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

Jeff Polakow and Frank Pfenning

LFM'00, Santa Barbara, CA
June 25, 2000

1. Introduction
2. A-normal form, ordering, and evaluation
3. Ordered Logical Framework
4. Continuation-passing style
5. Stack-based evaluation
6. Conclusion

Logical Frameworks

- Meta-languages for deductive systems
  (logics, type systems, operational semantics, ...)
- Some design goals
  - *Specification:* direct, elegant, concise
  - *Implementation:* simple, correct, usable
  - *Meta-Theory:* feasible, automated
- Support common concepts as directly as possible!

The Logical Framework LF

- Based on dependent types (Π, $\rightarrow$)
- Supports:
  - bound variables, substitution
  - parametric and hypothetical judgments
  - unification, logic programming, constraints (Elf)
  - meta-level induction, regular worlds (Twelf)
- [Harper, Honsell & Plotkin'87,'93] [Pf'89,'91] [Schürmann'00]

Linear Logical Framework (LLF)

- Conservative extension of LF
- Adds $\multimap$, &, $\top$ to support:
  - linear and strict variables, "contexts"
  - linear hypothetical judgments
  - state and resources
  - meta-level induction, reachable worlds (?)
- [Cervesato & Pf '96,'97,'00]

Ordered Logical Framework (OLF)

- Conservative extension of LF and LLF
- Adds $\twoheadrightarrow$ (right ordered implication)
  and $\rightarrowtail$ (left ordered implication) to support:
  - variable order restrictions
  - ordered hypothetical judgments
  - sequentiality constraints
  - stacks and queues
- [Polakow & Pf'99][Polakow'00]
- *Disclaimer:* Design and theory in progress!

This Talk

- A-normal forms, stacks, and ordering
- Ordered Logical Framework (OLF)
- Representing CPS terms in OLF
- Ordering properties of CPS known
  [Lawall'94] [Danvy & Pf'95] [Danvy, Dzafic & Pf'99]
- Simplified and unified here

A-Normal Forms

- Sequentialize computation
- Name intermediate results
- No explicit continuations
- Used in modern compilers
- Results extend to continuation-passing style

Translation to A-Normal Form

- For simplicity, source is untyped
- Source (pure λ-calculus)

$$\begin{array}{rrcl} \text{Serious Exps} & E & ::= & E_1\, E_2 \mid T \\ \text{Trivial Exps} & T & ::= & \underline{\lambda} x.\, E \mid x \end{array}$$

- Target (A-normal forms)

$$\begin{array}{rrcl} \text{Serious Exps} & e & ::= & \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e \mid \mathbf{return}\, t \\ \text{Trivial Exps} & t & ::= & \underline{\lambda} x.\, e \mid x \mid v \end{array}$$

- Distinguish λ-bound variables x (function parameters) from $\mathbf{let}$-bound variables v (intermediate results)

Example Translation

- Using call-by-value, left-to-right evaluation
- Show translation $|\cdot|$ of trivial expression

$$
\begin{aligned}
&|\underline{\lambda}x.\ f\,x\,(g\,x\,x)| \\
&= \underline{\lambda}x.\ \ \mathbf{let}\,v_1 = f\,x\ \ \mathbf{in} \\
&\qquad\quad \mathbf{let}\,v_2 = g\,x\ \ \mathbf{in} \\
&\qquad\quad \mathbf{let}\,v_3 = v_2\,x\ \ \mathbf{in} \\
&\qquad\quad \mathbf{let}\,v_4 = v_1\,v_3\ \ \mathbf{in} \\
&\qquad\quad \mathbf{return}\,v_4
\end{aligned}
$$

Stackability of Intermediate Values

- Intermediate values can be allocated on a stack

$$
\begin{array}{lll}
|\underline{\lambda}x.\ f\,x\,(g\,x\,x)| & & \\
= \underline{\lambda}x. & \mathbf{let}\, v_1 = f\,x\ \ \mathbf{in} & v_1 \\
& \mathbf{let}\, v_2 = g\,x\ \ \mathbf{in} & v_1, v_2 \\
& \mathbf{let}\, v_3 = v_2\,x\ \ \mathbf{in} & v_1, v_3 \\
& \mathbf{let}\, v_4 = v_1\,v_3\ \ \mathbf{in} & v_4 \\
& \mathbf{return}\, v_4 & .
\end{array}
$$

- Function parameters x do not participate
- Compile all v_i as $\mathbf{pop}$

Order as a Judgment

- Stacks $\Phi ::= \cdot \mid \Phi, v$
- Judgments

  $\Phi \Vdash e$ $\quad e$ is ordered in Φ

  $\Phi \vdash t$ $\quad t$ is ordered in Φ

$$\frac{\Phi, v \Vdash e \qquad \Phi_1 \vdash t_1 \qquad \Phi_2 \vdash t_2}{\Phi, \Phi_1, \Phi_2 \Vdash \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e}$$

$$\frac{\Phi \vdash t}{\Phi \Vdash \mathbf{return}\, t} \qquad \frac{\cdot \Vdash e}{\cdot \vdash \underline{\lambda} x.\ e}$$

$$\frac{}{\cdot \vdash x} \qquad \frac{}{v \vdash v}$$

Some Theorems

[Lawall'94] [Danvy & Pf'95] [Danvy, Dzafic & Pf'99]

- Results of translation will be ordered
- Ordered expressions can be evaluated with a stack
- Left-to-right and right-to-left give rise to symmetric orders
- Ordered expressions can be translated back

Representation in a Logical Framework

- First: higher-order abstract syntax
- Second: refine to capture order
- Third: evaluation (judgments about terms)
- $\ulcorner\!\urcorner e \urcorner\!\urcorner$ for serious expressions
- $\ulcorner t \urcorner$ for trival expressions

Representing A-Normal Forms, Take I

$$
\begin{array}{rcl}
\mathsf{exp} & : & \mathsf{type} \\
\mathsf{triv} & : & \mathsf{type} \\
\ulcorner v \urcorner & = & v \\
\urcorner\!\!\urcorner \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e \ulcorner\!\!\ulcorner & = & \mathsf{let}\,(\lambda v{:}\mathsf{triv}.\, \urcorner\!\!\urcorner e \ulcorner\!\!\ulcorner)\, \ulcorner t_1 \urcorner\, \ulcorner t_2 \urcorner \\
\mathsf{let} & : & (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv} \to \mathsf{triv} \to \mathsf{exp} \\
\urcorner\!\!\urcorner \mathbf{return}\, t \ulcorner\!\!\ulcorner & = & \mathsf{return}\, \ulcorner t \urcorner \\
\mathsf{return} & : & \mathsf{triv} \to \mathsf{exp} \\
\ulcorner x \urcorner & = & x \\
\ulcorner \underline{\lambda} x.\ e \urcorner & = & \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\, \urcorner\!\!\urcorner e \ulcorner\!\!\ulcorner) \\
\mathsf{lam} & : & (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv}
\end{array}
$$

Adequacy of LF Representations

- $\Gamma = x_1$:triv, $\ldots, x_n$:triv function parameters in e or t
- $\Omega = v_1$:triv, $\ldots, v_k$:triv temporary variables in e or t
- $\Gamma, \Omega \vdash \ulcorner e \urcorner$:exp and $\ulcorner e \urcorner$ is canonical
- $\Gamma, \Omega \vdash \lceil t \rceil$:triv and $\lceil t \rceil$ is canonical
- If $\Gamma, \Omega \vdash M$: exp, M canonical then $M = \ulcorner e \urcorner$ for some e
- If $\Gamma, \Omega \vdash M$: triv, M canonical then $M = \lceil t \rceil$ for some t
- $\ulcorner \cdot \urcorner$ and $\lceil \cdot \rceil$ are compositional bijections.

Ordered Adequacy

- Representations of ordered terms are well-typed
- Goal: all well-typed canonical terms represent ordered terms
- Refine framework to express order invariant
- Proceed from LF via LLF to OLF

Unrestricted Functions $A \to B$ (LF)

- Hypothetical judgment $\Gamma \vdash M : A$
- Context Γ satisfies exchange, weakening, and contraction

$$\frac{}{\Gamma, x{:}A, \Gamma' \vdash x : A}$$

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x{:}A.\ M : A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

Linear Functions $A \multimap B$ (LLF)

- Linear hypothetical judgment $\Delta \vdash M : A$
- Linear context Δ satisfies exchange

$$\frac{}{y{:}A \vdash y : A}$$

$$\frac{\Delta, y{:}A \vdash M : B}{\Delta \vdash \hat{\lambda} y{:}A.\ M : A \multimap B}$$

$$\frac{\Delta_1 \vdash M : A \multimap B \qquad \Delta_2 \vdash N : A}{\Delta_1 \bowtie \Delta_2 \vdash M \hat{} N : B}$$

- $\Delta_1 \bowtie \Delta_2$ allows interleaving merge

Right Ordered Functions $A \twoheadrightarrow B$ (OLF)

- Ordered hypothetical judgment $\Omega \vdash M : A$
- Ordered context Ω satisfies no structural rules

$$\frac{}{z{:}A \vdash z : A}$$

$$\frac{\Omega, z{:}A \vdash M : B}{\Omega \vdash \overset{>}{\lambda} z{:}A.\ M : A \twoheadrightarrow B}$$

$$\frac{\Omega_1 \vdash M : A \twoheadrightarrow B \qquad \Omega_2 \vdash N : A}{\Omega_1, \Omega_2 \vdash M \overset{>}{\ } N : B}$$

- Ω_1, Ω_2 concatentes ordered contexts
- Ordered variables are also linear

Left Ordered Functions $A \rightarrowtail B$ (OLF)

- Same ordered context Ω

$$\frac{}{z{:}A \vdash z : A}$$

$$\frac{z{:}A, \Omega \vdash M : B}{\Omega \vdash \lambda^{<} z{:}A.\ M : A \rightarrowtail B}$$

$$\frac{\Omega_1 \vdash M : A \rightarrowtail B \qquad \Omega_2 \vdash N : A}{\Omega_2, \Omega_1 \vdash M \ ^{<} N : B}$$

Joint Propositional System

- Judgment $\Gamma; \Delta; \Omega \vdash M : A$
- Γ, Δ, and Ω interact straightforwardly

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

- Additives & and $\top$ permitted
- Substitution and subject reduction hold
- Unique canonical forms exist [Polakow & Pf'99]

Representing A-Normal Forms, Take II

- LF Representation

$$\begin{array}{rcl} \ulcorner v \urcorner & = & v \\ \ulcorner\!\ulcorner \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e \urcorner\!\urcorner & = & \mathsf{let}\,(\lambda v{:}\mathsf{triv}.\ \ulcorner\!\ulcorner e \urcorner\!\urcorner)\ \ulcorner t_1 \urcorner\ \ulcorner t_2 \urcorner \\ \mathsf{let} & : & (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv} \to \mathsf{triv} \to \mathsf{exp} \end{array}$$

- Ordering of **let**

$$\frac{\Phi, v \Vdash e \qquad \Phi_1 \vdash t_1 \qquad \Phi_2 \vdash t_2}{\Phi, \Phi_1, \Phi_2 \Vdash \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e} \qquad \frac{}{v \vdash v}$$

- Ordered Representation

$$\begin{array}{rcl} \ulcorner\!\ulcorner v \urcorner\!\urcorner & = & v \\ \ulcorner\!\ulcorner \mathbf{let}\, v = t_1\, t_2\ \mathbf{in}\ e \urcorner\!\urcorner & = & \mathsf{let}^{>}(\lambda^{>} v{:}\mathsf{triv}.\ \ulcorner\!\ulcorner e \urcorner\!\urcorner)\ {}^{>}\ulcorner t_1 \urcorner\ {}^{>}\ulcorner t_2 \urcorner \\ \mathsf{let} & : & (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{exp} \end{array}$$

Representing A-Normal Forms, Take II Ctd.

- LF representation

$$\begin{array}{rcl} \ulcorner\mathbf{return}\, t\urcorner & = & \mathsf{return}\ulcorner t\urcorner \\ \mathsf{return} & : & \mathsf{triv} \to \mathsf{exp} \end{array}$$

- Ordering of $\mathbf{return}$

$$\frac{\Phi \vdash t}{\Phi \Vdash \mathbf{return}\, t}$$

- Ordered representation

$$\begin{array}{rcl} \ulcorner\mathbf{return}\, t\urcorner & = & \mathsf{return}\ulcorner t\urcorner \\ \mathsf{return} & : & \mathsf{triv} \twoheadrightarrow \mathsf{exp} \end{array}$$

Representing A-Normal Forms, Take II Ctd.

- LF representation

$$\begin{array}{rcl} \ulcorner x \urcorner & = & x \\ \ulcorner \underline{\lambda} x.\, e \urcorner & = & \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\, \ulcorner e \urcorner) \\ \mathsf{lam} & : & (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv} \end{array}$$

- Ordering of $\underline{\lambda}$

$$\frac{\cdot \Vdash e}{\cdot \vdash \underline{\lambda} x.\, e} \qquad\qquad \frac{}{\cdot \vdash x}$$

- Ordered representation

$$\begin{array}{rcl} \ulcorner \underline{\lambda} x.\, e \urcorner & = & \mathsf{lam}\,(\lambda x{:}\mathsf{triv}.\, \ulcorner e \urcorner) \\ \mathsf{lam} & : & (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv} \end{array}$$

Summary Signature

$$\begin{aligned} \mathsf{let} &: (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{exp} \\ \mathsf{return} &: \mathsf{triv} \twoheadrightarrow \mathsf{exp} \\ \mathsf{lam} &: (\mathsf{triv} \to \mathsf{exp}) \to \mathsf{triv} \end{aligned}$$

Ordered Adequacy

- $\Gamma = x_1$:triv, $\ldots, x_n$:triv
- $\Omega = v_1$:triv, $\ldots, v_k$:triv
- $\Phi = v_1, \ldots, v_k$
- $\Gamma; \cdot; \Omega \vdash \ulcorner e \urcorner$: exp and $\ulcorner e \urcorner$ is canonical for $\Phi \Vdash e$
- $\Gamma; \cdot; \Omega \vdash \lceil t \rceil$: triv and $\lceil t \rceil$ is canonical for $\Phi \vdash t$
- If $\Gamma; \cdot; \Omega \vdash M$: exp, M canonical then $M = \ulcorner e \urcorner$ for some e with $\Phi \Vdash e$
- If $\Gamma; \cdot; \Omega \vdash M$: triv, M canonical then $M = \lceil t \rceil$ for some t with $\Phi \vdash t$
- $\ulcorner \cdot \urcorner$ and $\lceil \cdot \rceil$ are compositional bijections

Continuation-Passing Style

- Generalize from A-normal form
- Make continuation explicit
- Target (CPS terms)

$$\begin{array}{rrcl} \text{Serious Exps} & e & ::= & t_1\, t_2\, (\underline{\lambda} v.\ e) \mid k\, t \\ \text{Trivial Exps} & t & ::= & \underline{\lambda} x.\ \underline{\lambda} k.\ e \mid x \mid v \end{array}$$

- Distinguish λ-bound variables x (function parameters) from $\mathbf{let}$-bound variables v (intermediate results) from continuation parameters k

Example Translation

- Using call-by-value, left-to-right evaluation
- Instead of returning, call continuation
- Pass continuation to every function

$$
\begin{aligned}
&|\underline{\lambda}x.\ f\,x\,(g\,x\,x)| \\
&= \underline{\lambda}x.\ \underline{\lambda}k.\ \ f\,x\,(\underline{\lambda}v_1. \\
&\qquad\qquad g\,x\,(\underline{\lambda}v_2. \\
&\qquad\qquad v_2\,x\ (\underline{\lambda}v_3. \\
&\qquad\qquad v_1\,v_3\,(\underline{\lambda}v_4. \\
&\qquad\qquad k\,v_4))))
\end{aligned}
$$

Ordered CPS Encoding

- Joint stack for intermediate values and continuations
- app corresponds to let
- return M becomes $k\,M$ for bound k.
- Representation is order-adequate as before
- Uses third-order constructors

$$\begin{aligned} \mathsf{app} &: (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{triv} \twoheadrightarrow \mathsf{exp} \\ \mathsf{lam} &: (\mathsf{triv} \to (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \twoheadrightarrow \mathsf{exp}) \to \mathsf{triv} \end{aligned}$$

Dependent Types in OLF

- Dependent function type $\Pi x{:}A.\ B$ plausible
- Construct in analogy with linear LF
- Paper: 2-level (first-order) system
- Meta-theory simple
- Used for judgments on ordered terms

Some Judgments on Ordered Terms

- Requires 2-level dependent types
- A-normal form translation, generates only ordered terms
- CPS translation, generates only ordered terms
- Evaluation (direct and with stack-based machine)
- Preserves order (guaranteed by OLF)
- (see paper for details)

Example: Stack Abstract Machine

- Evaluation stack $\phi ::= \cdot \mid \phi, \underline{\lambda} v.\ e \mid \phi, t$
- Mixes continuations and values
- Evaluation judgments

  $\phi \Vdash e \hookrightarrow a$ $\quad$ e evaluates to answer a

  $\phi \vdash t \hookrightarrow t'; \phi'$ $\quad$ t evaluates to t' and stack ϕ'

$$\frac{\phi \vdash t_2 \hookrightarrow t_2'; \phi' \quad \phi' \vdash t_1 \hookrightarrow \underline{\lambda} x.\ \underline{\lambda} k.\ e_1; \phi'' \quad \phi'', \underline{\lambda} v.\ e \Vdash [t_2'/x] e_1 \hookrightarrow a}{\phi \Vdash t_1\, t_2\, (\underline{\lambda} v.\ e) \hookrightarrow a}$$

$$\frac{\phi \vdash t \hookrightarrow t'; \phi', \underline{\lambda} v.\ e \qquad \phi', t' \Vdash e \hookrightarrow a}{\phi \Vdash k\, t \hookrightarrow a}$$

$$\frac{}{\phi \vdash \underline{\lambda} x.\ \underline{\lambda} k.\ e \hookrightarrow \underline{\lambda} x.\ \underline{\lambda} k.\ e; \phi} \qquad \frac{}{\phi, t \vdash v \hookrightarrow t; \phi}$$

Machine Properties

- Correct for ordered terms
  [Danvy & Pf'95] [Danvy, Dzafic & Pf'99]
- Paper: proof with the aid of OLF
- Temporary variables v treated as $\mathbf{pop}$
- Continuation parameters k treated as return.

Representation in OLF

- Add

$$\begin{array}{rcl} \mathsf{pop} & : & \mathsf{triv} \\ \mathsf{ret} & : & \mathsf{triv} \twoheadrightarrow \mathsf{exp} \\ \mathsf{cont} & : & (\mathsf{triv} \twoheadrightarrow \mathsf{exp}) \to \mathsf{type} \\ \mathsf{var} & : & \mathsf{triv} \to \mathsf{type} \\ \mathsf{eval} & : & \mathsf{exp} \to \mathsf{triv} \to \mathsf{type} \\ \mathsf{tval} & : & \mathsf{triv} \to \mathsf{triv} \to \mathsf{type} \end{array}$$

- Represent evaluation stack ϕ as ordered context with

$$\begin{array}{rcl} \ulcorner . \urcorner & = & . \\ \ulcorner \phi, \underline{\lambda} v.\ e \urcorner & = & \ulcorner \phi \urcorner, k_i{:}\mathsf{cont}\ (\overset{>}{\lambda} v{:}\mathsf{triv}.\ \urcorner\!\urcorner e \urcorner\!\urcorner) \\ \ulcorner \phi, t \urcorner & = & \ulcorner \phi \urcorner, v_i{:}\mathsf{var}\ \ulcorner t \urcorner \end{array}$$

Computation Judgment

$$\begin{aligned}
\text{evapp} \;:\; & \text{eval}\ (\text{app}\,{}^{>}(\overset{>}{\lambda} v.\ e\,{}^{>}v)\,{}^{>}t_1\,{}^{>}t_2)\ a \\
& \quad \twoheadleftarrow \text{tval}\ t_2\ t_2' \\
& \quad \twoheadleftarrow \text{tval}\ t_1\ (\text{lam}\ (\lambda x.\ \overset{>}{\lambda} k.\ e_1\, x\,{}^{>}k)) \\
& \quad \twoheadleftarrow (\text{cont}\ (\overset{>}{\lambda} v.\ e\,{}^{>}v) \twoheadrightarrow \text{eval}\ (e_1\, t_2'\,{}^{>}\text{ret})\ a) \\
\text{evret} \;:\; & \text{eval}\ (\text{ret}\,{}^{>}t)\ a \\
& \quad \twoheadleftarrow \text{tval}\ t\ t' \\
& \quad \twoheadleftarrow \text{cont}\ (\overset{>}{\lambda} v.\ e\,{}^{>}v) \\
& \quad \twoheadleftarrow (\text{var}\ t' \twoheadrightarrow \text{eval}\ (e\,{}^{>}\text{pop})\ a) \\
\text{evlam} \;:\; & \text{tval}\ (\text{lam}\ (\lambda x.\ \overset{>}{\lambda} k.\ e\, x\,{}^{>}k))\ (\text{lam}\ (\lambda x.\ \overset{>}{\lambda} k.\ e\, x\,{}^{>}k)) \\
\text{evpop} \;:\; & \text{tval}\ (\text{pop})\ t \\
& \quad \twoheadleftarrow \text{var}\ t
\end{aligned}$$

Remarks on Implementation

- Only right implication required
- Quantifiers omitted
- Adequacy (see paper)
- Preservation of order guaranteed by OLF typing
- Object language types can be added

Other Applications

- Computational linguistics [Lambek'58] (word order)
- Abstract machines (stacks)
- De Bruijn indices (variable order)
- Ordered logic programming [Polakow'00]
- Stack and queue data structures

Related Work

- Non-commutative logic [Ruet'97, ...]
  Classical, no proof terms, based on non-commutative and commutative conjunction and disjunction
- Natural deduction and sequent calculus for full ordered logic [Polakow & Pf'99]
- Ordering judgment explicitly in LF [Dzafic'99]

Future Work

- Efficient logic programming search [Polakow'00]
- Ordered higher-order unification?
- Fully dependent type theory?
- Meta-theorem proving?
- Further examples