow [13] and present a version *à la* Curry, where the canonical forms $\lambda x.M$ and $\mathcal{L}^{\mathcal{P}}_M[N]$ do not carry type information. The type rules would then be, *e.g.*:

$$\frac{\Gamma, x{:}\sigma \vdash_\Sigma M \Leftarrow \tau}{\Gamma \vdash_\Sigma \lambda x.M \Leftarrow \Pi x{:}\sigma.\tau}\ (O\cdot Abs) \qquad \frac{\Gamma \vdash_\Sigma M \Leftarrow \sigma \quad \Gamma \vdash_\Sigma N \Leftarrow \tau}{\Gamma \vdash_\Sigma \mathcal{L}^{\mathcal{P}}_M[N] \Leftarrow \mathcal{L}^{\mathcal{P}}_{M,\sigma}[\tau]}\ (O\cdot Lock)$$

This latter syntax is more suitable in implementations because it optimises the notation. Following [17], we stick to the typeful syntax because it allows for a more direct comparison with non-canonical systems. This, however, is technically immaterial. Since judgements in canonical systems have unique derivations, one can show by induction on derivations that any provable judgement in the system where object terms are *à la* Curry has a *unique* type decoration of its object subterms, which turns it into a provable judgement in the version *à la* Church. Vice versa, any provable judgement in the version *à la* Church can forget the types in its object subterms, yielding a provable judgement in the version *à la* Curry.

Substitution in Atomic Objects

$$\frac{}{c[M_0/x_0]^o_{\rho_0} = c}\,(\mathscr{S}\cdot O\cdot Const) \quad \frac{}{x_0[M_0/x_0]^o_{\rho_0} = M_0 : \rho_0}\,(\mathscr{S}\cdot O\cdot Var\cdot H) \quad \frac{x \neq x_0}{x[M_0/x_0]^o_{\rho_0} = x}\,(\mathscr{S}\cdot O\cdot Var)$$

$$\frac{A_1[M_0/x_0]^o_{\rho_0} = \lambda x{:}\rho_2.M_1' : \rho_2 \to \rho \quad M_2[M_0/x_0]^O_{\rho_0} = M_2' \quad M_1'[M_2'/x]^O_{\rho_2} = M'}{(A_1 M_2)[M_0/x_0]^o_{\rho_0} = M' : \rho}\,(\mathscr{S}\cdot O\cdot App\cdot H)$$

$$\frac{A_1[M_0/x_0]^o_{\rho_0} = A_1' \quad M_2[M_0/x_0]^O_{\rho_0} = M_2'}{(A_1 M_2)[M_0/x_0]^o_{\rho_0} = A_1' M_2'}\,(\mathscr{S}\cdot O\cdot App)$$

$$\frac{\sigma[M_0/x_0]^F_{\rho_0} = \sigma' \quad M[M_0/x_0]^O_{\rho_0} = M' \quad A[M_0/x_0]^o_{\rho_0} = \mathscr{L}^{\mathscr{P}}_{M',\sigma'}[M_1] : \mathscr{L}^{\mathscr{P}}_{M',\sigma'}[\rho]}{\mathscr{U}^{\mathscr{P}}_{M,\sigma}[A][M_0/x_0]^o_{\rho_0} = M_1 : \rho}\,(\mathscr{S}\cdot O\cdot Unlock\cdot H)$$

$$\frac{\sigma[M_0/x_0]^F_{\rho_0} = \sigma' \quad M[M_0/x_0]^O_{\rho_0} = M' \quad A[M_0/x_0]^o_{\rho_0} = A'}{\mathscr{U}^{\mathscr{P}}_{M,\sigma}[A][M_0/x_0]^o_{\rho_0} = \mathscr{U}^{\mathscr{P}}_{M',\sigma'}[A']}\,(\mathscr{S}\cdot O\cdot Unlock)$$

Substitution in Canonical Objects

$$\frac{A[M_0/x_0]^o_{\rho_0} = A'}{A[M_0/x_0]^O_{\rho_0} = A'}\,(\mathscr{S}\cdot O\cdot R) \quad \frac{A[M_0/x_0]^o_{\rho_0} = M' : \rho}{A[M_0/x_0]^O_{\rho_0} = M'}\,(\mathscr{S}\cdot O\cdot R\cdot H) \quad \frac{M[M_0/x_0]^O_{\rho_0} = M'}{\lambda x{:}\sigma.M[M_0/x_0]^O_{\rho_0} = \lambda x{:}\sigma.M'}\,(\mathscr{S}\cdot O\cdot Abs)$$

$$\frac{\sigma_1[M_0/x_0]^F_{\rho_0} = \sigma_1' \quad M_1[M_0/x_0]^O_{\rho_0} = M_1' \quad M_2[M_0/x_0]^O_{\rho_0} = M_2'}{\mathscr{L}^{\mathscr{P}}_{M_1,\sigma_1}[M_2][M_0/x_0]^O_{\rho_0} = \mathscr{L}^{\mathscr{P}}_{M_1',\sigma_1'}[M_2']}\,(\mathscr{S}\cdot O\cdot Lock)$$

Substitution in Contexts

$$\frac{}{[M_0/x_0]^C_{\rho_0} =}\,(\mathscr{S}\cdot Ctxt\cdot Empty) \quad \frac{x_0 \neq x \quad x \notin \mathsf{Fv}(M_0) \quad \Gamma[M_0/x_0]^C_{\rho_0} = \Gamma' \quad \sigma[M_0/x_0]^F_{\rho_0} = \sigma'}{(\Gamma, x{:}\sigma)[M_0/x_0]^C_{\rho_0} = \Gamma', x{:}\sigma'}\,(\mathscr{S}\cdot Ctxt\cdot Term)$$

Figure 5: Hereditary substitution, objects and contexts of $\mathsf{CLLF}_{\mathscr{P}}$

2.2 The Metatheory of $\mathsf{CLLF}_{\mathscr{P}}$

We start by studying the basic properties of hereditary substitution and the type system. First of all, we need to assume that the predicates are *well-behaved* in the sense of [18]. In the context of canonical systems, this notion needs to be rephrased as follows:

Definition 2.1 (Well-behaved predicates for canonical systems)**.** A finite set of predicates $\{\mathscr{P}_i\}_{i\in I}$ is *well-behaved* if each $\mathscr{P}$ in the set satisfies the following conditions:

1. ***Closure under signature and context weakening and permutation:***
   (a) If Σ and Ω are valid signatures such that $\Sigma \subseteq \Omega$ and $\mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)$, then $\mathscr{P}(\Gamma \vdash_\Omega N \Leftarrow \sigma)$.
   (b) If Γ and Δ are valid contexts such that $\Gamma \subseteq \Delta$ and $\mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)$, then $\mathscr{P}(\Delta \vdash_\Sigma N \Leftarrow \sigma)$.
2. ***Closure under hereditary substitution:*** If $\mathscr{P}(\Gamma, x{:}\sigma', \Gamma' \vdash_\Sigma N \Leftarrow \sigma)$ and $\Gamma \vdash_\Sigma N' : \sigma'$, then $\mathscr{P}(\Gamma, \Gamma'[N'/x]^C_{(\sigma')^-} \vdash_\Sigma N[N'/x]^O_{(\sigma')^-} \Leftarrow \sigma[N'/x]^F_{(\sigma')^-})$.

As canonical systems do not feature reduction, the "classical" third constraint for well-behaved predicates (closure under reduction) is not needed here. Moreover, the second condition (*closure under substitution*) becomes "closure under hereditary substitution".

Lemma 2.1 (Decidability of hereditary substitution)**.**

1. *For any T in $\{\mathscr{K}, \mathscr{A}, \mathscr{F}, \mathscr{O}, \mathscr{C}\}$, and any M, x, and ρ, it is decidable whether there exists a T' such that $T[M/x]^m_\rho = T'$ or there is no such T'.*
2. *For any M, x, ρ, and A, it is decidable whether there exists an A', such that $A[M/x]^o_\rho = A'$, or there exist M' and ρ', such that $A[M/x]^o_\rho = M' : \rho'$, or there are no such A' and M'.*

Lemma 2.2 (Head substitution size)**.** *If $A[M_0/x_0]^o_{\rho_0} = M{:}\rho$, then ρ is a subexpression of ρ_0.*

Proof. By induction on the derivation of $A[M_0/x_0]_{\rho_0} = M : \rho$. □

Lemma 2.3 (Uniqueness of substitution and synthesis)**.**

1. *It is not possible that $A[M_0/x_0]^o_{\rho_0} = A'$ and $A[M_0/x_0]^o_{\rho_0} = M{:}\rho$.*
2. *For any T, if $T[M_0/x_0]^m_{\rho_0} = T'$, and $T[M_0/x_0]^m_{\rho_0} = T''$, then $T' = T''$.*
3. *If $\Gamma \vdash_\Sigma \alpha \Rightarrow K$, and $\Gamma \vdash_\Sigma \alpha \Rightarrow K'$, then $K = K'$.*
4. *If $\Gamma \vdash_\Sigma A \Rightarrow \sigma$, and $\Gamma \vdash_\Sigma A \Rightarrow \sigma'$, then $\sigma = \sigma'$.*

Proof. From the definition of hereditary substitution and the $\mathsf{CLLF}_{\mathscr{P}}$ type system. □

Lemma 2.4 (Composition of hereditary substitution)**.** *Let $x \neq x_0$ and $x \notin \mathsf{Fv}(M_0)$. Then:*

1. *For all T_1' in $\{\mathscr{K}, \mathscr{F}_a, \mathscr{F}, \mathscr{O}_a, \mathscr{O}\}$, if $M_2[M_0/x_0]^O_{\rho_0} = M_2'$, $T_1[M_2/x]^m_{\rho_2} = T_1'$, and $T_1[M_0/x_0]^m_{\rho_0} = T_1''$, then there exists a T: $T_1'[M_0/x_0]^m_{\rho_0} = T$, and $T_1''[M_2'/x]^m_{\rho_2} = T$.*
2. *If $M_2[M_0/x_0]^O_{\rho_0} = M_2'$, $A_1[M_2/x]^o_{\rho_2} = M : \rho$, and $A_1[M_0/x_0]^o_{\rho_0} = A$, then there exists an M': $M[M_0/x_0]^O_{\rho_0} = M'$, and $A[M_2'/x]^o_{\rho_2} = M' : \rho$.*
3. *If $M_2[M_0/x_0]^O_{\rho_0} = M_2'$, $A_1[M_2/x]^o_{\rho_2} = A$, and $A_1[M_0/x_0]^o_{\rho_0} = M : \rho$, then there exists an M': $A[M_0/x_0]^o_{\rho_0} = M' : \rho$, and $M[M_2'/x]^O_{\rho_2} = M'$.*

By induction on derivations, similar to one in [13] p.14–15, we can prove:

Theorem 2.5 (Transitivity)**.** *Let Σ sig, $\vdash_\Sigma \Gamma, x_0{:}\rho_0, \Gamma'$ and $\Gamma \vdash_\Sigma M_0 \Leftarrow \rho_0$, and assume that all predicates are well-behaved. Then,*

1. *There exists a Γ'': $[M_0/x_0]^C_{\rho_0} = \Gamma''$ and $\vdash_\Sigma \Gamma, \Gamma''$.*
2. *If $\Gamma, x_0{:}\rho_0, \Gamma' \vdash_\Sigma K$ then there exists a K': $[M_0/x_0]^K_{\rho_0} K = K'$ and $\Gamma, \Gamma'' \vdash_\Sigma K'$.*
3. *If $\Gamma, x_0{:}\rho_0, \Gamma' \vdash_\Sigma \sigma$ type, then there exists a σ': $[M_0/x_0]^F_{\rho_0}\sigma = \sigma'$ and $\Gamma, \Gamma'' \vdash_\Sigma \sigma'$ type.*
4. *If $\Gamma, x_0{:}\rho_0, \Gamma' \vdash_\Sigma \sigma$ type and $\Gamma, x_0{:}\rho_0, \Gamma' \vdash_\Sigma M \Leftarrow \sigma$, then there exist σ' and M': $[M_0/x_0]^F_{\rho_0}\sigma = \sigma'$ and $[M_0/x_0]^O_{\rho_0} M = M'$ and $\Gamma, \Gamma'' \vdash_\Sigma M' \Leftarrow \sigma'$.*

Theorem 2.6 (Decidability of typing)**.** *If predicates in $\mathsf{CLLF}_{\mathscr{P}}$ are decidable, then all of the judgements of the system are decidable.*

Proof. By induction on the complexity of judgements. □

We can now precisely state the relationship between $\mathsf{CLLF}_{\mathscr{P}}$ and the $\mathsf{LLF}_{\mathscr{P}}$ system of [19]:

Theorem 2.7 (Soundness)**.** *For any predicate $\mathscr{P}$ of $\mathsf{CLLF}_{\mathscr{P}}$, we define a corresponding predicate in $\mathsf{LLF}_{\mathscr{P}}$ as follows: $\mathscr{P}(\Gamma \vdash_\Sigma M : \sigma)$ holds if and only if $\Gamma \vdash_\Sigma M : \sigma$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$ and $\mathscr{P}(\Gamma \vdash_\Sigma M \Leftarrow \sigma)$ holds in $\mathsf{CLLF}_{\mathscr{P}}$. Then, we have:*

1. *If Σ sig is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then Σ sig is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
2. *If $\vdash_\Sigma \Gamma$ is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\vdash_\Sigma \Gamma$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
3. *If $\Gamma \vdash_\Sigma K$ is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\Gamma \vdash_\Sigma K$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
4. *If $\Gamma \vdash_\Sigma \alpha \Rightarrow K$ is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\Gamma \vdash_\Sigma \alpha : K$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
5. *If $\Gamma \vdash_\Sigma \sigma$ type is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\Gamma \vdash_\Sigma \sigma : \mathsf{type}$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
6. *If $\Gamma \vdash_\Sigma A \Rightarrow \sigma$ is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\Gamma \vdash_\Sigma A : \sigma$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*
7. *If $\Gamma \vdash_\Sigma M \Leftarrow \sigma$ is derivable in $\mathsf{CLLF}_{\mathscr{P}}$, then $\Gamma \vdash_\Sigma M : \sigma$ is derivable in $\mathsf{LLF}_{\mathscr{P}}$.*

Vice versa, all $\mathsf{LLF}_{\mathscr{P}}$ judgements in *long* $\beta\eta$*-normal form* ($\beta\eta$-lnf) are derivable in $\mathsf{CLLF}_{\mathscr{P}}$. The definition of a judgement in $\beta\eta$-lnf is based on the following extension of the standard η-rule to the lock constructor $\lambda x{:}\sigma.Mx \to_\eta M$ and $\mathscr{L}^{\mathscr{P}}_{N,\sigma}[\mathscr{U}^{\mathscr{P}}_{N,\sigma}[M]] \to_\eta M$.

Definition 2.2. An occurrence ξ of a constant or a variable in a term of an $\mathsf{LLF}_{\mathscr{P}}$ judgement is *fully applied and unlocked* w.r.t. its type or kind $\Pi\overrightarrow{x}_1{:}\overrightarrow{\sigma}_1.\overrightarrow{\mathscr{L}}_1[\ldots\Pi\overrightarrow{x}_n{:}\overrightarrow{\sigma}_n.\overrightarrow{\mathscr{L}}_n[\alpha]\ldots]$, where $\overrightarrow{\mathscr{L}}_1,\ldots,\overrightarrow{\mathscr{L}}_n$ are vectors of locks, if ξ appears only in contexts that are of the form $\overrightarrow{\mathscr{U}}_n[(\ldots(\overrightarrow{\mathscr{U}}_1[\xi\overrightarrow{M}_1])\ldots)\overrightarrow{M}_n]$, where $\overrightarrow{M}_1,\ldots,\overrightarrow{M}_n$, $\overrightarrow{\mathscr{U}}_1,\ldots,\overrightarrow{\mathscr{U}}_n$ have the same arities of the corresponding vectors of Π's and locks.

Definition 2.3 (Judgements in long $\beta\eta$-normal form)**.**

1. A term T in a judgement is in $\beta\eta$*-lnf* if T is in normal form and every constant and variable occurrence in T is fully applied and unlocked w.r.t. its classifier in the judgement.
2. A judgement is in $\beta\eta$*-lnf* if all terms appearing in it are in $\beta\eta$-lnf.

Theorem 2.8 (Correspondence)**.** *Assume that all predicates in* $\mathsf{LLF}_{\mathscr{P}}$ *are well-behaved, according to Definition 2.1 [18]. For any predicate* $\mathscr{P}$ *in* $\mathsf{LLF}_{\mathscr{P}}$*, we define a corresponding predicate in* $\mathsf{CLLF}_{\mathscr{P}}$ *with:* $\mathscr{P}(\Gamma\vdash_\Sigma M \Leftarrow \sigma)$ *holds if* $\Gamma\vdash_\Sigma M \Leftarrow \sigma$ *is derivable in* $\mathsf{CLLF}_{\mathscr{P}}$ *and* $\mathscr{P}(\Gamma\vdash_\Sigma M : \sigma)$ *holds in* $\mathsf{LLF}_{\mathscr{P}}$*. Then, we have:*

1. *If* Σ sig *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* Σ sig *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
2. *If* $\vdash_\Sigma \Gamma$ *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\vdash_\Sigma \Gamma$ *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
3. *If* $\Gamma\vdash_\Sigma K$ *is in* $\beta\eta$*-lnf, and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\Gamma\vdash_\Sigma K$ *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
4. *If* $\Gamma\vdash_\Sigma \alpha : K$ *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\Gamma\vdash_\Sigma \alpha \Rightarrow K$ *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
5. *If* $\Gamma\vdash_\Sigma \sigma{:}\mathsf{type}$ *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\Gamma\vdash_\Sigma \sigma$ type *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
6. *If* $\Gamma\vdash_\Sigma A : \alpha$ *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\Gamma\vdash_\Sigma A \Rightarrow \alpha$ *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*
7. *If* $\Gamma\vdash_\Sigma M : \sigma$ *is in* $\beta\eta$*-lnf and is* $\mathsf{LLF}_{\mathscr{P}}$*-derivable, then* $\Gamma\vdash_\Sigma M \Leftarrow \sigma$ *is* $\mathsf{CLLF}_{\mathscr{P}}$*-derivable.*

Proof. Follows closely the proof of the corresponding Correspondence Theorem 7 [17]. □

Notice that, by the Correspondence Theorem above, any well-behaved predicate $\mathscr{P}$ in $\mathsf{LLF}_{\mathscr{P}}$ in the sense of Definition 2.1 [18] induces a well-behaved predicate in $\mathsf{CLLF}_{\mathscr{P}}$. Finally, notice that *not* all $\mathsf{LLF}_{\mathscr{P}}$ judgements have a corresponding $\beta\eta$-lnf. Namely, the judgement $x{:}\mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]\vdash_\Sigma x : \mathscr{L}^{\mathscr{P}}_{N,\sigma}[\rho]$ does not admit an η-expanded normal form when the predicate $\mathscr{P}$ does *not* hold on N, as the rule $(O{\cdot}Unlock)$ can be applied only when the predicate holds.

3 The Type System $\mathsf{CLLF}_{\mathscr{P}?}$

The main idea behind $\mathsf{CLLF}_{\mathscr{P}?}$ (see Figures 6, 7, and 8)[1] is to "empower" the framework of $\mathsf{CLLF}_{\mathscr{P}}$ by *adding* to the lock/unlock mechanism the possibility to receive from the external oracle a *witness* satisfying suitable constraints. Thus, we can pave way for the gluing together of different proof development environments beyond proof irrelevance scenarios. In this context, the lock constructor behaves as a *binder*. The new $(O{\cdot}Lock)$ rule is the following:

$$\frac{\Gamma, x{:}\sigma\vdash_\Sigma M \Leftarrow \rho}{\Gamma\vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho]}$$

[1]For lack of space, we present in these figures only the categories and rules of $\mathsf{CLLF}_{\mathscr{P}?}$ that differ from their $\mathsf{CLLF}_{\mathscr{P}}$ counterparts.

$$\begin{array}{rclrcll} \sigma,\tau,\rho & \in & \mathscr{F} & \sigma & ::= & \alpha \mid \Pi x{:}\sigma.\tau \mid \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] & \textit{Canonical Families} \\ M,N & \in & \mathscr{O} & M & ::= & A \mid \lambda x{:}\sigma.M \mid \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M] & \textit{Canonical Objects} \end{array}$$

Figure 6: $\mathsf{CLLF}_{\mathscr{P}?}$ Syntax — changes w.r.t. $\mathsf{CLLF}_{\mathscr{P}}$

Canonical Family rules

$$\frac{\Gamma, x{:}\sigma \vdash_\Sigma \rho \ \mathsf{type}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] \ \mathsf{type}} \ (F{\cdot}Lock)$$

$$\frac{\begin{array}{l}\Gamma, y:\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] \ \mathsf{type} \\ \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\tau] \\ \rho[\mathscr{U}^{\mathscr{P}}_{x,\sigma}[A]/y]^F_{(\tau)^-} = \rho'\end{array}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho'] \mathsf{type}} \ (F{\cdot}Nested{\cdot}Unlock)$$

Atomic Object rules

$$\frac{\begin{array}{ll}\Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] & \Gamma \vdash_\Sigma N \Leftarrow \sigma \\ \mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma) & \rho[N/x]^F_{(\sigma)^-} = \rho'\end{array}}{\Gamma \vdash_\Sigma \mathscr{U}^{\mathscr{P}}_{N,\sigma}[A] \Rightarrow \rho'} \ (O{\cdot}Unlock)$$

Canonical Object rules

$$\frac{\Gamma x{:}\sigma \vdash_\Sigma M \Leftarrow \rho}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho]} \ (O{\cdot}Lock)$$

$$\frac{\begin{array}{ll}\Gamma, y{:}\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] & \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\tau] \\ \rho[\mathscr{U}^{\mathscr{P}}_{x,\sigma}[A]/y]^F_{(\tau)^-} = \rho' & M[\mathscr{U}^{\mathscr{P}}_{x,\sigma}[A]/y]^O_{(\tau)^-} = M'\end{array}}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M'] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho']} \ (O{\cdot}Nested{\cdot}Unlock)$$

Figure 7: The $\mathsf{CLLF}_{\mathscr{P}?}$ Type System — changes w.r.t. $\mathsf{CLLF}_{\mathscr{P}}$

where the variable x is a placeholder bound in M and ρ, which will be replaced by the concrete term that will be returned by the external oracle call. The intuitive meaning behind the $(O{\cdot}Lock)$ rule is, therefore, that of recording the need to delegate to the external oracle the inference of a suitable witness of a given type. Indeed, M can be thought of as an "incomplete" term which needs to be completed by an inhabitant of a given type σ satisfying the constraint $\mathscr{P}$. The actual term, possibly synthesized by the external tool, will be "released" in $\mathsf{CLLF}_{\mathscr{P}?}$, by the unlock constructor in the $(O{\cdot}Unlock)$ rule as follows:

$$\frac{\Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] \quad \rho[N/x]^F_{(\sigma)^-} = \rho' \quad \Gamma \vdash_\Sigma N \Leftarrow \sigma \quad \mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)}{\Gamma \vdash_\Sigma \mathscr{U}^{\mathscr{P}}_{N,\sigma}[A] \Rightarrow \rho'}$$

The term $\mathscr{U}^{\mathscr{P}}_{N,\sigma}[M]$ intuitively means that N is precisely the synthesized term satisfying the constraint $\mathscr{P}(\Gamma \vdash_\Sigma N \Leftarrow \sigma)$ that will replace in $\mathsf{CLLF}_{\mathscr{P}?}$ all the free occurrences of x in ρ. This replacement is executed in the $(S{\cdot}O{\cdot}Unlock{\cdot}H)$ hereditary substitution rule (Figure 8).

Similarly to $\mathsf{CLLF}_{\mathscr{P}}$, in $\mathsf{CLLF}_{\mathscr{P}?}$ it is also possible to "postpone" or delay the verification of an external predicate. Whence, the synthesis of the actual inhabitant N can be delayed, thanks to the $(O{\cdot}Nested{\cdot}Unlock)$ rule:

$$\frac{\Gamma, y{:}\tau \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho] \quad \Gamma \vdash_\Sigma A \Rightarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\tau] \quad \rho[\mathscr{U}^{\mathscr{P}}_{x,\sigma}[A]/y]^F_{(\tau)^-} = \rho' \quad M[\mathscr{U}^{\mathscr{P}}_{x,\sigma}[A]/y]^O_{(\tau)^-} = M'}{\Gamma \vdash_\Sigma \mathscr{L}^{\mathscr{P}}_{x,\sigma}[M'] \Leftarrow \mathscr{L}^{\mathscr{P}}_{x,\sigma}[\rho']}$$

The Metatheory of $\mathsf{CLLF}_{\mathscr{P}?}$ follows closely that of $\mathsf{CLLF}_{\mathscr{P}}$ as far as decidability. We have no correspondence theorem since we did not introduce a non-canonical variant $\mathsf{CLLF}_{\mathscr{P}?}$. This could have been done similarly to $\mathsf{LLF}_{\mathscr{P}}$.

4 Case studies

In this section, we discuss the encodings of a collection of logical systems which illustrate the expressive power and the flexibility of $\mathsf{CLLF}_{\mathscr{P}}$ and $\mathsf{CLLF}_{\mathscr{P}?}$. We discuss Fitch-Prawitz Consistent Set theory, FPST

Substitution in Canonical Families

$$\frac{\sigma_1[M_0/x_0]^F_{\rho_0} = \sigma_1' \quad \sigma_2[M_0/x_0]^F_{\rho_0} = \sigma_2'}{\mathscr{L}^{\mathscr{P}}_{x,\sigma_1}[\sigma_2][M_0/x_0]^F_{\rho_0} = \mathscr{L}^{\mathscr{P}}_{x,\sigma_1'}[\sigma_2']} \ (\mathscr{S}{\cdot}F{\cdot}Lock)$$

Substitution in Atomic Objects

$$\frac{\sigma[M_0/x_0]^F_{\rho_0} = \sigma' \quad M[M_0/x_0]^o_{\rho_0} = M' \quad M_1[M'/x]^o_{(\sigma')^-} = M_2 \quad A[M_0/x_0]^o_{\rho_0} = \mathscr{L}^{\mathscr{P}}_{x,\sigma'}[M_1] : \mathscr{L}^{\mathscr{P}}_{x,\sigma'}[\rho]}{\mathscr{U}^{\mathscr{P}}_{M,\sigma}[A][M_0/x_0]^o_{\rho_0} = M_2 : \rho} \ (\mathscr{S}{\cdot}O{\cdot}Unlock{\cdot}H)$$

Substitution in Canonical Objects

$$\frac{\sigma_1[M_0/x_0]^F_{\rho_0} = \sigma_1' \quad M_1[M_0/x_0]^O_{\rho_0} = M_1'}{\mathscr{L}^{\mathscr{P}}_{x,\sigma_1}[M_1][M_0/x_0]^O_{\rho_0} = \mathscr{L}^{\mathscr{P}}_{x,\sigma_1'}[M_1']} \ (\mathscr{S}{\cdot}O{\cdot}Lock)$$

Figure 8: CLLF$_{\mathscr{P}?}$ Hereditary Substitution — changes w.r.t. CLLF$_{\mathscr{P}}$

[29], some applications of FPST to normalizing λ-calculus, a system of Light Linear Logic in CLLF$_{\mathscr{P}}$, and an the encoding of a *partial* function in CLLF$_{\mathscr{P}?}$.

The crucial step in encoding a logical system in CLLF$_{\mathscr{P}}$ or CLLF$_{\mathscr{P}?}$ is to define the predicates involved in locks. Predicates defined on closed terms are usually unproblematic. Difficulties arise in enforcing the properties of closure under hereditary substitution and closure under signature and context extension, when predicates are defined on open terms. To be able to streamline the definition of well-behaved predicates we introduce the following:

Definition 4.1. Given a signature Σ let Λ_Σ (respectively Λ^o_Σ) be the set of LLF$_{\mathscr{P}}$ terms (respectively *closed* LLF$_{\mathscr{P}}$ terms) definable using constants from Σ. A term M has a *skeleton* in Λ_Σ if there exists a term $N[x_1,\dots,x_n] \in \Lambda_\Sigma$, whose free variables (called *holes* of the skeleton) are in $\{x_1,\dots,x_n\}$, and there exist terms $M_1,\dots,M_n$ such that $M \equiv N[M_1/x_1,\dots,M_n/x_n]$.

4.1 Fitch Set Theory *à la* Prawitz - FPST

In this section, we present the encoding of a formal system of remarkable logical as well as historical significance, namely the system of consistent *Naïve* Set Theory, FPST, introduced by Fitch [10]. This system was first presented in Natural Deduction style by Prawitz [29]. As Naïve Set Theory is inconsistent, to prevent the derivation of inconsistencies from the unrestricted *abstraction* rule, only normalizable *deductions* are allowed in FPST. Of course, this side-condition is extremely difficult to capture using traditional tools.

In the present context, instead, we can put to use the machinery of CLLF$_{\mathscr{P}}$ to provide an appropriate encoding of FPST where the *global* normalization constraint is enforced *locally* by checking the proof-object. This system is a beautiful illustrative example of the *bag of tricks* that CLLF$_{\mathscr{P}}$ supports. Checking that a proof term is normalizable would be the obvious predicate to use in the corresponding lock-type, but this would not be a well-behaved predicate if free variables, *i.e.* assumptions, are not sterilized. To this end, we introduce a distinction between *generic* judgements, which cannot be directly utilized in arguments, but which can be assumed, and *apodictic* judgements, which are directly involved in proof rules. In order to make use of generic judgements, one has to downgrade them to an apodictic one. This is achieved by a suitable coercion function.

Definition 4.2 (Fitch Prawitz Set Theory, FPST)**.** The full system of Fitch (see [29]), as presented by Prawitz is defined into Appendix A, here we only give the crucial rules for implication and for *set-*

abstraction and the corresponding elimination rules:

$$\frac{\Gamma, A \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} A \supset B}\,(\supset I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} A \quad \Gamma \vdash_{\mathsf{FPST}} A \supset B}{\Gamma \vdash_{\mathsf{FPST}} B}\,(\supset E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}{\Gamma \vdash_{\mathsf{FPST}} T \in \lambda x.A}\,(\lambda I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} T \in \lambda x.A}{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}\,(\lambda E)$$

In Fitch's system, FPST, conjunction and universal quantification are defined as usual, while negation is defined constructively, but it still allows for the usual definitions of disjunction and existential quantification. What makes FPST *consistent* is that not all standard deductions in FPST are legal. Standard deductions are called *quasi-deductions* in FPST. A *legal deduction* in FPST is defined instead, as a quasi-deduction which is *normalizable* in the standard sense of Natural Deduction, namely it can be transformed in a derivation where all elimination rules occur before introductions.

Definition 4.3 (LLF$_{\mathscr{P}}$ signature Σ_{FPST} for Fitch Prawitz Set Theory)**.** The following constants are introduced:

`o`	`: Type`	ι	`: Type`
`T`	`: o -> Type`	δ	$: \Pi$`A:o. (V(A) -> T(A))`
`V`	`: o -> Type`	λ`_intro`	$: \Pi$`A:`ι` ->o.`Π`x:`ι`.T(A x) -> T(`ε` x (lam A))`
`lam`	`: (`ι` -> o)-> `ι	λ`_elim`	$: \Pi$`A:`ι` ->o.`Π`x:`ι`.T(`ε` x (lam A))->T(A x)`
ε	`: `ι` -> `ι` -> o`	$\supset$`_intro`	$: \Pi$`A,B:o.(V(A) -> T(B)) -> (T(A `$\supset$`B))`
$\supset$	`: o -> o -> o`	$\supset$`_elim`	$: \Pi\mathtt{A,B{:}o}.\Pi\mathtt{x{:}T(A)}.\Pi\mathtt{y{:}T(A{\supset}B)}\ \mathtt{->}\ \mathscr{L}^{\mathtt{Fitch}}_{\langle\mathtt{x,y}\rangle,\mathtt{T(A)}\times\mathtt{T(A\supset B)}}[\mathtt{T(B)}]$

where `o` is the type of propositions, $\supset$ and the "membership" predicate ε are the syntactic constructors for propositions, `lam` is the "abstraction" operator for building "sets", `T` is the apodictic judgement, `V` is the generic judgement, δ is the coercion function, and $\langle \mathtt{x}, \mathtt{y}\rangle$ denotes the encoding of pairs, whose type is denoted by $\sigma\times\tau$, *e.g.* $\lambda \mathtt{u}{:}\sigma \to \tau \to \rho.\ \mathtt{u\ x\ y} : (\sigma \to \tau \to \rho) \to \rho$. The predicate in the lock is defined as follows:

$$\mathtt{Fitch}(\Gamma \vdash_{\Sigma_{\mathsf{FPST}}} \langle \mathtt{x}, \mathtt{y}\rangle \ \Leftarrow\ \mathtt{T(A)}\times\mathtt{T(A \supset B)})$$

it holds iff `x` and `y` have skeletons in $\Lambda_{\Sigma_{\mathsf{FPST}}}$, all the holes of which have either type `o` or are guarded by a δ, and hence have type `V(A)`, and, moreover, the proof derived by combining the skeletons of `x` and `y` is normalizable in the natural sense.

For lack of space, we do not spell out the rules concerning the other logical operators, because they are all straightforward provided we use only the apodictic judgement $\mathtt{T}(\cdot)$, but a few remarks are mandatory. The notion of *normalizable proof* is the standard notion used in natural deduction. The predicate `Fitch` is well-behaved because it considers terms only up-to holes in the skeleton, which can have type `o` or are generic judgements. Adequacy for this signature can be achieved in the format of [18]:

Theorem 4.1 (Adequacy for Fitch-Prawitz Naive Set Theory)**.** *If $A_1,\ldots,A_n$ are the atomic formulas occurring in $B_1,\ldots,B_m,A$, then $B_1 \ldots B_m \vdash_{\mathsf{FPST}} A$ iff there exists a normalizable* `M` *such that* $\mathtt{A_1{:}o},\ldots,\mathtt{A_n{:}o}$, $\mathtt{x_1{:}V(B_1)},\ldots,\mathtt{x_m{:}V(B_m)} \vdash_{\Sigma_{\mathsf{FPST}}} \mathtt{M} \Leftarrow \mathtt{T(A)}$ *(where* `A`*, and* $\mathtt{B_i}$ *represent the encodings of, respectively, A and B_i in* CLLF$_{\mathscr{P}}$*, for $1 \le i \le m$).*

4.2 A Type System for strongly normalizing λ-terms

Fitch-Prawitz Set Theory, FPST, is a rather intriguing, albeit unexplored, set theoretic system. The normalizability criterion for accepting a quasi-deduction prevents the derivation of contradictions, and hence makes the system consistent. Of course, some intuitive rules are not derivable. For instance *modus ponens* does not hold and if $t \in \lambda x.A$ then we do not have necessarily that $A[t/x]$ holds. Similarly, the *transitivity* property does not hold. However, it is interesting to point out that FPST is also a very expressive type system which encompasses all kinds of quantification.

In this subsection, we sketch how to use FPST to define a type system which can type *precisely all* the strongly normalizing λ-terms. First we need to adapt to FPST, Proposition 4, Appendix A.1 of [12], originally given by J-Y. Girard for Light Linear Logic, as follows:

Theorem 4.2 (Fixpoint)**.** *Let $A[P,x_1 \dots, x_n]$ be a formula of* FPST *with an n-ary predicate variable P. Then, there exists a formula B of* FPST*, such that there exists a normalizable deduction in* FPST *between $A[\lambda x_1 \dots, x_n.B[x_1, \dots, x_n], x_1 \dots, x_n]$ and B, and viceversa.*

Proof. Let equality be defined as Leibniz equality, then, assuming $n = 1$, define $\Lambda \equiv \lambda z.\exists x.\exists y.z = \langle x,y\rangle \& A[(\lambda w.\langle w,y\rangle \in y), x]$. Then $\langle x,\Lambda\rangle \in \Lambda$ is equivalent, in the sense of FPST, to $A[(\lambda w.\langle w,\Lambda\rangle \in \Lambda), x]$. □

The above Fixpoint Theorem can be used to define recursive functions and inductive definitions. Hence, let us consider a concrete representation, possibly over natural numbers, of the terms of λ-calculus in FPST, and call it Λ_0. It exists, because even if the theory is non-standard, it allows the definition of some putative notion of a natural number. Using again the Fixed Point Theorem, we can define a fixpoint Λ such that $x \in \Lambda$ is equivalent, in the sense of Fitch, to $x \in \Lambda_0 \& \forall y.y \in \Lambda_0 \subset app(x,y) \in \Lambda$. Here, $app(x,y)$ denotes the concrete representation of "applying" x to y. It is now straightforward to check that only normalizing terms can "be typed in FPST by Λ", *i.e.* belong to Λ. Since one can derive in FPST that a representation of a λ-term, say M, belongs to Λ, only if there is a normalizable derivation of $M \in \Lambda$, then there is a natural reflection of the normalizability of the FPST derivation of the typing judgement $M \in \Lambda$, and the fact that the term represented by M is indeed normalizable.

4.3 A Normalizing call-by-value λ-calculus

In this section we sketch how to express in $\mathsf{CLLF}_{\mathscr{P}}$ a call-by-value λ-calculus where β-reductions fire only if the result is *normalizing*.

Definition 4.4 (Normalizing call-by-value λ-calculus, $\Sigma_{\lambda N}$)**.**

```
o      : Type        Eq  : o -> o -> Type       app : o -> o -> o
v      : Type        var : v -> o               lam : (v -> o) -> o
```

c_beta : ΠM:o->o,N:o.$\mathcal{L}^{\mathscr{P}^{\mathtt{N}}}_{\langle \mathtt{M},\mathtt{N}\rangle,\mathtt{(o\text{->}o)}\times\mathtt{o}}$[Eq (app (lam λx:v.M(var x)) N) (MN)]

where the predicate $\mathscr{P}^{\mathtt{N}}$ holds on $\Gamma \vdash_{\Sigma_{\lambda N}} \langle \mathtt{M},\mathtt{N}\rangle \Leftarrow \mathtt{(o\text{->}o)}\times\mathtt{o}$ if both `M` and `N` have skeletons in $\Lambda_{\Sigma_{\lambda N}}$ whose holes are guarded by a `var` and, moreover, `MN` "normalizes", in the intuitive sense, outside terms guarded by a `var`.

4.4 Elementary Affine Logic

In this section we give a *shallow* encoding of *Elementary Affine Logic* as presented in [2]. This example will exemplify how locks can be used to deal with global syntactic constraints as in the *promotion rule* of Elementary Affine Logic.

Definition 4.5 (Elementary Affine Logic [2])**.** Elementary Affine Logic can be specified by the following rules:

$$\frac{}{A \vdash_{EAL} A}\ (Var) \qquad \frac{\Gamma \vdash_{EAL} B}{\Gamma, A \vdash_{EAL} B}\ (Weak) \qquad \frac{\Gamma, A \vdash_{EAL} B}{\Gamma \vdash_{EAL} A \multimap B}\ (Abst)$$

$$\frac{\Gamma \vdash_{EAL} A \quad \Delta \vdash_{EAL} A \multimap B}{\Gamma, \Delta \vdash_{EAL} B}\ (Appl) \qquad \frac{\Gamma \vdash_{EAL} !A \quad \Delta, !A, \ldots, !A \vdash_{EAL} B}{\Gamma, \Delta \vdash_{EAL} B}\ (Contr)$$

$$\frac{A_1, \ldots, A_n \vdash_{EAL} A \quad \Gamma_1 \vdash_{EAL} !A_1 \quad \ldots \quad \Gamma_n \vdash_{EAL} !A_n}{\Gamma_1 \ldots \Gamma_n \vdash_{EAL} !A}\ (Prom)$$

Definition 4.6 ($\mathsf{LLF}_{\mathscr{P}}$ signature Σ_{EAL} for Elementary Affine Logic)**.**

$\texttt{o : Type} \quad \texttt{T : o -> Type} \quad \texttt{V : o -> Type} \quad \multimap \texttt{ : o -> o -> o} \quad \texttt{! : o -> o}$

$\texttt{c\_appl} \ : \Pi\texttt{A,B :o. T(A) -> T(A} \multimap \texttt{B)-> T(B)} \qquad \texttt{c\_val} : \Pi\texttt{A:o. V(A) -> T(!A)}$

$\texttt{c\_abstr} \ : \Pi\texttt{A,B :o.}\ \Pi\texttt{x:(T(A) -> T(B)) ->}\ \mathscr{L}^{\mathit{Light}}_{\texttt{x,T(A)->T(B)}}[\texttt{T(A} \multimap \texttt{B)}]$

$\texttt{c\_promV\_1} : \Pi\texttt{A,B :o.}\ \Pi\texttt{x:(T(A} \multimap \texttt{B)) ->}\ \mathscr{L}^{\mathit{Closed}}_{\texttt{x,T(A}\multimap\texttt{ B)}}[\texttt{T(!A) -> V(B)}]$

$\texttt{c\_promV\_2} : \Pi\texttt{A,B :o.}\ \Pi\texttt{x:(V(A} \multimap \texttt{B)) ->}\ \mathscr{L}^{\mathit{Closed}}_{\texttt{x,V(A}\multimap\texttt{ B)}}[\texttt{T(!A) -> V(B)}]$

where $\texttt{o}$ is the type of propositions, $\multimap$ and ! are the obvious syntactic constructors, $\texttt{T}$ is the basic judgement, and $\texttt{V}(\cdot)$ is an auxiliary judgement. The predicates involved in the locks are defined as follows:

- $\mathit{Light}(\Gamma \vdash_{\Sigma_{EAL}} \texttt{x} \Leftarrow \texttt{T(A)} \to \texttt{T(B)})$ holds iff if $\texttt{A}$ is not of the shape $\texttt{!A}$ then the bound variable of $\texttt{x}$ occurs at most once in the normal form of $\texttt{x}$.
- $\mathit{Closed}(\Gamma \vdash_{\Sigma_{EAL}} \texttt{x} \Leftarrow \texttt{T(A)})$ holds iff the skeleton of x contains only free variables of type o, *i.e.*no variables of type $\texttt{T(B)}$, for any $\texttt{B} : \texttt{o}$.

A few remarks are mandatory. The promotion rule in [2] is in effect a *family* of natural deduction rules with a growing number of assumptions. Our encoding achieves this via the auxiliary judgement $\texttt{V}(\cdot)$, the effect of which is self-explanatory. Adequacy for this signature can be achieved only in the format of [18], namely:

Theorem 4.3 (Adequacy for Elementary Affine Logic)**.** *if $A_1, \ldots, A_n$ are the atomic formulas occurring in $B_1, \ldots, B_m, A$, then $B_1 \ldots B_m \vdash_{EAL} A$ iff there exists* $\texttt{M}$ *and* $\texttt{A}_1\texttt{:o}, \ldots, \texttt{A}_n\texttt{:o}, \texttt{x}_1\texttt{: T(B}_1\texttt{)}, \ldots, \texttt{x}_m\texttt{:T(B}_m\texttt{)} \vdash_{\Sigma_{EAL}} \texttt{M} \Leftarrow \texttt{T(A)}$ *(where* $\texttt{A}$*, and* $\texttt{B}_\texttt{i}$ *represent the encodings of, respectively, A and B_i in* $\mathsf{CLLF}_{\mathscr{P}}$*, for $1 \leq i \leq m$) and all variables* $\texttt{x}_1 \ldots \texttt{x}_m$ *occurring more than once in* $\texttt{M}$ *have type of the shape* $\texttt{T(B}_\texttt{i}\texttt{)} \equiv \texttt{T(!C}_\texttt{i}\texttt{)}$ *for some suitable formula* $\texttt{C}_i$*.*

The check on the context of the Adequacy Theorem is *external* to the system $\mathsf{LLF}_{\mathscr{P}}$, but this is in the nature of results which relate *internal* and *external* concepts. For example, the very concept of $\mathsf{LLF}_{\mathscr{P}}$ context, which appears in any adequacy result, is external to $\mathsf{LLF}_{\mathscr{P}}$. Of course, this check is internalized if the term is closed.

4.5 Square roots of natural numbers in $\mathsf{CLLF}_{\mathscr{P}?}$

It is well-known that logical frameworks based on Constructive Type Theory do not permit definitions of non-terminating functions (*i.e.*, all the functions one can encode in such frameworks are total). One interesting example of $\mathsf{CLLF}_{\mathscr{P}?}$ system is the possibility of reasoning about partial functions by delegating their computation to external oracles, and getting back their possible outputs, via the lock-unlock mechanism of $\mathsf{CLLF}_{\mathscr{P}?}$.

For instance, we can encode natural numbers and compute their square roots by means of the following signature ($\langle \texttt{x}, \texttt{y} \rangle$ denotes the encoding of pairs, whose type is denoted by $\sigma \times \tau$, and `fst` and `snd` are the first and second projections, respectively):

$\texttt{nat: type} \quad \texttt{0: nat} \quad \texttt{S: nat->nat} \quad \texttt{plus : nat->nat->nat} \quad \texttt{minus : nat->nat->nat}$

$\texttt{mult} \ \texttt{: nat->nat->nat} \qquad \texttt{sqroot: nat->nat} \qquad \texttt{eval} \ \texttt{: nat->nat->type}$

$\texttt{sqrt} \ : \Pi\text{x:nat.}\mathscr{L}^{\mathit{SQRT}}_{\texttt{y,nat}\times\sigma}[(\texttt{eval (sqroot x) (fst y)})]$

where `eval` represents the usual evaluation predicate, the variable `y` is a pair and

$$\sigma \equiv (\texttt{eval}\ (\texttt{plus}\ (\texttt{minus}\ \texttt{x}\ (\texttt{mult}\ \texttt{z}\ \texttt{z}))\ (\texttt{minus}\ (\texttt{mult}\ \texttt{z}\ \texttt{z})\ \texttt{x})\ \texttt{0}))$$

and $SQRT(\Gamma \vdash_\Sigma \texttt{y} \Leftarrow \texttt{nat} \times \sigma)$ holds if and only if the first projection of `y` is the minimum number `N` such that $(\texttt{x} \mathbin{\dot{-}} \texttt{N} * \texttt{N}) + (\texttt{N} * \texttt{N} \mathbin{\dot{-}} \texttt{x}) = 0$, where $+$ and $*$ are represented by `plus` and `mult`, while $\dot{-}$ (represented by `minus` in our signature) is defined as follows:

$$\texttt{x} \mathbin{\dot{-}} \texttt{y} \stackrel{\Delta}{=} \begin{cases} \texttt{x} - \texttt{y} & \texttt{if}\ \texttt{x} \geq \texttt{y} \\ 0 & \texttt{otherwise} \end{cases}$$

Thus, the specification of `sqroot` is not explicit in $\mathsf{CLLF}_{\mathscr{P}?}$, since it is implicit in the definition of *SQRT*.

5 Related work

Building a universal framework with the aim of "gluing" different tools and formalisms together is a long standing goal that has been extensively explored in the inspiring work on Logical Frameworks by [4, 26, 32, 6, 25, 27, 28, 16]. Moreover, the appealing monadic structure and properties of the lock/unlock mechanism go back to Moggi's notion of computational monads [24]. Indeed, our system can be seen as a generalization to a family of dependent *lax* operators of Moggi's *partial* λ-calculus [23] and of the work carried out in [7, 22] (which is also the original source of the term "lax"). A correspondence between lax modalities and monads in functional programming was pointed out in [1, 11]. On the other hand, although the connection between constraints and monads in logic programming was considered in the past, *e.g.*, in [25, 9, 8], to our knowledge, our systems are the first attempt to establish a clear correspondence between side conditions and monads in a *higher-order dependent-type theory* and in logical frameworks. Of course, there are a lot of interesting points of contact with other systems in the literature which should be explored. For instance, in [25], the authors introduce a contextual modal logic, where the notion of context is rendered by means of monadic constructs. We only point out that, as we did in our system, they could have also simplified their system by doing away with the `let` construct in favor of a deeper substitution. Schröder-Heister has discussed in a number of papers, see *e.g.* [31, 30], various restrictions and side conditions on rules and on the nature of assumptions that one can add to logical systems to prevent the arising of paradoxes. There are some potential connections between his work and ours. It would be interesting to compare his requirements on side conditions being "closed under substitution" to our notion of *well-behaved* predicate. Similarly, there are commonalities between his distinction between *specific* and *unspecific* variables, and our treatment of free variables in well-behaved predicates.

References

[1] N. Alechina, M. Mendler, V. De Paiva, E. Ritter. Categorical and Kripke semantics for constructive s4 modal logic. In *Computer Science Logic*, pp. 292–307. Springer, 2001.

[2] P. Baillot, P. Coppola, U. Dal Lago. Light logics and optimal reduction: Completeness and complexity. In *LICS*, pp. 421–430. IEEE Computer Society, 2007.

[3] H.P. Barendregt, E. Barendsen. Autarkic computations in formal proofs. *Journal of Automated Reasoning*, 28:321–336, 2002.

[4] G. Barthe, H. Cirstea, C. Kirchner, L. Liquori. Pure Pattern Type Systems. In *POPL'03*, pp. 250–261, ACM.

[5] R. Boulton, A. Gordon, M. Gordon, J. Harrison, J. Herbert, J. Van Tassel. Experience with embedding hardware description languages in HOL. In *TPCD*, pp. 129–156. North-Holland, 1992.

[6] D. Cousineau, G. Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In *Proc. TLCA*, vol. 4583 of *LNCS*, pp. 102–117. Springer-Verlag, 2007.

[7] M. Fairtlough, M. Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, 1997.

[8] M. Fairtlough, M. Mendler, X. Cheng. Abstraction and refinement in higher order logic. In *Theorem Proving in Higher Order Logics*, pp. 201–216. Springer, 2001.

[9] M Fairtlough, M. Mendler, M. Walton. First-order lax logic as a framework for constraint logic programming. Technical report, 1997.

[10] F. B. Fitch. *Symbolic logic*. New York, 1952.

[11] D. Garg, M. C. Tschantz. From indexed lax logic to intuitionistic logic. Tech. rep. CMU, 2008.

[12] J.-Y. Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

[13] R. Harper, D. Licata. Mechanizing metatheory in a logical framework. *JFP*, 17:613–673, 2007.

[14] D. Hirschkoff. Bisimulation proofs for the π-calculus in the Calculus of Constructions. In *Proc. TPHOL'97*, number 1275 in LNCS. Springer-Verlag, 1997.

[15] F. Honsell. 25 years of formal proof cultures: Some problems, some philosophy, bright future. In *Proc. of LFMTP'13*, pp. 37–42, New York, NY, USA, 2013. ACM.

[16] F. Honsell, M. Lenisa, L. Liquori. A Framework for Defining Logical Frameworks. *Volume in Honor of G. Plotkin, ENTCS*, 172:399–436, 2007.

[17] F. Honsell, M. Lenisa, L. Liquori, P. Maksimovic, I. Scagnetto. $\mathsf{LF}_{\mathscr{P}}$: a logical framework with external predicates. In *Proc. of LFMTP*, pp. 13–22. ACM, 2012.

[18] F. Honsell, M. Lenisa, L. Liquori, P. Maksimovic, I. Scagnetto. An open logical framework. Accepted for publication in *Journal of Logic and Computation*, doi: 10.1093/logcom/ext028.

[19] F. Honsell, L. Liquori, P. Maksimovic, I. Scagnetto. $\mathsf{LLF}_{\mathscr{P}}$: A Logical Framework for modeling External Evidence, Side Conditions, and Proof Irrelevance using Monads. Available at `http://www.dimi.uniud.it/scagnett/LLFP_LMCS.pdf`.

[20] F. Honsell, L. Liquori, I. Scagnetto. $\mathsf{L}^{\mathsf{ax}}\mathsf{F}$: Side Conditions and External Evidence as Monads. In *Proc. of MFCS 2014, Part I*, vol. 8634 of *LNCS*, pp. 327–339, Budapest, Hungary, August 2014. Springer.

[21] F. Honsell, M. Miculan, I. Scagnetto. π-calculus in (Co)Inductive Type Theories. *Theoretical Computer Science*, 253(2):239–285, 2001.

[22] M. Mendler. Constrained proofs: A logic for dealing with behavioral constraints in formal hardware verification. In *Designing Correct Circuits*, pp. 1–28. Springer-Verlag, 1991.

[23] E. Moggi. *The partial lambda calculus*. PhD thesis, University of Edinburgh, 1988.

[24] E. Moggi. Computational lambda-calculus and monads. In *Proc. of LICS 1989*, pp. 14–23. IEEE Press.

[25] A. Nanevski, F. Pfenning, B. Pientka. Contextual Modal Type Theory. *ACM TOCL*, 9(3), 2008.

[26] F. Pfenning, C. Schürmann. System description: Twelf – a meta-logical framework for deductive systems. In *Proc. CADE*, v. 1632 of *LNCS*, pp. 202–206. Springer-Verlag, 1999.

[27] B. Pientka, J. Dunfield. Programming with proofs and explicit contexts. In *PPDP'08*, pp. 163–173, ACM.

[28] B. Pientka, J. Dunfield. Beluga: A framework for programming and reasoning with deductive systems (system description). In *Automated Reasoning*, v. 6173 of *LNCS*, pp. 15–21. Springer-Verlag, 2010.

[29] D. Prawitz. *Natural Deduction. A Proof Theoretical Study*. Almqvist Wiksell, Stockholm, 1965.

[30] P. Schroeder-Heister. Paradoxes and Structural Rules. *Insolubles and consequences : essays in honor of Stephen Read*, pp. 203–211. College Publications, London, 2012.

[31] P. Schroeder-Heister. Proof-theoretic semantics, self-contradiction, and the format of deductive reasoning. *Topoi*, 31(1):77–85, 2012.

[32] K. Watkins, I. Cervesato, F. Pfenning, D. Walker. A Concurrent Logical Framework I: Judgments and Properties. Tech. Rep. CMU-CS-02-101, CMU, 2002.

A Fitch Set Theory

Definition A.1 (Fitch Prawitz Set Theory)**.** The full system of Fitch (see [29]), as presented by Prawitz is defined as follows:

$$\frac{\Gamma, A \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} A \supset B}\ (\supset I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} A \quad \Gamma \vdash_{\mathsf{FPST}} A \supset B}{\Gamma \vdash_{\mathsf{FPST}} B}\ (\supset E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}{\Gamma \vdash_{\mathsf{FPST}} T \in \lambda x.A}\ (\lambda I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} T \in \lambda x.A}{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}\ (\lambda E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A \quad \Gamma \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} A \& B}\ (\& I) \quad \frac{\Gamma \vdash_{\mathsf{FPST}} A \& B}{\Gamma \vdash_{\mathsf{FPST}} A}\ (\& E1) \quad \frac{\Gamma \vdash_{\mathsf{FPST}} A \& B}{\Gamma \vdash_{\mathsf{FPST}} B}\ (\& E2)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A}{\Gamma \vdash_{\mathsf{FPST}} A \vee B}\ (\vee I1) \quad \frac{\Gamma \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} A \vee B}\ (\vee I2) \quad \frac{\Gamma \vdash_{\mathsf{FPST}} A \vee B \quad \begin{array}{c}\Gamma, A \vdash_{\mathsf{FPST}} C \\ \Gamma, B \vdash_{\mathsf{FPST}} C\end{array}}{\Gamma \vdash_{\mathsf{FPST}} C}\ (\vee E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A}{\Gamma \vdash_{\mathsf{FPST}} \forall x A[x/a]}\ (\forall I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \forall x A}{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}\ (\forall E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A[T/x]}{\Gamma \vdash_{\mathsf{FPST}} \exists x A}\ (\exists I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \exists x A \quad \Gamma, A[a/x] \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} B}\ (\exists E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} \curlywedge}{\Gamma \vdash_{\mathsf{FPST}} A}\ (\curlywedge_I) \qquad \frac{\Gamma, \neg A \vdash_{\mathsf{FPST}} \curlywedge}{\Gamma \vdash_{\mathsf{FPST}} A}\ (\curlywedge_C)$$

As noticed in [29], negation in Fitch's system is usually defined in a constructive way as follows:

$$\frac{\Gamma \vdash_{\mathsf{FPST}} \neg A}{\Gamma \vdash_{\mathsf{FPST}} \neg(A \& B)}\ (\neg \& I1) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \neg B}{\Gamma \vdash_{\mathsf{FPST}} \neg(A \& B)}\ (\neg \& I2)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} \neg(A \& B) \quad \begin{array}{c}\Gamma, \neg A \vdash_{\mathsf{FPST}} C \\ \Gamma, \neg B \vdash_{\mathsf{FPST}} C\end{array}}{\Gamma \vdash_{\mathsf{FPST}} C}\ (\neg \& E) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} A \quad \Gamma \vdash_{\mathsf{FPST}} \neg B}{\Gamma \vdash_{\mathsf{FPST}} \neg(A \supset B)}\ (\neg \supset I)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} \neg(A \supset B)}{\Gamma \vdash_{\mathsf{FPST}} A}\ (\neg \supset E1) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \neg(A \supset B)}{\Gamma \vdash_{\mathsf{FPST}} \neg B}\ (\neg \supset E2)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} A}{\Gamma \vdash_{\mathsf{FPST}} \neg\neg A}\ (\neg\neg I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \neg\neg A}{\Gamma \vdash_{\mathsf{FPST}} A}\ (\neg\neg E)$$

$$\frac{\Gamma \vdash_{\mathsf{FPST}} \neg A[T/x]}{\Gamma \vdash_{\mathsf{FPST}} \neg\forall x A}\ (\neg\forall I) \qquad \frac{\Gamma \vdash_{\mathsf{FPST}} \neg\forall x A \quad \Gamma, \neg A[a/x] \vdash_{\mathsf{FPST}} B}{\Gamma \vdash_{\mathsf{FPST}} B}\ (\neg\forall E)$$