Properties of Terms in Continuation-Passing Style in an Ordered Logical Framework

Jeff Polakow[*] and Frank Pfenning[†]
Department of Computer Science
Carnegie Mellon University

`jpolakow@cs.cmu.edu` and `fp@cs.cmu.edu`

April 2000

1 Introduction

A logical framework is a meta-language for the formalization of deductive systems as used in the description of logics and programming languages. It should directly support common notions and techniques of this domain, thereby achieving two related goals: on one hand, it provides a conceptual tool for the concise definition and rigorous reasoning about programming languages and logics; on the other hand, it significantly reduces the effort required to actually implement deductive systems.

There are various styles of logical frameworks (see, for example, [Pfe96]) with different strengths and weaknesses. Our starting point is the logical framework LF [HHP93] which supports variable binding and capture-avoiding substitution through higher-order abstract syntax, and parametric and hypothetical judgments through dependently typed functions. In previous work, I. Cervesato and the second author have investigated a conservative extension of LF which allows the natural representation of systems with state via linear functions and additive pairs [CP98, Cer96]. In this paper we explore a further conservative extension with an intrinsic notion of order. The impetus for the development of such a framework came from the observation of certain analogies between ordering properties of CPS terms [DP95, DDP99] and substitution properties of linear λ-calculi. In turn, the coding in a logical framework immediately suggested an improvement in the abstract machines for the execution of CPS terms, namely the merging of a control stack and a data stack.

In this paper we sketch the relevant fragment of an ordered logical framework and then show how it supports the representation of CPS terms and an improved machine for their evaluation. The ordering properties of CPS terms which at first appear somewhat ad hoc are directly captured as *typing* properties in the representation that are preserved during evaluation. Complicated stack invariants can be recognized as uniform substitution properties, providing an example of how the organizing principles of a logical framework can contribute conceptually to our understanding of object languages.

In this paper, we do not study the meta-theoretic properties of an ordered logical framework. However, the fragment required for our application here is particularly simple: it is a two-level type theory in the style of [PP99a, PP99b], where quantifiers range over terms from an ordered λ-calculus

[*]Partially supported by the National Science Foundation under grant CCR-9804014.
[†]Partially supported by the National Science Foundation under grant CCR-9619584.

without further dependencies. The existence of canonical forms in the ordered λ-calculus [PP99a] strongly suggests that canonical forms also exist for this first-order type theory, following standard lines of development. Unifying the layers as is customary in type theory leads to a much more economical, yet meta-theoretically much less tractable framework. Another simplification arises because we need only one among the two possible ordered implications familiar from the Lambek calculus [Lam58], and neither linear implication nor additive constructs are needed.

The rest of this paper is organized as follows. In section 2 we review the the basic rules and properties of a fragment of ordered linear logic. In section 3 we review CPS terms and their occurrence invariants. In section 4 we give a representation of CPS terms in an ordered logical framework in which the occurrence invariants are implicit in the ordered types. In section 5 we introduce a simple two-level logical framework which we use, in sections 6, 7 and 8, to give representations of the CPS transformation and different evaluation models for CPS terms. Finally we give some conclusions and directions for further work in section 9.

2 Ordered Linear Logic

Ordered linear logic (OLL) is a conservative extension of intuitionistic linear logic with ordered hypotheses. We begin with a review of the fragment of OLL which we will use. For a description of the full system see [PP99b, PP99a].

$$\begin{array}{llcll}
\textit{Types} & A & ::= & a & \text{atomic types} \\
 & & \mid & A_1 \rightarrow A_2 & \text{intuitionistic implication} \\
 & & \mid & A_1 \twoheadrightarrow A_2 & \text{ordered right implication}
\end{array}$$

$$\begin{array}{llcll}
\textit{Objects} & M & ::= & c & \text{constants} \\
 & & \mid & x \mid z & \text{variables} \\
 & & \mid & \lambda x{:}A.\ M \mid M_1\, M_2 & \text{intuitionistic functions } (A \rightarrow B) \\
 & & \mid & \overset{>}{\lambda} z{:}A.\ M \mid M_1 {}^{>} M_2 & \text{right ordered functions } (A \twoheadrightarrow B)
\end{array}$$

We build typing rules for OLL objects from the following judgment

$$\Gamma; \Omega \vdash M : A$$

where Γ is a list of unrestricted hypotheses, Ω is a list of ordered hypotheses, and a signature containing constant declarations is left implicit. The inference rules will be structured to allow copying, discarding, and exchanging of unrestricted hypotheses. However, ordered hypotheses will not enjoy those structural properties.

Here are the rules for unrestricted functions.

$$\frac{}{(\Gamma_1, x{:}A, \Gamma_2); \cdot \vdash x : A}\,\mathbf{ivar} \qquad \frac{(\Gamma, x{:}A); \Omega \vdash M : B}{\Gamma; \Omega \vdash \lambda x{:}A.\ M : A \rightarrow B} \rightarrow I$$

$$\frac{\Gamma; \Omega \vdash M : A \rightarrow B \qquad \Gamma; \cdot \vdash N : A}{\Gamma; \Omega \vdash M\, N : B} \rightarrow E$$

Note that the ordered context in the minor premise of $\rightarrow_E$ must be empty. This ensures that unrestricted functions, which may use their argument arbitrarily, may not be applied to ordered arguments, which must be used exactly once.

The rules for ordered functions follow.

$$\frac{}{\Gamma; z{:}A \vdash z : A}\,\textbf{ovar} \qquad \frac{\Gamma; (\Omega, z{:}A) \vdash M : B}{\Gamma; \Omega \vdash \overset{>}{\lambda} z{:}A.\ M : A \twoheadrightarrow B}\twoheadrightarrow I$$

$$\frac{\Gamma; \Omega_1 \vdash M : A \twoheadrightarrow B \qquad \Gamma; \Omega_2 \vdash N : A}{\Gamma; (\Omega_1, \Omega_2) \vdash M \overset{>}{\ } N : B}\twoheadrightarrow E$$

Note that the argument to an ordered function may only depend upon ordered hypotheses to the right of those use by the body of the function—the order of the hypotheses constrains their use by ordered functions.

The reduction rules for OLL objects are simply β-reduction for both kinds of functions. The appropriate notion of equality of objects also includes η-conversion so that every well-typed object has an equivalent canonical form.

Our calculus enjoys subject reduction, as proved in [PP99a].

Theorem 1 (Subject Reduction)
If $M \Longrightarrow M'$ *and* $\Gamma; \Omega \vdash M : A$ *then* $\Gamma; \Omega \vdash M' : A$.

Proof: For each reduction, we apply inversion to the given typing derivation and then use a substitution lemma to obtain the typing derivation for the conclusion. □

Finally, we note that this calculus has canonical forms as shown in [PP99a]. Thus all terms of functional type may be converted to the form $\lambda x{:}A.\ M$ or $\overset{>}{\lambda} z{:}A.\ M$; and all objects of atomic type may be reduced to a constant applied to zero or more canonical objects. We will use the judgment

$$\Gamma; \Omega \vdash M \Uparrow A$$

to denote that object M is canonical at well-formed type A.

The existence of canonical forms for this simple implicational fragment of OLL provides a basis for an ordered logical framework. We conjecture that an ordered logical framework based on a full type theory can be constructed along the lines of the linear logical framework [CP98]. In this paper we only need a two-level fragment as explained in Section 5.

3 CPS terms

In this section we review CPS terms and their ordering properties as investigated in [DP95, DDP99].

We use the following syntax for direct style (DS) terms:

DS Terms	r	$::=$	e
DS Expressions	e	$::=$	$e_1\, e_2 \mid t$
DS Trivial Expressions	t	$::=$	$\underline{\lambda} x.\ r \mid x$

and for CPS terms:

Root Terms	r	$::=$	$\underline{\lambda} k.\ e$
Serious Terms	e	$::=$	$t_1\, t_2\, c \mid c\, t$
Trivial Terms	t	$::=$	$\underline{\lambda} x.\ r \mid x \mid v$
Continuation Terms	c	$::=$	$\underline{\lambda} v.\ e \mid k$

Note that in the CPS syntax, we are distinguishing variables x which are parameters of functions from variables v which are parameters to continuations.

We formulate the left-to-right call-by-value CPS transform as three mutually recursive judgements. A direct-style term r is transformed into a CPS term r' whenever the judgment

$$\vdash r \xrightarrow{DR} r'$$

is satisfied. Given a CPS continuation c, a direct-style expression e is transformed into a CPS expression e' whenever the judgment

$$\vdash e \,;\, c \xrightarrow{DE} e'$$

is satisfied. Finally, a direct-style trivial expression t is transformed into a CPS trivial expression t' whenever the judgment

$$\vdash t \xrightarrow{DT} t'$$

is satisfied.

$$\frac{\vdash e \,;\, k \xrightarrow{DE} e'}{\vdash e \xrightarrow{DR} \underline{\lambda} k.\, e'} \qquad \frac{\vdash t \xrightarrow{DT} t'}{\vdash t \,;\, c \xrightarrow{DE} c\, t'}$$

$$\frac{\vdash e_2 \,;\, \underline{\lambda} v_2.\, v_1\, v_2\, c \xrightarrow{DE} e_2' \qquad \vdash e_1 \,;\, \underline{\lambda} v_1.\, e_2' \xrightarrow{DE} e'}{\vdash e_1\, e_2 \,;\, c \xrightarrow{DE} e'} \quad (v_1 \text{ not free in conclusion})$$

$$\frac{}{\vdash x \xrightarrow{DT} x} \qquad \frac{\vdash r \xrightarrow{DR} r'}{\vdash \underline{\lambda} x.\, r \xrightarrow{DT} \underline{\lambda} x.\, r'}$$

Terms resulting from a left-to-right call-by-value CPS translation of direct-style terms satisfy additional invariants, both on occurrences of continuation identifiers k and the parameters of continuations v. In [DP95] the occurrence properties on continuation identifiers and parameters are separately specified with two judgment families. However these occurrences are tightly coupled and may be naturally captured with just one family of judgments as follows. We shall use four mutually recursive judgments

$$\models^{\mathbf{Root}} r \qquad \Phi \models^{\mathbf{Exp}} e \qquad \Phi \models^{\mathbf{Triv}} t; \Phi' \qquad \Phi \models^{\mathbf{Cont}} c$$

where Φ is a stack of both continuation identifiers and parameters:

$$\Phi ::= \cdot \mid \Phi, k \mid \Phi, v$$

When Φ' is a prefix of Φ, we define $\Phi - \Phi'$ as the remainder of Φ.

$$\frac{k \models^{\mathbf{Exp}} e}{\models^{\mathbf{Root}} \underline{\lambda} k.\, e}$$

$$\frac{\Phi \models^{\mathbf{Triv}} t; \Phi' \qquad \Phi' \models^{\mathbf{Cont}} c}{\Phi \models^{\mathbf{Exp}} c\, t} \qquad \frac{\Phi \models^{\mathbf{Triv}} t_1; \Phi' \qquad \Phi' \models^{\mathbf{Triv}} t_0; \Phi'' \qquad \Phi'' \models^{\mathbf{Cont}} c}{\Phi \models^{\mathbf{Exp}} t_0\, t_1\, c}$$

$$\frac{}{\Phi \models^{\textbf{Triv}} x; \Phi} \qquad \frac{\models^{\textbf{Root}} r}{\Phi \models^{\textbf{Triv}} \underline{\lambda} x.\ r; \Phi} \qquad \frac{}{\Phi, v \models^{\textbf{Triv}} v; \Phi}$$

$$\frac{}{k \models^{\textbf{Cont}} k} \qquad \frac{\Phi, v \models^{\textbf{Exp}} e}{\Phi \models^{\textbf{Cont}} \underline{\lambda} v.\ e}$$

Our presentation is general enough to serve as a target for both Plotkin's original call-by-value CPS transform and a one-pass version which avoids administrative redices.

It is easy to see that continuation identifiers, k, are used linearly in each root term and that continuation parameters, v, form a stack in each serious term. Furthermore, each k is used in a serious term only after all local parameters to continuations are used. These properties can be precisely captured with ordered types.

We would like to prove that $\vdash r \xrightarrow{DR} r'$ implies $\models^{\textbf{Root}} r'$. Proving this directly with the above definitions requires a logical relations style argument [DDP99, DP95]. However, by using an ordered logical framework, this may be proved directly.

4 Ordered Logical Framework Representation

We will now show how the OLL type theory presented in section 2 provides exactly what is needed to capture and reason about the ordering properties of CPS terms.

4.1 DS Terms

Our representation of DS terms will use three basic types corresponding to the three kinds of DS terms.

$$\mathsf{droot} : \mathsf{type}. \qquad \mathsf{dexp} : \mathsf{type}. \qquad \mathsf{dtriv} : \mathsf{type}.$$

We will then build our representations from term constructors corresponding to DS terms. Note that representation uses higher-order abstract syntax, so object level functions are represented by meta-level functions.

$$\begin{array}{rcl}
\mathsf{e2r} & : & \mathsf{dexp} \to \mathsf{droot}. \\
\mathsf{dapp} & : & \mathsf{dexp} \to \mathsf{dexp} \to \mathsf{dexp}. \\
\mathsf{t2e} & : & \mathsf{dtriv} \to \mathsf{dexp}. \\
\mathsf{dlam} & : & (\mathsf{triv} \to \mathsf{droot}) \to \mathsf{dtriv}.
\end{array}$$

Given the previous signature, there is an obvious compositional bijection between DS terms and canonical objects in the above signature. This bijection is established by the following mutually recursive representation functions, $\ulcorner - \urcorner^R$,$\ulcorner - \urcorner^E$,$\ulcorner - \urcorner^T$, and their inverses $\llcorner - \lrcorner^R$,$\llcorner - \lrcorner^E$,$\llcorner - \lrcorner^T$.

$$\begin{array}{rcl}
\ulcorner e \urcorner^R & = & \mathsf{e2r}\, \ulcorner e \urcorner^E
\end{array} \qquad
\begin{array}{rcl}
\llcorner \mathsf{e2r}\, E \lrcorner_R & = & \llcorner E \lrcorner_E
\end{array}$$

$$\begin{array}{rcl}
\ulcorner e_0\, e_1 \urcorner^E & = & \mathsf{dapp}\, \ulcorner e_0 \urcorner^E \ulcorner e_1 \urcorner^E \\
\ulcorner t \urcorner^E & = & \mathsf{d2e}\, \ulcorner t \urcorner^T
\end{array} \qquad
\begin{array}{rcl}
\llcorner \mathsf{dapp}\, E_0\, E_1 \lrcorner_E & = & \llcorner E_0 \lrcorner_E \llcorner E_1 \lrcorner_E \\
\llcorner \mathsf{d2e}\, T \lrcorner_E & = & \llcorner T \lrcorner_T
\end{array}$$

$$\begin{array}{rcl}
\ulcorner \underline{\lambda} x.\ r \urcorner^T & = & \mathsf{dlam}\, (\lambda x{:}\mathsf{dtriv}.\ \ulcorner r \urcorner^R) \\
\ulcorner x \urcorner^T & = & x
\end{array} \qquad
\begin{array}{rcl}
\llcorner \mathsf{dlam}\, (\lambda x{:}\mathsf{dtriv}.\ R) \lrcorner_T & = & \underline{\lambda} x.\ \llcorner R \lrcorner_R \\
\llcorner x \lrcorner_T & = & x
\end{array}$$

4.2 CPS Terms

Our representation of CPS terms will use four basic types corresponding to the four kinds of CPS terms.

$$\mathsf{root} : \mathsf{type}. \qquad \mathsf{exp} : \mathsf{type}. \qquad \mathsf{triv} : \mathsf{type}. \qquad \mathsf{cont} : \mathsf{type}.$$

We will then build our representations from term constructors corresponding to CPS terms. The use of ordered types forces the CPS term representations to have the linearity and ordering constraints noted at the end of section 3.

$$\begin{array}{rcl}
\mathsf{klam} & : & (\mathsf{cont} \twoheadrightarrow \mathsf{exp}) \to \mathsf{root}. \\
\mathsf{kapp} & : & \mathsf{cont} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{exp}. \\
\mathsf{app} & : & \mathsf{cont} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{exp}. \\
\mathsf{lam} & : & (\mathsf{triv} \to \mathsf{root}) \to \mathsf{triv}. \\
\mathsf{vlam} & : & (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \twoheadrightarrow \mathsf{cont}.
\end{array}$$

The intuition behind these type declarations may best be gleaned from the representation function and its adequacy theorem below. Note that a positive occurrence of an unrestricted function $\to$ as in the type of klam imposes a restriction on the corresponding argument: it may not depend on the continuation k or parameters v. On the other hand, a negative occurrence of $\to$ as in the type of lam licenses the unrestricted use of the corresponding bound variable x. The right ordered functions $\twoheadrightarrow$ impose the stack-like discipline on parameters of continuations and the continuations themselves explained in the previous section.

Given the previous signature, there is a compositional bijection between CPS terms satisfying the occurrence conditions and canonical objects in the above signature. This bijection is established by the following representation function, $\ulcorner - \urcorner$ and its inverse $\llcorner - \lrcorner$.

$$\begin{array}{rcl@{\qquad}rcl}
\ulcorner \underline{\lambda} k.\, e \urcorner & = & \mathsf{klam}\,(\overset{>}{\lambda} k{:}\mathsf{cont}.\, \ulcorner e \urcorner) & \llcorner \mathsf{klam}\,(\overset{>}{\lambda} k{:}\mathsf{cont}.\, E) \lrcorner & = & \underline{\lambda} k.\, \llcorner E \lrcorner \\[1em]
\ulcorner t_0\, t_1\, c \urcorner & = & \mathsf{app}^{>}\ulcorner c \urcorner^{>}\ulcorner t_0 \urcorner^{>}\ulcorner t_1 \urcorner & \llcorner \mathsf{app}^{>} C^{>} T_0{}^{>} T_1 \lrcorner & = & \llcorner T_0 \lrcorner \llcorner T_1 \lrcorner \llcorner C \lrcorner \\
\ulcorner c\, t \urcorner & = & \mathsf{kapp}^{>}\ulcorner c \urcorner^{>}\ulcorner t \urcorner & \llcorner \mathsf{kapp}^{>} C^{>} T \lrcorner & = & \llcorner C \lrcorner \llcorner T \lrcorner \\[1em]
\ulcorner \underline{\lambda} x.\, r \urcorner & = & \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\, \ulcorner r \urcorner) & \llcorner \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\, R) \lrcorner & = & \underline{\lambda} x.\, \llcorner R \lrcorner \\
\ulcorner x \urcorner & = & x & \llcorner x \lrcorner & = & x \\
\ulcorner v \urcorner & = & v & \llcorner v \lrcorner & = & v \\[1em]
\ulcorner \underline{\lambda} v.\, e \urcorner & = & \mathsf{vlam}^{>}(\overset{>}{\lambda} v{:}\mathsf{triv}.\, \ulcorner e \urcorner) & \llcorner \mathsf{vlam}^{>}(\overset{>}{\lambda} v{:}\mathsf{triv}.\, E) \lrcorner & = & \underline{\lambda} v.\, \llcorner E \lrcorner \\
\ulcorner k \urcorner & = & k & \llcorner k \lrcorner & = & k
\end{array}$$

$$\begin{array}{rcl@{\qquad}rcl}
\ulcorner . \urcorner & = & \cdot & \llcorner \cdot \lrcorner & = & \cdot \\
\ulcorner \Phi, v \urcorner & = & \ulcorner \Phi \urcorner, v{:}\mathsf{triv} & \llcorner \Phi, v{:}\mathsf{triv} \lrcorner & = & \llcorner \Phi \lrcorner, v \\
\ulcorner \Phi, k \urcorner & = & \ulcorner \Phi \urcorner, k{:}\mathsf{cont} & \llcorner \Phi, k{:}\mathsf{cont} \lrcorner & = & \llcorner \Phi \lrcorner, k
\end{array}$$

Note that and $\llcorner \ulcorner u \urcorner \lrcorner = u$ for any term u. Additionally, since variables are mapped to variables, the representation function and its inverse are compositional (i.e., commute with substitution).

We formally prove the correspondence in two parts.

Theorem 2 (Representations are Canonical Forms)
Consider terms r, e, and t with free ordinary variables among $x_1, \ldots, x_n$ and let $\Gamma = x_1{:}\mathsf{triv} \ldots x_n{:}\mathsf{triv}$.

1. *If $\models^{\mathbf{Root}} r$ then $\Gamma; \cdot \vdash \ulcorner r \urcorner \Uparrow \mathsf{root}$.*

2. *If* $\Phi \models^{\mathbf{Exp}} e$ *then* $\Gamma; \ulcorner\Phi\urcorner \vdash \ulcorner e\urcorner \Uparrow \mathsf{exp}$.

3. *If* $\Phi \models^{\mathbf{Triv}} t; \Phi'$ *then* $\Gamma; \ulcorner(\Phi - \Phi')\urcorner \vdash \ulcorner t\urcorner \Uparrow \mathsf{triv}$.

4. *If* $\Phi \models^{\mathbf{Cont}} c$ *then* $\Gamma; \ulcorner\Phi\urcorner \vdash \ulcorner c\urcorner \Uparrow \mathsf{cont}$.

Proof: By induction on the structure of the given derivations. □

Theorem 3 (Canonical Forms are Representations)
Let $\Gamma = x_1{:}\mathsf{triv}, \ldots, x_n{:}\mathsf{triv}$ *be given.*

1. *For any* M *such that* $\Gamma; \cdot \vdash M \Uparrow \mathsf{root}$,
   $\llcorner M \lrcorner$ *is defined and* $\models^{\mathbf{Root}} \llcorner M \lrcorner$.

2. *For any* $\Omega = v_1{:}\mathsf{triv} \ldots v_n{:}\mathsf{triv}$ *and* M *with* $\Gamma; k{:}\mathsf{cont}, \Omega \vdash M \Uparrow \mathsf{exp}$, $\llcorner M \lrcorner$ *is defined and* $k, \llcorner\Omega\lrcorner \models^{\mathbf{Exp}} \llcorner M \lrcorner$.

3. *For any* $\Omega = v_1{:}\mathsf{triv} \ldots v_n{:}\mathsf{triv}$ *and* M *such that* $\Gamma; \Omega \vdash M \Uparrow \mathsf{triv}$,
   $\llcorner M \lrcorner$ *is defined and* $\Phi, \llcorner\Omega\lrcorner \models^{\mathbf{Triv}} \llcorner M \lrcorner; \Phi$ *for any* Φ.

4. *For any* $\Omega = v_1{:}\mathsf{triv} \ldots v_n{:}\mathsf{triv}$ *and* M *with* $\Gamma; k{:}\mathsf{cont}, \Omega \vdash M \Uparrow \mathsf{cont}$, $\llcorner M \lrcorner$ *is defined and* $k, \llcorner\Omega\lrcorner \models^{\mathbf{Cont}} \llcorner M \lrcorner$.

Proof: By induction on the structure of the given canonical derivations. For the cases when $M = \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\ r)$, and $M = \mathsf{klam}\,(\overset{>}{\lambda} k{:}\mathsf{cont}.\ e)$ note that the ordered context Ω must be empty since no ordered variables can occur in the argument to an intuitionistic application. □

5 Two-Level Framework

While the propositional OLL type theory suffices for representing CPS terms and their ordering properties, it is not enough for representing and reasoning about operations involving CPS terms such as evaluation and the CPS translation itself. Towards this end, we extend the ordered λ-calculus from Section 2 to a simple two-level logical framework. Level 2 type families p are indexed by level 1 objects M, and we can quantify only over level 1 objects.

$$\begin{array}{lrcl} \textit{Level 2 types} & F & ::= & p\, M_1 \ldots M_n \\ & & \mid & F_1 \rightarrow F_2 \\ & & \mid & F_1 \twoheadrightarrow F_2 \\ & & \mid & \Pi x{:}A.\ F \end{array}$$

$$\begin{array}{lrcl} \textit{Level 2 objects} & D & ::= & c \\ & & \mid & x \mid z \\ & & \mid & \lambda x{:}F.\ D \mid D_1\, D_2 \\ & & \mid & \overset{>}{\lambda} z{:}F.\ D \mid D_1 \,{}^{>}\, D_2 \\ & & \mid & \lambda x{:}A.\ D \mid D\, M \end{array}$$

The extended typing judgment now has the form $\Gamma; \Omega \vdash D : F$, where Γ may contain declarations of the form $x{:}A$ or $x{:}F$ and Ω contains declarations $z{:}F$. We omit the typing rules which are very similar to the propositional case, except that we now need a rule of type conversion:

$$\frac{\Gamma; \Omega \vdash D : F \qquad F \equiv_{\beta\eta} F'}{\Gamma; \Omega \vdash D : F'}$$

Since we stratify the type theory into two syntactically distinct levels, $\beta\eta$-equality for level-2 types immediately reduces to $\beta\eta$-equality for propositional objects. Since propositional objects possess canonical (= long $\beta\eta$-normal) forms, this equality is easy to decide, and type-checking in the fragment presented above can easily be seen to be decidable.

6 CPS Transform

We represent CPS transform with three basic types corresponding to the three judgements of the transform.

$$\mathsf{cps\_r} : \mathsf{droot} \to \mathsf{root} \to \mathsf{type}. \qquad \mathsf{cps\_e} : \mathsf{dexp} \to \mathsf{cont} \to \mathsf{exp} \to \mathsf{type}. \qquad \mathsf{cps\_t} : \mathsf{dtriv} \to \mathsf{triv} \to \mathsf{type}.$$

We then use the following terms in the two-level framework to construct representations of the CPS transform.

$$\begin{array}{rcl}
\mathsf{cps\_root} & : & \Pi E{:}\mathsf{dexp}.\ \Pi E'{:}\mathsf{cont} \twoheadrightarrow \mathsf{exp}. \\
 & & (\Pi k{:}\mathsf{cont}.\ \mathsf{cps\_e}\, E\, k\, (E'\,{}^{>}\, k)) \to \mathsf{cps\_r}\, (\mathsf{e2r}\, E)\, (\mathsf{klam}\, E'). \\
 & & \\
\mathsf{cps\_app} & : & \Pi E_0{:}\mathsf{dexp}.\ \Pi E_1{:}\mathsf{dexp}.\ \Pi C{:}\mathsf{cont}.\ \Pi E_1'{:}\mathsf{triv} \twoheadrightarrow \mathsf{exp}.\ \Pi E'{:}\mathsf{exp}. \\
 & & \mathsf{cps\_e}\, E_0\, (\mathsf{vlam}\,{}^{>}\, E_1')\, E' \to \\
 & & (\Pi v_0{:}\mathsf{triv}.\ \mathsf{cps\_e}\, E_1\, (\mathsf{vlam}\,{}^{>}\, \overset{>}{\lambda} v_1{:}\mathsf{triv}.\ \mathsf{app}\,{}^{>}\, C\,{}^{>}\, v_0\,{}^{>}\, v_1)\, (E_1'\,{}^{>}\, v_0)) \to \\
 & & \mathsf{cps\_e}\, (\mathsf{dapp}\, E_0\, E_1)\, C\, E'. \\
 & & \\
\mathsf{cps\_triv} & : & \Pi T{:}\mathsf{dtriv}.\ \Pi C{:}\mathsf{cont}.\ \Pi T'{:}\mathsf{triv}. \\
 & & \mathsf{cps\_t}\, T\, T' \to \mathsf{cps\_e}\, (\mathsf{t2e}\, T)\, C\, (\mathsf{kapp}\,{}^{>}\, C\,{}^{>}\, T'). \\
 & & \\
\mathsf{cps\_lam} & : & \Pi R{:}\mathsf{droot}.\ \Pi R'{:}\mathsf{root}. \\
 & & (\Pi x{:}\mathsf{dtriv}.\ \Pi x'{:}\mathsf{triv}.\ \mathsf{cps\_t}\, x\, x' \to \mathsf{cps\_r}\, (R\, x)\, (R'\, x')) \to \\
 & & \mathsf{cps\_t}\, (\mathsf{dlam}\, R)\, (\mathsf{lam}\, R').
\end{array}$$

We may now show the adequacy of above representation in two parts. In the informal translation we map variables x to themselves; in the formalization we map each variable x from the direct-style term to a corresponding variable x' in the continuation-passing term. These variables and their relationship are captured in contexts

$$\begin{array}{rcl}
\Gamma & = & x_1{:}\mathsf{dtriv} \ldots x_n{:}\mathsf{dtriv} \\
\Gamma' & = & x_1'{:}\mathsf{triv} \ldots x_n'{:}\mathsf{triv} \\
\Gamma_m & = & m_1{:}\mathsf{cps\_t}\, x_1\, x_1' \ldots m_n{:}\mathsf{cps\_t}\, x_n\, x_n'
\end{array}$$

which always occur together in this manner. In addition we have contexts

$$\begin{array}{rcl}
\Gamma_k & = & k_1{:}\mathsf{cont} \ldots k_m{:}\mathsf{cont} \\
\Gamma_v & = & v_1{:}\mathsf{triv} \ldots v_l{:}\mathsf{triv}
\end{array}$$

which include all the continuation parameters k and temporary variables v which may occur in the continuation c and CPS terms resulting from the translation. Note that ordering constraints are ignored during the translation, but will be nonetheless be satisfied by the resulting terms.

Theorem 4 (Representations are Canonical Forms) *Let $\Gamma^* = \Gamma, \Gamma', \Gamma_m, \Gamma_k, \Gamma_v$ be a context of the form explained above that contains all free variables occurring in the relevant judgment. Then*

1. $\vdash r \xrightarrow{DR} r'$ *implies* $\exists M.\ \Gamma^*; \cdot \vdash M \Uparrow \mathsf{cps\_r} \ulcorner r \urcorner^R \ulcorner r' \urcorner$.

2. $\vdash e\ ;\ c \xrightarrow{DE} e'$ *implies* $\exists M.\ \Gamma^*; \cdot; \cdot \vdash M \Uparrow \mathsf{cps\_e} \ulcorner e \urcorner^E \ulcorner c \urcorner \ulcorner e' \urcorner$

3. $\vdash t \xrightarrow{DT} t'$ *implies* $\exists M.\ \Gamma^*; \cdot; \cdot \vdash M \Uparrow \mathsf{cps\_t} \ulcorner t \urcorner^T \ulcorner t' \urcorner$

Proof: By structural induction on the given derivation. □

Theorem 5 (Canonical Forms are Representations) *Let* $\Gamma^* = \Gamma, \Gamma', \Gamma_m, \Gamma_k, \Gamma_v$ *a context of the form explained above and assume the types below are canonical.*

1. $\Gamma^*; \cdot; \cdot \vdash M \Uparrow \mathsf{cps\_r}\, R\, R'$ *implies* $\vdash \llcorner R \lrcorner_R \xrightarrow{DR} \llcorner R' \lrcorner$.

2. $\Gamma^*; \cdot; \cdot \vdash M \Uparrow \mathsf{cps\_e}\, E\, C\, E'$ *implies* $\vdash \llcorner E \lrcorner_E\ ;\ \llcorner C \lrcorner \xrightarrow{DE} \llcorner E' \lrcorner$.

3. $\Gamma^*; \cdot; \cdot \vdash M \Uparrow \mathsf{cps\_t}\, T\, T'$ *implies* $\vdash \llcorner T \lrcorner_T \xrightarrow{DR} \llcorner T' \lrcorner$.

Proof: By structural induction on the given canonical derivation. □

The adequacy of our representation gives us a simple proof that the terms resulting from a CPS transformation satisfy the occurrence conditions of section 3.

Theorem 6 $\vdash r \xrightarrow{DR} r'$ *implies* $\models^{\mathbf{Root}} r'$.

Proof: By theorem 4 we know $\cdot; \cdot; \cdot \vdash \ulcorner r' \urcorner \Uparrow \mathsf{root}$.
Then by theorem 3 we know $\models^{\mathbf{Root}} \llcorner \ulcorner r' \urcorner \lrcorner$.
Then we are done since $\llcorner \ulcorner r' \urcorner \lrcorner = r'$. □

The simplicity of the proof above may be surprising. It is so direct, because the work has been distributed to the proof of the adequacy theorems (which are clearly not trivial), combined with some deep properties of the logical framework such as the existence of canonical forms. This factoring of effort is typical in the use of logical frameworks.

It is also possible to represent a one-pass CPS transformation directly using third-order constructors and still guarantee ordering properties for the results. A further examination of this optimized translation is beyond the scope of this paper.

7 Bare Abstract Machine

We now begin extending our representation to include evaluation of CPS terms. We will begin by showing a representation of a naive evaluator which makes no use of the ordering invariants. The following is a bare abstract machine for CPS evaluation.

$$\frac{\vdash^{\mathbf{Exp}}_{\mathbf{B}} e \hookrightarrow a}{\vdash^{\mathbf{Root}}_{\mathbf{B}} \underline{\lambda} k.\ e \hookrightarrow a} \qquad \frac{}{\vdash^{\mathbf{Exp}}_{\mathbf{B}} k\, t \hookrightarrow t}$$

$$\frac{\vdash^{\mathbf{Exp}}_{\mathbf{B}} e[t/v] \hookrightarrow a}{\vdash^{\mathbf{Root}}_{\mathbf{B}} (\underline{\lambda} v.\ e)\, t \hookrightarrow a} \qquad \frac{\vdash^{\mathbf{Exp}}_{\mathbf{B}} e[t/x][c/k] \hookrightarrow a}{\vdash^{\mathbf{Root}}_{\mathbf{B}} (\underline{\lambda} x.\ \underline{\lambda} k.\ e)\, t\, c \hookrightarrow a}$$

Notice that this machine describes a regular big-step operational semantics for a λ-calculus—every redex is reduced by a substitution.

We introduce two type constructors to represent bare evaluations:

$$\mathsf{evalr}_B : \mathsf{root} \to \mathsf{triv} \to \mathsf{type}. \qquad \mathsf{evale}_B : \mathsf{exp} \to \mathsf{triv} \to \mathsf{type}.$$

Additionally, we introduce a new object to our signature for CPS terms:

$$\mathsf{ret} : \mathsf{cont}$$

ret [1] will be substituted for the continuation identifiers, k, in bare evaluations. In order to make the inverse representation function on objects well-defined, we augment it with a continuation identifier k when applied to a continuation term or a serious term:

$$\begin{array}{rcl}
\llcorner \mathsf{klam}\,(\overset{>}{\lambda} k{:}\mathsf{cont}.\ E)\lrcorner & = & \underline{\lambda} k.\ \llcorner E \lrcorner_k \\
\llcorner \mathsf{app}\,{}^{>}C\,{}^{>}T_0\,{}^{>}T_1\lrcorner_k & = & \llcorner T_0\lrcorner \llcorner T_1 \lrcorner \llcorner C \lrcorner_k \\
\llcorner \mathsf{kapp}\,{}^{>}C\,{}^{>}T\lrcorner_k & = & \llcorner C\lrcorner_k \llcorner T \lrcorner \\
\llcorner \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\ R)\lrcorner & = & \underline{\lambda} x.\ \llcorner R \lrcorner \\
\llcorner x \lrcorner & = & x \\
\llcorner v \lrcorner & = & v \\
\llcorner \mathsf{vlam}\,{}^{>}(\overset{>}{\lambda} v{:}\mathsf{triv}.\ E)\lrcorner_k & = & \underline{\lambda} v.\ \llcorner E \lrcorner_k \\
\llcorner k \lrcorner_k & = & k \\
\llcorner \mathsf{ret} \lrcorner_k & = & k
\end{array}$$

We use the following object constructors to represent bare evaluations:

$$\begin{array}{lcl}
\mathsf{evr}_B & : & \Pi A{:}\mathsf{triv}.\ \Pi E{:}\mathsf{cont} \twoheadrightarrow \mathsf{exp}. \\
 & & \mathsf{evale}_B\,(E\,{}^{>}\mathsf{ret})\,A \to \mathsf{evalr}_B\,(\mathsf{klam}\,E)\,A. \\
 & & \\
\mathsf{eve}_B\_0 & : & \Pi T{:}\mathsf{triv}.\ \mathsf{evale}_B\,(\mathsf{kapp}\,{}^{>}\mathsf{ret}\,{}^{>}T)\,T. \\
 & & \\
\mathsf{eve}_B\_1 & : & \Pi A{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}.\ \Pi E{:}\mathsf{triv} \twoheadrightarrow \mathsf{exp}. \\
 & & \mathsf{evale}_B\,(E\,{}^{>}T)\,A \to \mathsf{evale}_B\,(\mathsf{kapp}\,{}^{>}(\mathsf{vlam}\,{}^{>}E)\,{}^{>}T)\,A. \\
 & & \\
\mathsf{eve}_B\_\mathsf{app} & : & \Pi A{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}.\ \Pi C{:}\mathsf{cont}.\ \Pi E{:}\mathsf{triv} \to \mathsf{cont} \twoheadrightarrow \mathsf{exp}. \\
 & & \mathsf{evale}_B\,((E\,T)\,{}^{>}C)\,A \to \\
 & & \mathsf{evale}_B\,(\mathsf{app}\,{}^{>}C\,{}^{>}(\mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\ \mathsf{klam}\,(E\,x)))\,{}^{>}T)\,A.
\end{array}$$

We prove that our representation of bare evaluations is in bijective correspondence with actual bare evaluations in two parts as follows.

Theorem 7 (Bare evaluations are canonical forms) *Assume r and e have no free x.*

1. $\vDash^{\mathbf{Root}} r$, *and* $\vdash_B^{\mathbf{Root}} r \hookrightarrow a$ *implies* $\exists M.\ \ \cdot;\cdot \vdash M \Uparrow \mathsf{evalr}_B \ulcorner r \urcorner \ulcorner a \urcorner$.
2. $k \vDash^{\mathbf{Exp}} e$, *and* $\vdash_B^{\mathbf{Exp}} e \hookrightarrow a$ *implies* $\exists M.\ \ \cdot;\cdot \vdash M \Uparrow \mathsf{evale}_B\,(\ulcorner e \urcorner[\mathsf{ret}/k])\,\ulcorner a \urcorner$.

Proof: By structural induction on the given canonical derivation. □

[1] We chose the term ret for continuation identifiers since a continuation is not invoked until the end of the current computation, thus it is like a return statement.

Theorem 8 (Canonical forms are bare evaluations)

1. $\cdot; \cdot \vdash M \Uparrow \mathsf{evalr}_B\, R\, A$ *implies* $\vdash^{\mathbf{Root}} \llcorner R \lrcorner$ *and* $\vdash^{\mathbf{Root}}_{B} \llcorner R \lrcorner \hookrightarrow \llcorner A \lrcorner$.

2. $\cdot; \cdot \vdash M \Uparrow \mathsf{evale}_B\, E\, A$ *implies* $\vdash^{\mathbf{Exp}}_{B} \llcorner E \lrcorner_k \hookrightarrow \llcorner A \lrcorner$ *for any* k.

Proof: By induction on the given canonical typing derivation making use of α-conversion to allow choice of any k. □

In addition to proving that we really have represented bare evaluations, we have also proved that bare evaluation preserves the ordering invariants of CPS terms. This comes for free as a result of theorem 3.

8 Stack Abstract Machine

We now consider a more sophisticated evaluation model which makes use of the ordering constraints on CPS terms. Rather than substituting for continuations and continuation parameters, we can evaluate terms by keeping a stack of continuations and their parameters and then effectively treating k and v as pop instructions.

We need stacks, ϕ, of both trivial terms and continuation terms for our stack evaluator.

$$\phi ::= \cdot \mid \phi, t \mid \phi, c$$

We give a big step operational semantics for stack evaluation as follows:

$$\frac{\bullet \vdash^{\mathbf{Exp}}_{\mathbf{St}} e \hookrightarrow a}{\vdash^{\mathbf{Root}}_{\mathbf{St}} \underline{\lambda} k.\ e \hookrightarrow a} \qquad \frac{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} t \hookrightarrow a; \bullet}{\phi \vdash^{\mathbf{Exp}}_{\mathbf{St}} k\, t \hookrightarrow a}$$

$$\frac{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} t \hookrightarrow t'; \phi', \underline{\lambda} v.\ e \qquad \phi', t' \vdash^{\mathbf{Exp}}_{\mathbf{St}} e \hookrightarrow a}{\phi \vdash^{\mathbf{Exp}}_{\mathbf{St}} k\, t \hookrightarrow a} \qquad \frac{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} t \hookrightarrow t'; \phi' \qquad \phi', t' \vdash^{\mathbf{Exp}}_{\mathbf{St}} e \hookrightarrow a}{\phi \vdash^{\mathbf{Exp}}_{\mathbf{St}} (\underline{\lambda} v.\ e)\, t \hookrightarrow a}$$

$$\frac{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} t_1 \hookrightarrow t; \phi' \qquad \phi' \vdash^{\mathbf{Triv}}_{\mathbf{St}} t_0 \hookrightarrow \underline{\lambda} x.\ \underline{\lambda} k.\ e; \phi'' \qquad \phi'' \vdash^{\mathbf{Exp}}_{\mathbf{St}} e[t/x] \hookrightarrow a}{\phi \vdash^{\mathbf{Exp}}_{\mathbf{St}} t_0\, t_1\, k \hookrightarrow a}$$

$$\frac{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} t_1 \hookrightarrow t; \phi' \qquad \phi' \vdash^{\mathbf{Triv}}_{\mathbf{St}} t_0 \hookrightarrow \underline{\lambda} x.\ \underline{\lambda} k.\ e'; \phi'' \qquad \phi'', \underline{\lambda} v.\ e \vdash^{\mathbf{Exp}}_{\mathbf{St}} e'[t/x] \hookrightarrow a}{\phi \vdash^{\mathbf{Exp}}_{\mathbf{St}} t_0\, t_1\, (\underline{\lambda} v.\ e) \hookrightarrow a}$$

$$\frac{}{\phi \vdash^{\mathbf{Triv}}_{\mathbf{St}} \underline{\lambda} x.\ r \hookrightarrow \underline{\lambda} x.\ r; \phi} \qquad \frac{}{\phi, t \vdash^{\mathbf{Triv}}_{\mathbf{St}} v \hookrightarrow t; \phi}$$

Notice that this machine only performs substitution to reduce redices between trivial terms. Arguments to continuations are instead pushed onto the stack.

We show an OLF representation of the stack machine which uses the ordered context to represent the stack. We introduce the following new type constructors for representing stack evaluations:

$$\mathsf{evalr}_{St} : \mathsf{root} \to \mathsf{triv} \to \mathsf{type}. \qquad \mathsf{evale}_{St} : \mathsf{exp} \to \mathsf{triv} \to \mathsf{type}. \qquad \mathsf{evalt} : \mathsf{triv} \to \mathsf{triv} \to \mathsf{type}.$$

Since we use the ordered context as a stack, we also need constructors which allow us to put continuation terms and trivial terms into the ordered context:

$$\mathsf{cnt} : \mathsf{cont} \to \mathsf{type}. \qquad \mathsf{var} : \mathsf{triv} \to \mathsf{type}.$$

cnt and var will be used to represent continuation terms and trivial terms stored in the stack. We also introduce a new object:

$$\mathsf{pop} : \mathsf{triv}$$

which will be substituted for the continuation parameters, v during stack evaluation. As before we need to augment the inverse representation function with Φ as follows:

$$\begin{array}{rcl}
\llcorner \mathsf{klam}\,(\overset{>}{\lambda} k{:}\mathsf{cont}.\ E)\lrcorner & = & \underline{\lambda} k.\ \llcorner E \lrcorner_k \\
\llcorner \mathsf{app}^{>} C^{>} T_0{}^{>} T_1 \lrcorner_{\Phi\Phi_0\Phi_1} & = & \llcorner T_0 \lrcorner_{\Phi_0}\ \llcorner T_1 \lrcorner_{\Phi_1}\ \llcorner C \lrcorner_{\Phi} \\
\llcorner \mathsf{kapp}^{>} C^{>} T \lrcorner_{\Phi\Phi_t} & = & \llcorner C \lrcorner_{\Phi}\ \llcorner T \lrcorner_{\Phi_t} \\
\llcorner \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\ R)\lrcorner_{\cdot} & = & \underline{\lambda} x.\ \llcorner R \lrcorner \\
\llcorner x \lrcorner_{\cdot} & = & x \\
\llcorner v \lrcorner_{v} & = & v \\
\llcorner \mathsf{pop} \lrcorner_{v} & = & v \\
\llcorner \mathsf{vlam}^{>}(\overset{>}{\lambda} v{:}\mathsf{triv}.\ E)\lrcorner_{\Phi} & = & \underline{\lambda} v.\ \llcorner E \lrcorner_{\Phi,v} \\
\llcorner k \lrcorner_{k} & = & k \\
\llcorner \mathsf{ret} \lrcorner_{k} & = & k
\end{array}$$

The inverse representation function is stated non-deterministically– the splittings for Φ must be guessed. However, it is easy to see that the constraints on Φ in the variable cases ensure there will be at most one correct splitting of Φ at any point in the execution of the function.

We use the following term constructors to represent stack evaluations:

$$\begin{array}{lcl}
\mathsf{evr}_{St} & : & \Pi A{:}\mathsf{triv}.\ \Pi E{:}\mathsf{cont} \twoheadrightarrow \mathsf{exp}. \\
 & & \mathsf{evale}_{St}\,(E^{>}\mathsf{ret})\,A \to \mathsf{evalr}_{St}\,(\mathsf{klam}\,E)\,A. \\
 & & \\
\mathsf{eve}_{St}\_0 & : & \Pi A{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}. \\
 & & \mathsf{evalt}\,T\,A \twoheadrightarrow \mathsf{evale}_{St}\,(\mathsf{kapp}^{>}\mathsf{ret}^{>}T)\,A. \\
 & & \\
\mathsf{eve}_{St}\_1 & : & \Pi A{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}.\ \Pi T'{:}\mathsf{triv}.\ \Pi E{:}\mathsf{triv} \twoheadrightarrow \mathsf{exp}. \\
 & & (\mathsf{var}\,T' \twoheadrightarrow \mathsf{evale}_{St}\,(E^{>}\mathsf{pop})\,A) \twoheadrightarrow \\
 & & \mathsf{cnt}\,(\mathsf{vlam}^{>}E) \twoheadrightarrow \\
 & & \mathsf{evalt}\,T\,T' \twoheadrightarrow \\
 & & \mathsf{evale}_{St}\,(\mathsf{kapp}^{>}\mathsf{ret}^{>}T)\,A. \\
 & & \\
\mathsf{eve}_{St}\_2 & : & \Pi A{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}.\ \Pi T'{:}\mathsf{triv}.\ \Pi E{:}\mathsf{triv} \twoheadrightarrow \mathsf{exp}. \\
 & & (\mathsf{var}\,T' \twoheadrightarrow \mathsf{evale}_{St}\,(E^{>}\mathsf{pop})\,A) \twoheadrightarrow \\
 & & \mathsf{evalt}\,T\,T' \twoheadrightarrow \\
 & & \mathsf{evale}_{St}\,(\mathsf{kapp}^{>}(\mathsf{vlam}^{>}E)^{>}T)\,A.
\end{array}$$

$$
\begin{array}{lll}
\mathsf{eve}_{St}\mathsf{\_app\_0} & : & \Pi A{:}\mathsf{triv}.\ \Pi T_0{:}\mathsf{triv}.\ \Pi T_1{:}\mathsf{triv}.\ \Pi T{:}\mathsf{triv}.\ \Pi E{:}\mathsf{triv} \to \mathsf{cont} \twoheadrightarrow \mathsf{exp}.\\
 & & \mathsf{evale}_{St}\,((E\,T)\,{}^{>}\,\mathsf{ret})\,A \twoheadrightarrow\\
 & & \mathsf{evalt}\,T_0\,(\mathsf{lam}\,\lambda x{:}\mathsf{triv}.\ \mathsf{klam}\,(E\,x)) \twoheadrightarrow\\
 & & \mathsf{evalt}\,T_1\,T \twoheadrightarrow\\
 & & \mathsf{evale}_{St}\,(\mathsf{app}\,{}^{>}\,\mathsf{ret}\,{}^{>}\,T_0\,{}^{>}\,T_1)\,A.\\
 & & \\
\mathsf{eve}_{St}\mathsf{\_app\_1} & : & \Pi A{:}\mathsf{triv}.\ \Pi T_0{:}\mathsf{triv}.\ \Pi T_1{:}\mathsf{triv}.\ \Pi E'{:}\mathsf{triv} \twoheadrightarrow \mathsf{exp}.\ \Pi T{:}\mathsf{triv}.\ \Pi E{:}\mathsf{triv} \to \mathsf{cont} \twoheadrightarrow \mathsf{exp}.\\
 & & (\mathsf{cnt}\,(\mathsf{vlam}\,{}^{>}\,E') \twoheadrightarrow \mathsf{evale}_{St}\,((E\,T)\,{}^{>}\,\mathsf{ret})\,A) \twoheadrightarrow\\
 & & \mathsf{evalt}\,T_0\,(\mathsf{lam}\,\lambda x{:}\mathsf{triv}.\ \mathsf{klam}\,(E\,x)) \twoheadrightarrow\\
 & & \mathsf{evalt}\,T_1\,T \twoheadrightarrow\\
 & & \mathsf{evale}_{St}\,(\mathsf{app}\,{}^{>}\,(\mathsf{vlam}\,{}^{>}\,E')\,{}^{>}\,T_0\,{}^{>}\,T_1)\,A.\\
 & & \\
\mathsf{evt\_lam} & : & \Pi R{:}\mathsf{triv} \to \mathsf{root}.\ \mathsf{evalt}\,(\mathsf{lam}\,R)\,(\mathsf{lam}\,R).\\
 & & \\
\mathsf{evt\_vp} & : & \Pi T{:}\mathsf{triv}.\ \mathsf{var}\,T \twoheadrightarrow \mathsf{evalt}\,\mathsf{pop}\,T.
\end{array}
$$

Note that this representation does not contain an explicit stack. Instead, the ordered context of the type theory implicitly provides the representation of the evaluation machine's stack. In order to prove our representations are in bijective correspondence to stack evaluations, we need the following auxiliary definitions.

For any term u, $|\ulcorner u \urcorner|$ denotes $\ulcorner u \urcorner$ with all free k replaced by ret and all free v replaced by pop.

We define validity for evaluation stacks, ϕ, with respect to Φ (as defined in section 3), as follows:

$$
\frac{}{\models \cdot : \cdot}
\qquad
\frac{\cdot \models^{\mathbf{Triv}} t; \cdot \qquad \models \phi : \Phi}{\models (\phi, t) : (\Phi, v)}
\qquad
\frac{\Phi' \models^{\mathbf{cont}} c \qquad \models (\phi, \phi') : (\Phi, \Phi')}{\models (\phi, \phi', c) : (\Phi, \Phi', k)}
$$

Finally we need a representation function, and its inverse, for evaluation stacks.

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\ulcorner . \urcorner & = & \cdot & \llcorner \cdot \lrcorner & = & \cdot\\
\ulcorner (\phi, t) \urcorner & = & \ulcorner \phi \urcorner, vv{:}\mathsf{var}\ulcorner t \urcorner & \llcorner \Omega, vv{:}\mathsf{var}\,T \lrcorner & = & \llcorner \Omega \lrcorner, \llcorner T \lrcorner.\\
\ulcorner (\phi, c) \urcorner & = & \ulcorner \phi \urcorner, cv{:}\mathsf{cnt}\,|\ulcorner c \urcorner| & \llcorner \Omega, cv{:}\mathsf{cnt}\,C \lrcorner & = & \llcorner \Omega \lrcorner, \llcorner C \lrcorner_{\Phi}
\end{array}
$$

where $\models \llcorner \Omega_C \lrcorner : \Phi$ when $\Omega = \Omega', cv{:}\mathsf{cnt}\,C', \Omega_C$ and $\models \llcorner \Omega \lrcorner : \Phi$ when $cv{:}\mathsf{cnt}\,C' \notin \Omega$.

Theorem 9 (Stack evaluations are canonical forms) *Assume all r, e, and t have no free x.*

1. $\vdash^{\mathbf{Root}}_{St} r \hookrightarrow a$ *and* $\models^{\mathbf{Root}} r$ *implies* $\exists M.\ \ \cdot; \cdot \vdash M \Uparrow \mathsf{evalr}_{St} \ulcorner r \urcorner \ulcorner a \urcorner$.
2. $\phi', \phi \vdash^{\mathbf{Exp}}_{St} e \hookrightarrow a$ *and* $\models (\phi', \phi) : (\Phi', \Phi)$ *and* $\Phi \models^{\mathbf{Exp}} e$ *implies*
   $\exists M.\ \ \cdot; \ulcorner \phi', \phi \urcorner \vdash M \Uparrow \mathsf{evale}_{St}\,|\ulcorner e \urcorner|\,\ulcorner a \urcorner$.
3. $\phi', \phi \vdash^{\mathbf{Triv}}_{St} t \hookrightarrow a; \phi'$ *and* $\models (\phi', \phi) : (\Phi', \Phi)$ *and* $\forall \Phi''.\ \ \Phi'', \Phi \models^{\mathbf{Triv}} t; \Phi''$ *implies*
   $\exists M.\ \ \cdot; \ulcorner \phi \urcorner \vdash M \Uparrow \mathsf{evalt}\,|\ulcorner t \urcorner|\,\ulcorner a \urcorner$.

Proof: By structural induction on the given derivations. □

Theorem 10 (Canonical forms are stack evaluations)

1. $\cdot; \cdot; \cdot \vdash M \Uparrow \mathsf{evalr}_{St}\,R\,A$ *implies* $\vdash^{\mathbf{Root}}_{St} \llcorner R \lrcorner \hookrightarrow \llcorner A \lrcorner.$.
2. $\cdot; \cdot; \Omega \vdash M \Uparrow \mathsf{evale}_{St}\,E\,A$ *and* $\models \llcorner \Omega \lrcorner : \Phi$ *implies* $\llcorner \Omega \lrcorner \vdash^{\mathbf{Exp}}_{St} \llcorner E \lrcorner_{\Phi} \hookrightarrow \llcorner A \lrcorner.$.
3. $\cdot; \cdot; \Omega \vdash M \Uparrow \mathsf{evalt}_{St}\,T\,A$ *and* $\models \llcorner \Omega \lrcorner : \Phi$ *implies* $\phi', \llcorner \Omega \lrcorner \vdash^{\mathbf{Exp}}_{St} \llcorner T \lrcorner_{\Phi} \hookrightarrow \llcorner A \lrcorner.; \phi'$ *for any* ϕ'.

Proof: By structural induction on the given canonical derivations. □

9 Conclusion

We have shown that an ordered logical framework provides the necessary machinery for a natural encoding of CPS terms satisfying the given occurrence invariants. We have further shown that the framework is rich enough to allow a natural representation of stack-based evaluation. Furthermore we have seen that preservation of CPS invariants under evaluation is then trivial to prove. We feel this in itself is significant considering the difficulty involved in carrying out such representations in a framework with no inherent notion of order. Dzafic [Dza98] has shown how to represent system and properties closely related to ours in LF, with considerable overhead since stacks and the necessary substitution properties all have to be represented explicitly.

We conjecture that many systems with constrained resource access will have a natural representation in an ordered logical framework. Thus we believe further research into the meta-theory of the framework is justified. We also believe there are many possible applications of a logical framework based on a full type theory that includes unrestricted dependent types as well as linear and ordered connectives.

10 Acknowledgements

We would like to thank the anonymous referees for helpful comments on the presentation. We would also like to acknowledge many useful discussions with Olivier Danvy who inspired much of this work.

References

[Cer96] Iliano Cervesato. *A Linear Logical Framework*. PhD thesis, Dipartimento di Informatica, Università di Torino, February 1996.

[CP98] Iliano Cervesato and Frank Pfenning. A linear logical framework. *Information and Computation*, 1998. To appear in a special issue with invited papers from LICS'96, E. Clarke, editor.

[DDP99] Olivier Danvy, Belmina Dzafic, and Frank Pfenning. On proving syntactic properties of CPS programs. In Andrew Gordon and Andrew Pitts, editors, *Proceedings of the Third International Workshop on Higher Order Operational Techniques in Semantics (HOOTS'99)*, Paris, September 1999. Electronic Notes in Theoretical Computer Science, Volume 26.

[DP95] Olivier Danvy and Frank Pfenning. The occurrence of continuation parameters in CPS terms. Technical Report CMU-CS-95-121, Department of Computer Science, Carnegie Mellon University, February 1995.

[Dza98] Belmina Dzafic. Formalizing program transformations. Master's thesis, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark, December 1998.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:363–386, 1958.

[Pfe96] Frank Pfenning. The practice of logical frameworks. In Hélène Kirchner, editor, *Proceedings of the Colloquium on Trees in Algebra and Programming*, pages 119–134, Linköping, Sweden, April 1996. Springer-Verlag LNCS 1059. Invited talk.

[PP99a] Jeff Polakow and Frank Pfenning. Natural deduction for intuitionistic non-commutative linear logic. In J.-Y. Girard, editor, *Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications (TLCA'99)*, pages 295–309, L'Aquila, Italy, April 1999. Springer-Verlag LNCS 1581.

[PP99b] Jeff Polakow and Frank Pfenning. Relating natural deduction and sequent calculus for intuitionistic non-commutative linear logic. In Andre Scedrov and Achim Jung, editors, *Proceedings of the 15th Conference on Mathematical Foundations of Programming Semantics*, New Orleans, Louisiana, April 1999. Electronic Notes in Theoretical Computer Science, Volume 20.