Higher-Order Superposition for Dependent Types *

Roberto Virga

Carnegie Mellon University, Pittsburgh PA 15213, USA

Abstract. We describe a proof of the Critical Pair Lemma for Plotkin's LF calculus [4]. Our approach basically follows the one used by Nipkow [12] for the simply-typed case, though substantial modifications and some additional theoretical machinery are needed to ensure well-typedness of rewriting in this richer type system. We conclude the paper presenting some significant applications of the theory.

1 Overview

Higher-order rewriting has always attracted considerable interest for the potentially wide range of application it has on several fields of theoretical computer science, ranging from program transformation [5] to automated theorem proving [13]. Until recently, however, undecidability of higher-order unification discouraged any attempt to lift at least some the most interesting results of the first-order theory to the higher-order setting.

One of the first steps in this direction, to the author's knowledge, is due to Wolfram, who in [22] offers a first definition of rewriting in the simply-typed lambda calculus, and suggests how other related concepts can also be similarly generalized to this framework. However, no indication is given as what additional restrictions it is necessary to impose to overcome the undecidability results and to make these notions of practical interest.

The most important result, from a computational standpoint, may be considered a paper by Nipkow, published the same year [12]: by imposing some restrictions to the form of rules, he was is able to prove an analogue of the Critical Pair Lemma. Nipkow's work has inspired a lot of activity the area of higher-order rewriting, and contributed to a series of important results, such as Prehofer's work on higher-order narrowing [18], or Loria's study of conditional higher-order rewriting [8].

In this paper we extend Nipkow's proof to the LF calculus; here the possibility of indexing types by terms, while providing enormous flexibility of representation, it also creates considerable problems. In particular, unlike the simply-typed case, replacing an expression with another of the same type may in some cases produce an ill-typed overall expression. For this reason we introduce the notion of *(type) dependency relation*, and use it to obtain a formulation of rewriting which preserves well-typedness of expressions.

* This work was supported by NSF Grant CCR-9303383.

The paper is organized as follows: section two is used to introduce some basic terminology; section three defines the notions of higher-order rewriting and congruence modulo a set of (higher-order) rules, and explores the relation between the two; in section four we give a definition of critical pair, and use it to prove a version of the Critical Pair Lemma for the calculus. We show some applications of rewriting with dependent types in section five, and we finally conclude by briefly mentioning some promising directions of future research. Due to the excessive complexity, as well as lack of space, proofs will be omitted in most cases; the results presented are however proved in full detail in [21].

2 Preliminaries

In order to make this paper more self-contained, we briefly introduce Plotkin's LF calculus; an interested reader may refer to [14] for a more exhaustive presentation.

Definition 1. The LF calculus is a three-level calculus for *terms*, *type families*, and *kinds*

$$\begin{array}{ll} \textit{Kinds} & K := \text{type} \mid \Pi x : A.K \\ \textit{Families} & A := a \mid \Pi x : A.B \mid AM \\ \textit{Terms} & M := c \mid x \mid \lambda x : A.M \mid MN \end{array}$$

In the following, K denotes kinds, A, B families, M, N terms; a will be used to indicate type constants, c for constants at the level of terms, x, y, z for variables.

We will make use of the usual notions of α, β and η-reductions. These, although defined on terms, extend naturally by congruence to type families and kinds. All objects will be considered equal modulo α-conversion. We will denote by $\equiv$ the smallest equivalence relation including α-, β-, and η-reductions.

By $[N/x]M$ ($[N/x]A$, $[N/x]K$ respectively) we intend, as usual, the replacement of all the free occurrences of x by N inside M (A, K, respectively). As usual, α-conversion will be used, if necessary, to ensure the that no free variable occurrence is captured inside the scope of a quantifier.

The notation $\mathcal{FV}(E)$ and $\mathcal{BV}(E)$ is used to denote the set of free and bound variables, respectively, in E, where E is a term, a type family or a kind.

Definition 2. To define the class of well-typed kinds, type families, and terms we make use of signatures and contexts:

$$\begin{array}{ll} \textit{Signatures} & \Sigma := \cdot \mid \Sigma, a : K \mid \Sigma, c : A \\ \textit{Contexts} & \Gamma := \cdot \mid \Gamma, x : A \end{array}$$

We will use Γ and Δ to range over contexts.

Well-formed expressions are then constructed accordingly to the judgements

$$\Gamma \vdash_\Sigma M : A$$
$$\Gamma \vdash_\Sigma A : K$$
$$\Gamma \vdash_\Sigma K \; Kind$$

These, in turn, are defined in terms of the auxiliary judgements

$$\vdash \Sigma \; Sig$$
$$\vdash_\Sigma \Gamma \; Ctx$$

which specify how valid signatures and contexts are formed.

The rules for the calculus are listed below:

$$\frac{\Sigma(c)=A}{\Gamma\vdash_\Sigma c{:}A} \qquad \frac{\Gamma(x)=A}{\Gamma\vdash_\Sigma x{:}A} \qquad \frac{\Gamma\vdash_\Sigma M{:}A \quad A\equiv A' \quad \Gamma\vdash_\Sigma A'{:}\text{type}}{\Gamma\vdash_\Sigma M{:}A'}$$

$$\frac{\Gamma\vdash_\Sigma M{:}\Pi x{:}A.B \quad \Gamma\vdash_\Sigma N{:}A}{\Gamma\vdash_\Sigma MN{:}[N/x]B} \qquad \frac{\Gamma\vdash_\Sigma A{:}\text{type} \quad \Gamma,x{:}A\vdash_\Sigma M{:}B}{\Gamma\vdash_\Sigma \lambda x{:}A.M{:}\Pi x{:}A.B}$$

$$\frac{\Sigma(a)=K}{\Gamma\vdash_\Sigma a{:}K} \qquad \frac{\Gamma\vdash_\Sigma A{:}K \quad K\equiv K' \quad \Gamma\vdash_\Sigma K' \; Kind}{\Gamma\vdash_\Sigma A{:}K'}$$

$$\frac{\Gamma\vdash_\Sigma A{:}\Pi x{:}B.K \quad \Gamma\vdash_\Sigma M{:}B}{\Gamma\vdash_\Sigma AM{:}[M/x]K} \qquad \frac{\Gamma\vdash_\Sigma A{:}\text{type} \quad \Gamma,x{:}A\vdash_\Sigma B{:}\text{type}}{\Gamma\vdash_\Sigma \Pi x{:}A.B{:}\text{type}}$$

$$\frac{}{\Gamma\vdash_\Sigma \text{type} \; Kind} \qquad \frac{\Gamma\vdash_\Sigma A{:}\text{type} \quad \Gamma,x{:}A\vdash_\Sigma K \; Kind}{\Gamma\vdash_\Sigma \Pi x{:}A.K \; Kind}$$

$$\frac{}{\vdash_\Sigma \cdot \; Ctx} \qquad \frac{\vdash_\Sigma \Gamma \; Ctx \quad \Gamma\vdash_\Sigma A{:}\text{type}}{\vdash_\Sigma \Gamma,x{:}A \; Ctx}$$

$$\frac{}{\vdash \cdot \; Sig}$$

$$\frac{\vdash_\Sigma K \; Kind \quad \vdash \Sigma \; Sig}{\vdash \Sigma,a{:}K \; Sig} \qquad \frac{\vdash_\Sigma A{:}\text{type} \quad \Gamma\vdash \Sigma \; Sig}{\vdash \Sigma,c{:}A \; Sig}$$

We will use $M\,\overline{N}$ to denote the repeated application $M\,N_1\,N_2\ldots N_n$; similarly for type families. The notation $[\overline{N}/\overline{x}]$ will stand for the repeated replacement $[N_n/x_n]\ldots[N_1/x_1]$.

Differently from the simply-typed lambda-calculus, in the LF calculus replacing a subterm with another of the same type inside a term may affect the type of the overall expression. This is illustrated by the following:

Example 1. Consider the following representation of a fragment of arithmetic:

$$\begin{aligned} \mathbf{nat} &: \text{type} \\ \\ \mathbf{0} &: \mathbf{nat} \\ \mathbf{s} &: \mathbf{nat} \Rightarrow \mathbf{nat} \\ \mathbf{+} &: \mathbf{nat} \Rightarrow (\mathbf{nat} \Rightarrow \mathbf{nat}) \end{aligned}$$

where we used the notation $A \Rightarrow B$ and $A \Rightarrow K$ for the abstractions $\Pi x : A.B$ and $\Pi x : A.K$, respectively, where $x \notin \mathcal{FV}(B)$ and $x \notin \mathcal{FV}(K)$.

We want now to formalize the (first-order) predicate "n is even", together with some inference rules that allow us to decide if a number is even:

$$\begin{aligned}
\mathbf{o} &: \text{type} \\
\mathbf{proof} &: \mathbf{o} \Rightarrow \text{type} \\
\mathbf{even} &: \mathbf{nat} \Rightarrow \mathbf{o} \\
\\
\mathbf{even}_0 &: \mathbf{proof}(\mathbf{even}\ \mathbf{0}) \\
\mathbf{even}_{ss} &: \Pi x : \mathbf{nat}.\ (\mathbf{proof}(\mathbf{even}\ x) \Rightarrow \mathbf{proof}(\mathbf{even}\ (\mathbf{s}\ (\mathbf{s}\ x)))) \\
\mathbf{even}_{+} &: \Pi x : \mathbf{nat}.\ \Pi y : \mathbf{nat}.\ (\mathbf{proof}(\mathbf{even}\ x) \Rightarrow (\mathbf{proof}(\mathbf{even}\ y) \Rightarrow \mathbf{proof}(\mathbf{even}\ (\mathbf{+}\ x\ y)))) \\
\mathbf{even}_{s+} &: \Pi x : \mathbf{nat}.\ \Pi y : \mathbf{nat}.\ (\mathbf{proof}(\mathbf{even}\ (\mathbf{+}\ x\ y)) \Rightarrow \mathbf{proof}(\mathbf{even}\ (\mathbf{+}\ (\mathbf{s}\ x)\ (\mathbf{s}\ y))))
\end{aligned}$$

According to this signature, the term

$$\mathbf{even}_{+}\ \mathbf{0}\ (\mathbf{+}\ \mathbf{0}\ \mathbf{0})\ \mathbf{even}_0\ (\mathbf{even}_{+}\ \mathbf{0}\ \mathbf{0}\ \mathbf{even}_0\ \mathbf{even}_0)$$

is well-typed, but rewriting $(\mathbf{+}\ \mathbf{0}\ \mathbf{0})$ into $\mathbf{0}$ (both of type $\mathbf{nat}$) yields

$$\mathbf{even}_{+}\ \mathbf{0}\ \mathbf{0}\ \mathbf{even}_0\ (\mathbf{even}_{+}\ \mathbf{0}\ \mathbf{0}\ \mathbf{even}_0\ \mathbf{even}_0)$$

which is not.

In defining a notion of rewriting, we must therefore be careful to rule out all these pathological cases leading to ill-typed expressions.

From a close analysis, it appears that all these cases derive from a common problem, namely that, by use of the application rule

$$\frac{\Gamma \vdash_\Sigma M : \Pi x{:}A.B \quad \Gamma \vdash_\Sigma N : A}{\Gamma \vdash_\Sigma M\ N : [N/x]B},$$

the rewritten term is propagated inside a type. Hence, a way to ensure well-typedness is to allow rewriting only inside those terms whose type does not contain terms which can be, in turn, rewritten. The notion of type dependency relation provides a clean and formal way to extract this information from a LF signature.

Definition 3. Define

$$\text{head}(\Pi x_1 : A_1 \ldots \Pi x_n : A_n.a\,\overline{M}) = a,$$

let Σ_0 be a signature, a pair $\prec_0 = (\prec_0^A, \prec_0^M)$ of binary transitive relations over the set of type constants of Σ_0 is called a *dependency relation* if it satisfies the following conditions:

- $a_i \prec_0^A a$ if $\Sigma_0(a) = \Pi x_1 : A_1 \ldots \Pi x_n : A_n.\text{type}$, $\text{head}(A_i) = a_i$, $1 \leq i \leq n$;
- $a \prec_0^A a'$ if, for some b, $a \prec_0^A b \prec_0^M a'$ or $a \prec_0^M b \prec_0^A a'$;

- $a \prec_0^M b$ if $a \prec_0^A b$;
- $\vdash^{\prec_0} \Sigma_0\ Sig$;

where $\vdash^{\prec_0} \Sigma\ Sig$ is defined recursively using the judgements

$$\begin{array}{l} \Gamma \vdash^{\prec_0}_{\Sigma} M : A \\ \Gamma \vdash^{\prec_0}_{\Sigma} A : K \\ \Gamma \vdash^{\prec_0}_{\Sigma} K\ Kind \\ \quad\vdash^{\prec_0}_{\Sigma} \Gamma\ Ctx \end{array}$$

by an inference system very similar to the one for general LF calculus, but with an additional side condition in the rules for abstraction:

$$\frac{\Gamma \vdash^{\prec_0}_{\Sigma} A{:}\text{type} \quad \Gamma, x{:}A \vdash^{\prec_0}_{\Sigma} B{:}\text{type}}{\Gamma \vdash^{\prec_0}_{\Sigma} \Pi x{:}A.B{:}\text{type}}* \qquad \frac{\Gamma \vdash^{\prec_0}_{\Sigma} A{:}\text{type} \quad \Gamma, x{:}A \vdash^{\prec_0}_{\Sigma} M{:}B}{\Gamma \vdash^{\prec_0}_{\Sigma} \lambda x{:}A.M{:}\Pi x{:}A.B}*$$

* $\text{head}(A) \prec_0^M \text{head}(B)$ or $\text{head}(A) = \text{head}(B)$

By abuse of notation, given two type families A, B, we will write $A \prec^A B$ and $A \prec^M B$ for $\text{head}(A) \prec^A \text{head}(B)$ and $\text{head}(A) \prec^M \text{head}(B)$, respectively. We will use $A \preceq^M_\Sigma B$ to say that $A \prec^M B$ or $head(A) = head(B)$.

The idea underlying the introduction of the relations $\prec^A$ and $\prec^M$ is to restrict, using the $\vdash^{\preceq}_{\Sigma}$ judgements, the generation of valid terms and type families to those which preserve the dependencies satisfied by the signature Σ; in particular, we want terms of type A to be allowed to appear inside B only if $A \prec^A B$, and, similarly, terms of type A will be subterms of terms of type B only if $A \preceq^M B$.

When looking for a dependency relation, we will usually prefer coarser ones, so that the class of dependency-preserving terms (i.e. terms well typed according to the $\vdash^{\preceq}_{\Sigma}$ judgement) is as wide as possible. In practice, given a derivation of $\vdash \Sigma\ Sig$, we will try to determine the least $\prec$ such that $\vdash^{\prec} \Sigma\ Sig$ holds.

Example 1. In our previous example about even numbers, the following is easily seen to be a dependency relation:

$$\prec\ = (\{\mathbf{nat} \prec^A \mathbf{proof}, \mathbf{o} \prec^A \mathbf{proof}\}, \{\mathbf{nat} \prec^M \mathbf{o}, \mathbf{nat} \prec^A \mathbf{proof}, \mathbf{o} \prec^M \mathbf{proof}\})$$

The condition $\mathbf{o} \prec^A \mathbf{proof}$ comes from the type of **proof**; $\mathbf{nat} \prec^M \mathbf{o}$ is obtained from type checking on **even**; finally $\mathbf{nat} \prec^A \mathbf{proof}$ since $\prec^A \supseteq \prec^M \cdot \prec^A$. All the other pairs in $\prec^M$ follow from $\prec^M \supseteq \prec^A$.

From now on, we will assume that a signature Σ and a dependency relation $\prec$ for it have been fixed. Also sometimes, when mentioning a contex Γ we will tacitly assume it is well-typed and dependency-preserving, i.e. $\vdash^{\preceq}_{\Sigma} \Gamma\ Ctx$.

Most basic properties of the LF calculus relativize to dependency-preserving terms. We list a few of them:

Proposition 4 (Substitution). *Let $\Gamma \vdash^{\preceq}_{\Sigma} N : C$, then:*

1. *if $\Gamma, y : C, \Delta \vdash^{\preceq}_{\Sigma} M : A$ then $\Gamma, [N/y]\Delta \vdash^{\preceq}_{\Sigma} [N/y]M : [N/y]A$;*

2. *if* $\Gamma, y : C, \Delta \vdash^{\prec}_{\Sigma} A : K$ *then* $\Gamma, [N/y]\Delta \vdash^{\prec}_{\Sigma} [N/y]A : [N/y]K$*;*
3. *if* $\Gamma, y : C, \Delta \vdash^{\prec}_{\Sigma} K\ Kind$ *then* $\Gamma, [N/y]\Delta \vdash^{\prec}_{\Sigma} [N/y]K\ Kind$*.*

Proposition 5 (Weakening). *Let* $\Sigma' \subseteq \Sigma$*,* $\Gamma' \subseteq \Gamma$*, and* $\vdash^{\prec}_{\Sigma} \Gamma\ Ctx$*.*

1. *If* $\Gamma' \vdash^{\prec}_{\Sigma'} M : A$ *then* $\Gamma \vdash^{\prec}_{\Sigma} M : A$*.*
2. *If* $\Gamma' \vdash^{\prec}_{\Sigma'} A : K$ *then* $\Gamma \vdash^{\prec}_{\Sigma} A : K$*.*
3. *If* $\Gamma' \vdash^{\prec}_{\Sigma'} K\ Kind$ *then* $\Gamma \vdash^{\prec}_{\Sigma} K\ Kind$*.*

Proposition 6. *If* $\Gamma \vdash^{\prec}_{\Sigma} M : A$ *and* N *is a subterm of* M*, then there is* $\Gamma' \supseteq \Gamma$ *and type* A' *such that* $\Gamma' \vdash^{\prec}_{\Sigma} N : A'$*.*

In the rest of this paper, we will write $\Gamma(M, N)$ and $A(M, N)$ for the context Γ' and type A', respectively, obtained by the Proposition above.

The following result clarifies the motivating property of the two relations $\prec^{A}$ and $\prec^{M}$:

Lemma 7. *Let* $\vdash^{\prec}_{\Sigma} \Gamma, x : C, \Delta\ Ctx$*,*

1. *if* $\Gamma, x : C, \Delta \vdash^{\prec}_{\Sigma} A : K$ *and* $x \in \mathcal{FV}(A)$ *then* $C \prec^{A} A$*;*
2. *if* $\Gamma, x : C, \Delta \vdash^{\prec}_{\Sigma} M : A$ *and* $x \in \mathcal{FV}(M)$ *then* $C \preceq^{M}_{\Sigma} A$*.*

Definition 8. Environments are expressions with a "hole", which we will denote by $\circ$, constructed according to the following syntax:

$$\textit{Environments} \qquad E := \circ \mid \lambda x : A.E \mid M\ E \mid E\ N$$

Well-typed environments are constructed by means of the judgement

$$\Gamma \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : A,$$

and the rules

$$\frac{\Gamma_{\circ} \vdash^{\prec}_{\Sigma} A_{\circ}\text{:type} \quad \Gamma_{\circ} \subseteq \Gamma}{\Gamma \vdash^{\prec}_{\Sigma} \circ\ [\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : A_{\circ}} \qquad \frac{\Gamma \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : A \quad A \equiv B \quad \Gamma \vdash^{\prec}_{\Sigma} B\text{:type}}{\Gamma \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : B}$$

$$\frac{\Gamma \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : \Pi x{:}A.B \quad \Gamma \vdash^{\prec}_{\Sigma} N : A}{\Gamma \vdash^{\prec}_{\Sigma} (E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!])\ N : [N/x]B}$$

$$\frac{\Gamma \vdash^{\prec}_{\Sigma} M : \Pi x{:}A.B \quad \Gamma \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : A}{\Gamma \vdash^{\prec}_{\Sigma} M\ (E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!]) : B} \quad A_{\circ} \not\prec^{A} B$$

$$\frac{\Gamma \vdash^{\prec}_{\Sigma} A\text{:type} \quad \Gamma, x{:}A \vdash^{\prec}_{\Sigma} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : B}{\Gamma \vdash^{\prec}_{\Sigma} \lambda x{:}A.E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : \Pi x{:}A.B} \quad A \preceq^{M}_{\Sigma} B$$

The side condition $A_{\circ} \not\prec^{A} B$ in the application rule enforces our requirement that, whenever we substitute a term for the hole, the effect of this substitution does not propagate to the type.

Example 1. The environment

$$\mathbf{even}_+\ \mathbf{0}\ (\circ\ [\![\vdash \circ : \mathbf{nat}]\!]),$$

and therefore also

$$\mathbf{even}_+\ \mathbf{0}\ (\circ\ [\![\vdash \circ : \mathbf{nat}]\!])\ (\mathbf{even}_+\ \mathbf{0}\ \mathbf{0}\ \mathbf{even}_0\ \mathbf{even}_0)$$

are not well-typed. This because in the application

$$\frac{\vdash^{\preceq}_{\Sigma} \mathbf{even}_+\ \mathbf{0}{:}\Pi y{:}\mathbf{nat}.(\mathbf{proof}(\mathbf{even}\ 0)\Rightarrow(\mathbf{proof}(\mathbf{even}\ y)\Rightarrow\mathbf{proof}(\mathbf{even}\ (+\ 0\ y)))) \quad \vdash^{\preceq}_{\Sigma} \circ\ [\![\vdash\circ{:}\mathbf{nat}]\!]{:}\mathbf{nat}}{\vdash^{\preceq}_{\Sigma} \mathbf{even}_+\ \mathbf{0}\ (\circ\ [\![\vdash\circ{:}\mathbf{nat}]\!]){:}\mathbf{proof}(\mathbf{even}\ 0)\Rightarrow(\mathbf{proof}(\mathbf{even}\ \circ)\Rightarrow\mathbf{proof}(\mathbf{even}\ (+\ 0\ \circ)))}$$

the side condition $\mathbf{nat} \not\preceq^{A} \mathbf{proof}$ is violated.

Given an environment E and a term M, we will write $E[\![M]\!]$ for the term obtained by replacing the $\circ$ with M.

As expected, when the hole is replaced by an expression of compatible type, environments always produce well-typed expressions:

Lemma 9. *If $\Gamma \vdash^{\preceq}_{\Sigma} E[\![\Gamma_\circ \vdash \circ : A_\circ]\!] : A$, and $\Delta \vdash^{\preceq}_{\Sigma} M : A_\circ$ with $\Delta \subseteq \Gamma_\circ$, then $\Gamma \vdash^{\preceq}_{\Sigma} E[\![M]\!] : A$.*

In [9], the definition of substitution makes use of the existence and uniqueness of long $\beta\eta$ normal forms. In the LF calculus, these find an analogue in the concept of canonical form:

Definition 10. We define canonical forms for terms and type families by the judgements

$$\begin{array}{rl} \Gamma \vdash_\Sigma M \Downarrow A & M \text{ is canonical of type } A \\ \Gamma \vdash_\Sigma A \Downarrow \text{type} & A \text{ is a canonical type} \end{array}$$

$$\begin{array}{rl} \Gamma \vdash_\Sigma M \downarrow A & M \text{ is atomic of type } A \\ \Gamma \vdash_\Sigma A \downarrow K & A \text{ is atomic of kind } K \end{array}$$

formed according to the following inference rules:

$$\frac{\Gamma\vdash_\Sigma A \Downarrow \text{type} \quad \Gamma,x{:}A\vdash_\Sigma M \Downarrow B}{\Gamma\vdash_\Sigma \lambda x{:}A.M \Downarrow \Pi x{:}A.B} \qquad \frac{\Gamma\vdash_\Sigma M \Downarrow A \quad A\equiv B \quad \Gamma\vdash_\Sigma B{:}\text{type}}{\Gamma\vdash_\Sigma M \Downarrow B}$$

$$\frac{\Gamma\vdash_\Sigma A \downarrow \text{type} \quad \Gamma\vdash_\Sigma M \downarrow A}{\Gamma\vdash_\Sigma M \Downarrow A}$$

$$\frac{\Sigma(c)=A}{\Gamma\vdash_\Sigma c \downarrow A} \qquad \frac{\Gamma(x)=A}{\Gamma\vdash_\Sigma x \downarrow A} \qquad \frac{\Gamma\vdash_\Sigma M \downarrow A \quad A\equiv B \quad \Gamma\vdash_\Sigma B{:}\text{type}}{\Gamma\vdash_\Sigma M \downarrow B}$$

$$\frac{\Gamma\vdash_\Sigma M \downarrow \Pi x{:}A.B \quad \Gamma\vdash_\Sigma N \Downarrow A}{\Gamma\vdash_\Sigma M\ N \downarrow [N/x]B}$$

$$\frac{\Gamma\vdash_\Sigma A \Downarrow \text{type} \quad \Gamma,x{:}A\vdash_\Sigma B \Downarrow \text{type}}{\Gamma\vdash_\Sigma \Pi x{:}A.B \Downarrow \text{type}} \qquad \frac{\Gamma\vdash_\Sigma A \downarrow \text{type}}{\Gamma\vdash_\Sigma A \Downarrow \text{type}}$$

$$\frac{\Sigma(a){=}K}{\Gamma{\vdash_\Sigma}a\downarrow K} \qquad \frac{\Gamma{\vdash_\Sigma}A\downarrow K \quad K{\equiv}K' \quad \Gamma{\vdash_\Sigma}K'\ Kind}{\Gamma{\vdash_\Sigma}A\downarrow K'}$$

$$\frac{\Gamma{\vdash_\Sigma}A\downarrow \Pi x{:}B.K \quad \Gamma{\vdash_\Sigma}M \Downarrow B}{\Gamma{\vdash_\Sigma}A\ M\downarrow[M/x]K}$$

We will make use of the abbreviations:

$$\Gamma \vdash^{\preceq}_{\Sigma} M \Downarrow A \overset{\text{def}}{\iff} \Gamma \vdash_{\Sigma} M \Downarrow A \text{ and } \Gamma \vdash^{\preceq}_{\Sigma} M : A$$

$$\Gamma \vdash^{\preceq}_{\Sigma} A \Downarrow \text{type} \overset{\text{def}}{\iff} \Gamma \vdash_{\Sigma} A \Downarrow \text{type and } \Gamma \vdash^{\preceq}_{\Sigma} A : \text{type}$$

and similarly for $\Gamma \vdash^{\preceq}_{\Sigma} M \downarrow A$ and $\Gamma \vdash^{\preceq}_{\Sigma} A \downarrow K$.

The following holds:

Theorem 11. *Let $\vdash_\Sigma \Gamma\ Ctx$.*

1. *If $\Gamma \vdash_\Sigma M : A$ then there is a unique M' such that $M' \equiv M$ and $\Gamma \vdash_\Sigma M' \Downarrow A$.*
2. *If $\Gamma \vdash_\Sigma A : \text{type}$ then there is a unique A' such that $A \equiv A'$ and $\Gamma \vdash_\Sigma A' \Downarrow \text{type}$.*

Proof. These facts can be proved for generic LF expressions, i.e. when $\vdash_\Sigma$ instead of $\vdash^{\preceq}_{\Sigma}$ is considered; see [1, 3, 20]. In [21] it is shown that they relativize to dependencies.

Given a well-typed term M or a type family A, we will denote their canonical form by $M_{\Downarrow}$ and $A_{\Downarrow}$, respectively.

Definition 12. Given two contexts Γ and Δ, a substitution from Γ to Δ is a type-preserving, finite-support mapping from variables to terms $\theta : \Gamma \rightharpoondown \Delta$ formed according to the following rules:

$$\frac{}{\cdot{:}\cdot\rightharpoondown\Delta} \qquad \frac{\{\overline{x}{\mapsto}\overline{M}\}{:}\Gamma{\rightharpoondown}\Delta \quad \Delta{\vdash_\Sigma}N{\Downarrow}[\overline{M}/\overline{x}]A}{\{\overline{x}{\mapsto}\overline{M},y{\mapsto}N\}{:}(\Gamma,y{:}A){\rightharpoondown}\Delta}$$

Similarly, dependencies-preserving substitution are also defined:

$$\frac{}{\cdot{:}\cdot\overset{\preceq}{\rightharpoondown}\Delta} \qquad \frac{\{\overline{x}{\mapsto}\overline{M}\}{:}\Gamma\overset{\preceq}{\rightharpoondown}\Delta \quad \Delta{\vdash^{\preceq}_\Sigma}N{\Downarrow}[\overline{M}/\overline{x}]A}{\{\overline{x}{\mapsto}\overline{M},y{\mapsto}N\}{:}(\Gamma,y{:}A)\overset{\preceq}{\rightharpoondown}\Delta}$$

Definition 13. Given any well-typed term $\Gamma \vdash_\Sigma M : A$ and substitution $\theta = \{\overline{x} \mapsto \overline{N}\} : \Gamma \rightharpoondown \Delta$, define θM to be the (unique) canonical form of

$$\Delta \vdash_\Sigma (\lambda\overline{x} : \overline{B}.M)\ \overline{N} : [\overline{N}/\overline{x}]A$$

Note that here, in analogy to [9], we define the result of a substitution application to be a canonical term.

Definition 14.

1. Given two substitutions $\theta_1 = \{\overline{x} \mapsto \overline{M}\} : \Gamma_1 \rightharpoondown \Gamma_2$ and $\theta_2 : \Gamma_2 \rightharpoondown \Gamma_3$, the *composition* $\theta_2 \circ \theta_1$ is the substitution $\{\overline{x} \mapsto \overline{\theta_2 M}\} : \Gamma_1 \rightharpoondown \Gamma_3$.
2. A substitution $\theta = \{\overline{x} \mapsto \overline{M}\} : \Gamma \rightharpoondown \Delta$ is a *renaming* if all the terms M_i are (convertible to) distinct variables.
3. A substitution $\theta_1 : \Gamma \rightharpoondown \Delta$ is said to be *more general* than $\theta_2 : \Gamma \rightharpoondown \Delta'$ if there is $\rho : \Delta \rightharpoondown \Delta'$ such that $\theta_2 = \rho \circ \theta_1$.
4. Given two well typed terms $\Gamma \vdash_\Sigma M : A$ and $\Gamma \vdash_\Sigma N : A'$, a substitution $\theta : \Gamma \rightharpoondown \Delta$ is said to be a *unifier* of M and N if $\theta M = \theta N$; M and N are then said to unify.

The class of dependency-preserving terms is closed with respect to substitution:

Proposition 15. *If $\Gamma \vdash^{\preceq}_{\Sigma} M : A$ and $\theta = \{\overline{x} \mapsto \overline{N}\} : \Gamma \overset{\preceq}{\rightharpoondown} \Delta$ then $\Delta \vdash^{\preceq}_{\Sigma} \theta M : [\overline{N}/\overline{x}]A$.*

Corollary 16. *If $\theta_1 : \Gamma_1 \overset{\preceq}{\rightharpoondown} \Gamma_2$ and $\theta_2 : \Gamma_2 \overset{\preceq}{\rightharpoondown} \Gamma_3$, then $\theta_2 \circ \theta_1 : \Gamma_1 \overset{\preceq}{\rightharpoondown} \Gamma_3$.*

One particular class of terms for which unification is decidable are patterns [10]; the definition derives, almost unchanged, from the simply-typed case [15].

Definition 17. A canonical term $\Gamma \vdash_\Sigma M \Downarrow A$ is said to be a pattern if each $x \in \operatorname{dom} \Gamma$ can appear in M and A only applied to terms η-equivalent to distinct bound variables.

Theorem 18. *Unification of patterns is decidable; if two patterns unify, there is a unique (up to conversion) most general unifier.*

Proof. See [15].

3 Higher-Order Term Rewriting

In this section we extend the notion of term rewriting system and rewriting relation to a higher-order setting with dependent types.

Definition 19. A rewrite rule $\Gamma \vdash^{\preceq}_{\Sigma} l \rightarrow r : A$ is a pair of well typed terms such that

- $\Gamma \vdash^{\preceq}_{\Sigma} l \Downarrow A$ is a pattern, $\Gamma \vdash^{\preceq}_{\Sigma} r : A$,
- $\Gamma \vdash^{\preceq}_{\Sigma} A \downarrow \text{type}$,
- $\mathcal{FV}(l) \supseteq \mathcal{FV}(r)$.

A Higher-order Term Rewriting System (HTRS) R is a finite set of rewrite rules, such that, for each pair of rules $\Gamma_1 \vdash^{\preceq}_{\Sigma} l_1 \rightarrow r_1 : A_1, \Gamma_2 \vdash^{\preceq}_{\Sigma} l_2 \rightarrow r_2 : A_2 \in R$, $A_1 \not\prec^{A} A_2$.

The condition above translates to the requirement that it is not possible to use a rule to rewrite the type of another. This is therefore consistent with the original goal to define rewriting in such a way that it does not modify types, and hence preserve well-typedness of expressions.

Definition 20. Given a HTRS R and two terms $\Gamma \vdash^{\preceq}_{\Sigma} M : A$ and $\Gamma \vdash^{\preceq}_{\Sigma} N : A$ we define R-rewriting as follows:

$$\Gamma \vdash^{\preceq}_{\Sigma} M \xrightarrow{R} N : A \overset{\text{def}}{\Longleftrightarrow} \begin{array}{l} M_{\Downarrow} = E[\![\theta l]\!], N_{\Downarrow} = E[\![\theta r]\!] \\ \text{for some } (\Delta \vdash l \to r : B) \in R, \theta : \Delta \xrightarrow{\preceq} \Gamma(M_{\Downarrow}, \theta l), \\ \text{and } \Gamma \vdash^{\preceq}_{\Sigma} E[\![\Gamma(M_{\Downarrow}, \theta l) \vdash \circ : A(M_{\Downarrow}, \theta l)]\!] : A. \end{array}$$

We furthermore define R-conversion as the judgement $\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N : A$ formed according to the following rules:

$$\frac{\Gamma \vdash^{\preceq}_{\Sigma} M{:}A \quad M \equiv N \quad \Gamma \vdash^{\preceq}_{\Sigma} N{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N{:}A} \qquad \frac{\Gamma \vdash^{\preceq}_{\Sigma} M \xrightarrow{R} N{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N{:}A}$$

$$\frac{\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} N \xleftrightarrow{R} M{:}A} \qquad \frac{\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N'{:}A \quad \Gamma \vdash^{\preceq}_{\Sigma} N' \xleftrightarrow{R} N{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} M \xleftrightarrow{R} N{:}A}$$

In addition to R-conversion, a more natural notion of equality modulo R can be defined, as the smallest congruence relation containing all instances of R, and closed with respect to conversion:

Definition 21. Let R be a HTRS, congruence modulo R is defined by the judgement

$\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N : A \rhd \mathcal{D}$ $\qquad$ M and N of type A are congruent modulo R

where $\mathcal{D}$ is a set of type constants used to keep track of the dependency constraints. The rules associated to this judgement are the following:

$$\frac{\Gamma \vdash^{\preceq}_{\Sigma} M{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} M{:}A \rhd \emptyset} \qquad \frac{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N{:}A \rhd \mathcal{D}}{\Gamma \vdash^{\preceq}_{\Sigma} N \overset{R}{=} M{:}A \rhd \mathcal{D}} \qquad \frac{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N'{:}A \rhd \mathcal{D} \quad \Gamma \vdash^{\preceq}_{\Sigma} N' \overset{R}{=} N{:}A \rhd \mathcal{D}'}{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N{:}A \rhd (\mathcal{D} \cup \mathcal{D}')}$$

$$\frac{\Delta \vdash^{\preceq}_{\Sigma} l \to r{:}A \in R \quad \theta{:}\Delta \xrightarrow{\preceq} \Gamma}{\Gamma \vdash^{\preceq}_{\Sigma} \theta l \overset{R}{=} \theta r{:}\ \theta A \rhd \text{head}(A)} \qquad \frac{\Gamma \vdash^{\preceq}_{\Sigma} A{:}\text{type} \quad \Gamma, x{:}A \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N{:}B \rhd \mathcal{D}}{\Gamma \vdash^{\preceq}_{\Sigma} \lambda x{:}A.M \overset{R}{=} \lambda x{:}A.N{:}\Pi x{:}A.B \rhd \mathcal{D}}\ A \preceq^{M}_{\Sigma} B$$

$$\frac{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} M'{:}\Pi x{:}A.B \rhd \mathcal{D} \quad \Gamma \vdash^{\preceq}_{\Sigma} N \overset{R}{=} N'{:}A \rhd \mathcal{D}'}{\Gamma \vdash^{\preceq}_{\Sigma} M\ N \overset{R}{=} M'\ N'{:}[N/x]B \rhd (\mathcal{D} \cup \mathcal{D}')}\ a \not\prec^{A} B \text{ for all } a \in \mathcal{D}'$$

$$\frac{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N'{:}A \rhd \mathcal{D} \quad N' \equiv N \quad \Gamma \vdash^{\preceq}_{\Sigma} N'{:}A}{\Gamma \vdash^{\preceq}_{\Sigma} M \overset{R}{=} N{:}A \rhd \mathcal{D}}$$

The only place the set of dependencies $\mathcal{D}$ above plays a role is in the application rule: there, it restricts the rule to those cases where well-typedness of both sides is guaranteed. An analogous set can be defined for R-rewriting:

Definition 22. The set of dependency constraints generated by a R-rewriting step is defined as

$$(\Gamma \vdash_{\Sigma}^{\preceq} M \xrightarrow{R} N : A) \triangleright \{\mathrm{head}(B)\}$$

if $(\Delta \vdash_{\Sigma}^{\preceq} l \to r : B) \in R$ was the rewriting rule used in its definition.

This definition is extended to R-conversion:

$$\frac{\Gamma \vdash_{\Sigma}^{\preceq} M{:}A \quad M \equiv N \quad \Gamma \vdash_{\Sigma}^{\preceq} N{:}A}{(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N{:}A) \triangleright \emptyset} \qquad \frac{(\Gamma \vdash_{\Sigma}^{\preceq} M \xrightarrow{R} N{:}A) \triangleright \mathcal{D}}{(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N{:}A) \triangleright \mathcal{D}}$$

$$\frac{(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N{:}A) \triangleright \mathcal{D}}{(\Gamma \vdash_{\Sigma}^{\preceq} N \xleftrightarrow{R} M{:}A) \triangleright \mathcal{D}} \qquad \frac{(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N'{:}A) \triangleright \mathcal{D} \quad (\Gamma \vdash_{\Sigma}^{\preceq} N' \xleftrightarrow{R} N{:}A) \triangleright \mathcal{D}'}{(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N{:}A) \triangleright (\mathcal{D} \cup \mathcal{D}')}$$

This allows us to prove:

Theorem 23. *Let R be a HTRS, then for all M, N,*

$$(\Gamma \vdash_{\Sigma}^{\preceq} M \xleftrightarrow{R} N : A) \triangleright \mathcal{D} \Leftrightarrow \Gamma \vdash_{\Sigma}^{\preceq} M \stackrel{R}{=} N : A \triangleright \mathcal{D}.$$

4 Critical Pairs

As in the first order case, the check for local confluence of $\xrightarrow{R}$ goes through the search for critical pairs generated by the rules of the HTRS R. The definition of critical pairs here, however, is complicated by the presence of dependent types.

Definition 24. Let $\theta : \Gamma \xrightarrow{\preceq} \Delta$ be a substitution, an atomic term $\Gamma \vdash_{\Sigma} M \downarrow A$ is said to be stable for θ if $M = h\,\overline{N}$ where h is either a constant c or a variable $x \notin \mathrm{supp}(\theta)$.

Stability implies that the head of a canonical term is preserved by the application of a substitution, i.e. that $\theta(h\,\overline{N}) = h\,\overline{\theta N}$.

Definition 25. Let $\theta = \{\overline{x} \mapsto \overline{M}\} : \Gamma \xrightarrow{\preceq} \Delta$ be a substitution, an environment $\Gamma \vdash_{\Sigma}^{\preceq} E[\![\Gamma_{\circ} \vdash \circ : A_{\circ}]\!] : A$ is stable for θ if whenever the rule

$$\frac{\Gamma \vdash_{\Sigma}^{\preceq} M_1{:}\Pi x{:}A.B \quad \Gamma \vdash_{\Sigma}^{\preceq} E_2[\![\Gamma_{\circ} \vdash \circ{:}A_{\circ}]\!]{:}A}{\Gamma \vdash_{\Sigma}^{\preceq} M_1(E_2[\![\Gamma_{\circ} \vdash \circ{:}A_{\circ}]\!]){:}B} \quad A_{\circ} \not\preceq^{A} B$$

is applied, $\Gamma \vdash_{\Sigma}^{\preceq} M \downarrow \Pi x : A.B$ and M is stable for θ.

Since the application rule in a stable environment is applied to stable terms, one expects that when the substitution θ is applied, most of the original structure is preserved; the following lemma shows that this is actually the case:

Lemma 26. *Let*

$$\theta = \{\overline{x} \mapsto \overline{M}\} : \Gamma \xrightarrow{\preceq} \Delta$$
$$\Gamma \vdash^{\preceq}_{\Sigma} E[\![\Gamma_\circ \vdash \circ : A_\circ]\!] : A \text{ stable for } \theta,$$
$$\Gamma_\circ \vdash^{\preceq}_{\Sigma} M_\circ \Downarrow A_\circ, \qquad \Gamma_\circ \vdash^{\preceq}_{\Sigma} A_\circ \downarrow \mathrm{type},$$

and $M = E[\![M_\circ]\!]$*, then there are*

$$\theta' : \Gamma_\circ \xrightarrow{\preceq} \Delta_\circ \text{ with } \theta' \doteq \theta,$$
$$\Delta \vdash^{\preceq}_{\Sigma} E'[\![\Delta_\circ \vdash \circ : [\overline{M}/\overline{x}]A_\circ]\!] : [\overline{M}/\overline{x}]A$$

such that:

1. *if* $\Gamma \vdash^{\preceq}_{\Sigma} E[\![M_\circ]\!] \Downarrow A$ *then* $\theta M = E'[\![\theta' M_\circ]\!]$ *and* $\Delta \vdash^{\preceq}_{\Sigma} E'[\![\theta' M_\circ]\!] \Downarrow [\overline{M}/\overline{x}]A$*;*
2. *if* $\Gamma \vdash^{\preceq}_{\Sigma} E[\![M_\circ]\!] \downarrow A$ *then* $\theta M \to^{*}_{\eta} E'[\![\theta' M_\circ]\!]$ *and* $\Delta \vdash^{\preceq}_{\Sigma} E'[\![\theta' M_\circ]\!] \downarrow [\overline{M}/\overline{x}]A$*.*

We will write $E(\theta, M_\circ)$ and $\theta(E, M_\circ)$ to denote the environments E' and substitutions θ' obtained from Lemma 26.(1).

Definition 27 (Critical Pair). Let R be a HTRS, $\Gamma_1 \vdash^{\preceq}_{\Sigma} l_1 \to r_1 : C_1$, $\Gamma_2 \vdash^{\preceq}_{\Sigma} l_2 \to r_2 : C_2$ two rules in R, $\theta_1 : \Gamma_1 \xrightarrow{\preceq} \Delta$, $\theta_2 : \Gamma_2 \xrightarrow{\preceq} \Delta$, $\theta_1 = \{\overline{x} \mapsto \overline{N}\}$, and $\Gamma_1 \vdash^{\preceq}_{\Sigma} E[\![\Gamma_\circ \vdash \circ : A_\circ]\!] : A$ such that $l_1 = E[\![M_\circ]\!]$, $M_\circ$ not η-equivalent to a variable in $\mathrm{dom}\Gamma$, $\theta_1(E_1, M_\circ)M_\circ = \theta_2 l_2$, then

$$\Delta \vdash^{\preceq}_{\Sigma} < E(\theta_1, M_\circ)[\![\theta_2 r_2]\!], \theta_1 r_1 >: [\overline{N}/\overline{x}]C_1$$

is a *critical pair*

Remark. By applying a renaming substitution and using α-conversion, we can assume, without loss of generality, $\Gamma_1(M_\circ) \cap \Gamma_2 = \emptyset$. Then by Weakening it is easily verified that $\theta_1(E_1, M_\circ) \cup \theta_2 : \Gamma_1(M_\circ), \Gamma_2 \xrightarrow{\preceq} \Delta$ can be chosen to be the most general unifier of $M_\circ$ and l_2, and the definition above appears as a generalization of the familiar one for first-order TRSs.

Using the definition above, the following can be proved:

Theorem 28 (Critical Pair Lemma). *Let R be a HTRS, if* $\Gamma \vdash^{\preceq}_{\Sigma} M \xrightarrow{R} N_1 : A$ *and* $\Gamma \vdash^{\preceq}_{\Sigma} M \xrightarrow{R} N_2 : A$ *then either there is a critical pair in R, or there are rewriting sequences* $\Gamma \vdash^{\preceq}_{\Sigma} N_1^{(i)} \xrightarrow{R} N_1^{(i+1)}$ $(0 \leq i < n_1)$*,* $\Gamma \vdash^{\preceq}_{\Sigma} N_2^{(i)} \xrightarrow{R} N_2^{(i+1)}$ $(0 \leq i < n_2)$ *such that* $N_1^{(0)} = N_1$*,* $N_2^{(0)} = N_2$*,* $N_1^{(n_1)} \equiv N_2^{(n_2)}$*.*

5 Applications

In this section we briefly mention a few applications of higher order rewriting with dependent types.

Lambda calculus with simple types The simply-typed lambda calculus can be easily formalized within LF. We will need two type families: one, called **type**, for types, and the second, **term**, indexed by objects of the first, for terms.

$$\begin{aligned}
\mathbf{type} &: \mathrm{type} \\
\mathbf{arrow} &: \mathbf{type} \Rightarrow (\mathbf{type} \Rightarrow \mathrm{type}) \\
\\
\mathbf{term} &: \mathbf{type} \Rightarrow \mathrm{type} \\
\mathbf{lambda} &: \Pi x : \mathbf{type}.\Pi y : \mathbf{type}.(((\mathbf{term}\ x) \Rightarrow (\mathbf{term}\ y)) \Rightarrow \mathbf{term}(\mathbf{arrow}\ x\ y)) \\
\mathbf{app} &: \Pi x : \mathbf{type}.\Pi y : \mathbf{type}.((\mathbf{term}(\mathbf{arrow}\ x\ y)) \Rightarrow ((\mathbf{term}\ x) \Rightarrow (\mathbf{term}\ y)))
\end{aligned}$$

β- and η-reductions can then be expressed by the two rewrite rules

$$\begin{aligned}
\Gamma_1 \vdash^{\preceq}_{\Sigma} (\mathbf{app}\ A\ B\ (\mathbf{lambda}\ A\ B\ (\lambda x : (\mathbf{term}\ A).F\ x))\ U) &\rightarrow (F\ U) : \mathbf{term}\ B \\
\Gamma_2 \vdash^{\preceq}_{\Sigma} \mathbf{lambda}\ A\ B\ (\lambda x : (\mathbf{term}\ A).\mathbf{app}\ A\ B\ G\ x) &\rightarrow G : \mathbf{term}\ (\mathbf{arrow}\ A\ B)
\end{aligned}$$

where

$$\begin{aligned}
\Gamma_1 &= A : \mathbf{type}, B : \mathbf{type}, F : (\mathbf{term}\ A) \Rightarrow (\mathbf{term}\ B), U : \mathbf{term}\ A \\
\Gamma_2 &= A : \mathbf{type}, B : \mathbf{type}, G : \mathbf{term}\ (\mathbf{arrow}\ A\ B)
\end{aligned}$$

Using the critical pair criterion one can easily check that the system is locally confluent. Since it is also known to be terminating, we conclude by Newman's Lemma [11] that it is confluent.

Category Theory To represent categories we define types for objects and morphisms:

$$\begin{aligned}
\mathbf{categ} &: \mathrm{type} \\
\mathbf{object} &: \mathbf{categ} \Rightarrow \mathrm{type} \\
\mathbf{morph} &: \Pi x : \mathbf{categ}.((\mathbf{object}\ x) \Rightarrow (\mathbf{object}\ x)) \\
\\
\mathbf{o} &: \Pi x\ :\ \mathbf{categ}\ .\ \Pi u : (\mathbf{object}\ x)\ .\ \Pi v : (\mathbf{object}\ x)\ .\ \Pi w : (\mathbf{object}\ x)\ . \\
&\qquad\qquad ((\mathbf{morph}\ x\ v\ w) \Rightarrow ((\mathbf{morph}\ x\ u\ v) \Rightarrow (\mathbf{morph}\ x\ u\ w))) \\
\mathbf{id} &: \Pi x : \mathbf{categ}\ .\ \Pi u : (\mathbf{object}\ x)\ .\ \mathbf{morph}\ x\ u\ u
\end{aligned}$$

The basic axioms can be then formalized by the rewrite system

$$\begin{aligned}
\Gamma_1 \vdash^{\preceq}_{\Sigma} (\mathbf{o}\ x\ u\ u\ v\ f\ (\mathbf{id}\ x\ u)) &\rightarrow f : \mathbf{morph}\ x\ u\ v \\
\Gamma_1 \vdash^{\preceq}_{\Sigma} (\mathbf{o}\ x\ u\ v\ v\ (\mathbf{id}\ x\ v)\ f) &\rightarrow f : \mathbf{morph}\ x\ u\ v \\
\Gamma_2 \vdash^{\preceq}_{\Sigma} (\mathbf{o}\ x\ u\ w\ z\ h\ (\mathbf{o}\ x\ u\ v\ w\ g\ f)) &\rightarrow (\mathbf{o}\ x\ u\ v\ z\ (\mathbf{o}\ x\ u\ w\ z\ h\ g)\ f) : \mathbf{morph}\ x\ u\ z
\end{aligned}$$

where

$$\begin{aligned}
\Gamma_1 &= x : \mathbf{categ}, u : (\mathbf{object}\ x), v : (\mathbf{object}\ x), f : (\mathbf{morph}\ x\ u\ v) \\
\Gamma_2 &= x : \mathbf{categ}, u : (\mathbf{object}\ x), v : (\mathbf{object}\ x), w : (\mathbf{object}\ x), z : (\mathbf{object}\ x), \\
&\quad f : (\mathbf{morph}\ x\ u\ v), g : (\mathbf{morph}\ x\ v\ w), h : (\mathbf{morph}\ x\ w\ z).
\end{aligned}$$

More complex notions of category theory can be similarly represented: terminating and confluent HTRSs for natural transformations, adjunctions, and monads are presented in [2].

Cut Elimination Theorem In [16], F. Pfenning gives a LF representation of the cut elimination theorem for both intuitionistic and classical sequent calculi. It turns out that is possible to transform all the cases in this proof into rewrite rules, yielding a (terminating) HTRS. While this is not confluent, each of the (nontrivial) critical pairs appears to correspond to a different case of commutativity of couples of rules of the sequent calculus.

6 Future Developments

While the Critical Pair Lemma gives us a criterion to check for local confluence of a HTRS, we are are also interested in general methods to test for termination. Two methods have been recently proposed [6, 17], and it is our hope that they can be adapted to dependent types. Also, the same notion of dependency relation has been used to prove well foundedness of proofs by structural induction in LF [19]; this suggests that it might be useful in proving termination of HTRSs as well.

Another interesting direction of research would be to define, on the same line of [9], rewriting modulo an equational theory. This might allow to circumvent the restrictions imposed by the dependencies: a single HTRS could be decomposed into several ones, each involving disjoint sets of base types. The proof of confluence could then proceed by stages: as soon as one component HTRS has been proved terminating and confluent, it becomes part of the underlying equational theory on which the confluence of the next one is tested.

7 Acknowledgments

I would like to thank F. Pfenning for his help, suggestions, and support. I also thank E. Rohwedder and W. Gehrke for their valuable comments.

References

1. Coquand, T. *An algorithm for Testing Conversion in Type Theory.* Logical Frameworks, Cambridge University Press, 1991, pages 155-279
2. Gehrke, W., *Decidability Results For Categorical Notions Related to Monads by Rewriting Techniques* Ph.D. Thesis, Johannes Kepler Universität, Linz, 1995
3. Geuvers, H. *The Church-Rosser Property for $\beta\eta$-Reduction in Typed λ-Calculi.* Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS), 1992, pages 453-460
4. Harper, R., Honsell F., Plotkin, G. *A framework for Defining Logics.* Journal of the Association for Computing Machinery, January 1993, pages 143-184
5. Huet, G.P., Lang, B. *Proving and Applying Program Transformations Expressed with Second-Order Patterns* Acta Informatica 11, 1978, pages 31-55
6. Kahrs, D. *Towards a Domain Theory for Termination Proofs.* Proceedings of the 6th International Conference on Rewriting Techniques and Applications (RTA), 1995, pages 241-255

7. Knuth, D.E., Bendix, P.B. *Simple Word Problems in Universal Algebra.* Computational Problems in Abstract Algebra, Pergamon Press, 1972, pages 263-297
8. Loría-Sáenz, C. A. *A Theoretical Framework for Reasoning about Program Construction Based on Extensions of Rewrite Systems* Ph.D. Thesis, Universität Kaiserslautern, 1993
9. Mayr, R., Nipkow, T. *Higher-Order Rewrite Systems and their Confluence.* Technical Report TUM-I9433, Technische Universität München, 1994
10. Miller, D. *A Logic Programming Language With Lambda abstraction, Function Variables, and Simple Unification.* LFCS Report Series, University of Edinburgh, 1991, pages 253-281
11. Newman, M.H.A. *On theories with a combinatorial definition of 'equivalence'* Annals of Mathematics, 43(2), 1942, pages 223-243
12. Nipkow, T. *Higher-Order Critical Pairs.* Proceedings of the 5th IEEE Conference of Logic In Computer Science (LICS), 1990, pages 342-348
13. Paulson, L.C. *The Foundation of a Generic Theorem Prover* Journal of Automated Reasoning, vol. 5, 1989, pages 363-397
14. Pfenning, F. *Logic Programming in the LF Logical Framework.* G. Huet, G. Plotkin ed., Logical Frameworks, Cambridge University Press, 1991, pages 149-181
15. Pfenning, F. *Unification and anti-unification in the Calculus of Constructions.*, Proceedings of the 6th IEEE Conference of Logic In Computer Science (LICS), 1991, pages 149-181
16. Pfenning, F. *A Structural Proof of Cut Elimination and Its Representation in a Logic Framework* Technical Report CMU-CS-94-218, Carnegie Mellon University, 1994
17. Pol, J. *Termination Proofs for Higher-Order Rewrite Systems*, J. Heering, K. Meinke, B. Moller, T. Nipkow ed., Higher Order Algebra, Logic and Term Rewriting (HOA),Lecture Notes in Computer Science, vol 816, 1994, pages 305-325
18. Prehofer, C. *Solving Higher-Order Equations.*, Technical Report, Technische Universität München, 1994
19. Rohwedder, E. , Pfenning, F. *Mode and Termination Analysis for Higher-Order Logic.*, to appear at the 1996 European Symposium on Programming (ESOP)
20. Salvesen, A. *The Church-Rosser Property for Pure Systems with $\beta\eta$-Reduction.* Technical Report, University of Oslo, 1992
21. Virga, R. *Higher-Order Superposition for Dependent Types*, Technical Report CMU-CS-95-150, Carnegie Mellon University, 1995 (*http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rvirga/Web/dep-rel.ps*)
22. Wolfram, D.A *Rewriting, and Equational Unification: the Higher-Order Cases* Proceedings of the 4th International Conference on Rewriting Techniques and Applications (RTA), 1991, pages 25-36

This article was processed using the LaTeX macro package with LLNCS style