Verified Compilation of Weak Call-by-Value λ-Calculus into Combinators and Closures

Bachelor's Talk

Fabian Kunze

Advisor: Prof. Dr. Gert Smolka

SAARLAND UNIVERSITY
COMPUTER SCIENCE
Programming Systems Lab

11.12.2015

L: Weak Call-by-Value λ-Calculus

$$s, t, u \;::=\; x \mid \lambda x.s \mid st \qquad (x \in \mathbb{N})$$

$$x^x_u := u \qquad (st)^x_u := s^x_u t^x_u$$

$$y^x_u := y \qquad (\lambda y.s)^x_u := \lambda y.(s^x_u)$$

$$\frac{s \succ_L s'}{st \succ_L s't} \qquad \frac{s \succ_L t'}{st \succ_L st'} \qquad \frac{}{(\lambda x.s)(\lambda y.t) \succ_L s^x_{\lambda y.t}}$$

- Turing complete
- Data can be represented as procedure (closed λ-abstraction) using Church encodeding.

SK Combinatory Logic

$$X, Y, Z \;::=\; x \mid \mathbf{S} \mid \mathbf{K} \mid XY \qquad (x \in \mathbb{N})$$

$$\frac{}{\mathbf{K}XY \succ X} \qquad \frac{}{\mathbf{S}XYZ \succ (XZ)(YZ)} \qquad \frac{X \succ X'}{XY \succ X'Y} \qquad \frac{Y \succ Y'}{XY \succ XY'}$$

- also called SKI, but combinator **I** can be defined:

$$\mathbf{I} := \mathbf{SKK}$$

$$\mathbf{I}X = \mathbf{SKK}X \succ \mathbf{K}X(\mathbf{K}X) \succ X$$

SK Combinatory Logic (2)

SK can "simulate" substitution:

Example

$\lambda x.(xy) \sim \mathbf{SI}(\mathbf{K}y)$:

$$(\lambda x.(xy))z \succ zy$$

$$\mathbf{SI}(\mathbf{K}y)z \succ (\mathbf{I}z)(\mathbf{K}yz) \succ^* zy$$

- **S** : 'push' argument down in application
- **K** : discard 'pushed' argument
- **I** : take 'pushed' argument

- λ-calculus can be embedded in SK (but altered SK-equivalence).
- We will embed L into a call-by-value version of SK!

Content

SKv: Call-by-Value Combinatory Logic

$$X, Y, Z \ ::= \ x \mid \mathsf{K} \mid \mathsf{S} \mid XY \qquad (x \in \mathbb{N})$$

$$\mathrm{Val} \ni X, Y \ ::= \ x \mid \mathsf{K} \mid \mathsf{K}X \mid \mathsf{S} \mid \mathsf{S}X \mid \mathsf{S}XY \qquad (x \in \mathbb{N})$$

$$\frac{X \succ_{\mathrm{SK}} X'}{XY \succ_{\mathrm{SK}} X'Y} \qquad \frac{Y \succ_{\mathrm{SK}} Y'}{XY \succ_{\mathrm{SK}} XY'} \qquad \frac{X, Y \in \mathrm{Val}}{\mathsf{K}XY \succ_{\mathrm{SK}} X} \qquad \frac{X, Y, Z \in \mathrm{Val}}{\mathsf{S}XYZ \succ_{\mathrm{SK}} XZ(YZ)}$$

- If $X_1 \succ_{\mathrm{SK}}^{k_1} Y_1$ and $X_2 \succ_{\mathrm{SK}}^{k_2} Y_2$, then $X_1X_2 \succ_{\mathrm{SK}}^{k_1+k_2} Y_1Y_2$.
- Values are irreducible, and closed irreducible terms are values.
- $\mathsf{I} := \mathsf{SKK}$ yield $\mathsf{I}X \succ_{\mathrm{SK}}^{2} X$

Uniform Confluence

Uniform Confluence

$$\begin{tikzcd}
& x \arrow[dl, "k_1"'] \arrow[dr, "k_2"] & \\
y_1 \arrow[dr, dotted, "k_2 \geq l_1"'] & \begin{smallmatrix} k_1 \\ + \\ l_1 \end{smallmatrix} = \begin{smallmatrix} k_2 \\ + \\ l_2 \end{smallmatrix} & y_2 \arrow[dl, dotted, "l_2 \leq k_1"] \\
& z &
\end{tikzcd}$$

Uniform Diamond

If $y_1 \leftarrow x \rightarrow y_2$, then either $y_1 = y_2$ or $\exists z, y_1 \rightarrow z \leftarrow y_2$.

- $\Rightarrow$: Take $k_1 = k_2 = 1$.
- $\Leftarrow$: Induction on k_1 and k_2.

Call-by-value systems (like L and SKv) have the uniform diamond:
Redexes are not nested, so the two reductions either contract the same redex ($y_1 = y_2$).
Or they contract disjoint redexes; contracting both joins y_1 and y_2.

Substitution in SKv

$$x^x_Z := u \qquad \mathbf{K}^x_Z := \mathbf{K} \qquad (XY)^x_Z := X^x_Z Y^x_Z$$
$$y^x_Z := y \qquad \mathbf{S}^x_Z := \mathbf{S}$$

Substitutivity:

- If Y is closed and $x \neq z$, then $z \in \mathsf{FV}(X)$ iff $z \in \mathsf{FV}(X^x_Y)$.

No similar lemma for values, e.g. $x\mathbf{K} \notin \mathrm{Val}$, but $(x\mathbf{K})^x_{\mathbf{K}} = \mathbf{KK} \in \mathrm{Val}$.

X is a *maximal value* iff $X \in \mathit{Val}$, but $XY \notin \mathrm{Val}$

So maximal values are values of form x, $\mathbf{K}X$ and $\mathbf{S}XY$.

- If Y is a maximal value, then $X \in \mathrm{Val}$ iff $X^x_Y \in \mathrm{Val}$.

Pseudo-Abstraction

$$
\begin{aligned}
[x].x &:= \mathsf{I} & \\
[x].X &:= \mathsf{K}X & \text{if } x \notin \mathsf{FV}(X) \land X \in \mathrm{Val} \\
[x].(XY) &:= \mathsf{S}([x].X)([x].Y) & \text{otherwise}
\end{aligned}
$$

Similarities to L-abstractions:

- $[x].X$ is maximal value.
- Y value $\Longrightarrow ([x].X)Y \succ_{\mathrm{SK}}^{+} X_Y^x$
- $\mathsf{FV}([x].X) = \mathsf{FV}(X) \setminus \{x\}$

Commutes with Substitution:

- Y maximal value $\land\, z \notin \mathsf{FV}(Y) \land z \neq x \Longrightarrow ([z].X)_Y^x = [z].(X_Y^x)$

Proof by Induction on X.
Crucial: $z \notin \mathsf{FV}(X) \land X \in \mathrm{Val} \Longleftrightarrow z \notin \mathsf{FV}(X_Y^x) \land X_Y^x \in \mathrm{Val}$

Compiling L into SKv

$$\mathcal{C}\,x := x$$
$$\mathcal{C}\,(st) := (\mathcal{C}\,s)(\mathcal{C}\,t)$$
$$\mathcal{C}\,(\lambda x.s) := [x].(\mathcal{C}\,s)$$

For readability: $\underline{X} := \mathcal{C}\,X$

- If s is closed, then s is abstraction iff $\mathcal{C}\,s$ is a (maximal) value.
  - Thus a closed s is L-redex iff $\underline{s}$ is SKv-redex
- If t is a procedure, then $\underline{s}^x_{\underline{t}} = \underline{s^x_t}$

Soundness

If s is closed and $s \succ_{\mathsf{L}} t$, then $\mathcal{C}\,s \succ^{+}_{\mathrm{SK}} \mathcal{C}\,t$.

Implies (for closed s):

$$s \succ^{*}_{\mathsf{L}} t \Rightarrow \mathcal{C}\,s \succ^{*}_{\mathrm{SK}} \mathcal{C}\,t \quad \text{and} \quad s \Downarrow t \Rightarrow \mathcal{C}\,s \Downarrow \mathcal{C}\,t$$

Left-Invertibility of Compilation

$$
\begin{aligned}
[x]^{-1}.(\mathsf{SKK}) &:= x \\
[x]^{-1}.(\mathsf{S}XY) &:= ([x]^{-1}.X)([x]^{-1}.Y) \\
[x]^{-1}.(\mathsf{K}X) &:= X
\end{aligned}
$$

$$[x]^{-1}.([x].X) = X$$

$$
\begin{aligned}
\mathcal{C}^{\text{-}1}\, x &:= x \\
\mathcal{C}^{\text{-}1}\, X &:= \lambda x.(\mathcal{C}^{\text{-}1}\,([x]^{-1}.X)) && \text{if } X \in \mathrm{Val} \\
\mathcal{C}^{\text{-}1}\,(XY) &:= (\mathcal{C}^{\text{-}1}\, X)(\mathcal{C}^{\text{-}1}\, Y)
\end{aligned}
$$

$$\mathcal{C}^{\text{-}1}\,(\mathcal{C}\, s) = s$$

So $\mathcal{C}$ is injective (modulo α-conversion).

Completeness

$\mathcal{C}\ s \succ^*_{\mathrm{SK}} \mathcal{C}\ t \Longrightarrow s \succ^*_{\mathsf{L}} t$ does not hold:

Example

For a (reasonable) procedure u:
$\underline{(\lambda x.xx)u} = \mathbf{SII}\underline{u} \succ_{\mathrm{SK}} (\mathbf{I}\underline{u})\ (\mathbf{I}\underline{u}) = \underline{((\lambda x.x)u)\ ((\lambda x.x)u)}$,
but $(\lambda x.xx)u \not\succ^*_{\mathsf{L}} ((\lambda x.x)u)\ ((\lambda x.x)u)$

SKv-reductions can be extended to correspond to L-reductions:

$$s \text{ closed} \wedge \mathcal{C}\ s \succ_{\mathrm{SK}} X \Longrightarrow \exists t, X \succ^*_{\mathrm{SK}} \mathcal{C}\ t \wedge s \succ_{\mathsf{L}} t$$

Completeness on normalizing terms

SKv-reductions can be extended corresponding to L-reductions:

$$s \text{ closed} \wedge \mathcal{C}\, s \succ_{\mathrm{SK}} X \Longrightarrow \exists t, X \succ_{\mathrm{SK}}^{*} \mathcal{C}\, t \wedge s \succ_{\mathsf{L}} t$$

Proof

Induction on $\underline{s} \succ_{\mathrm{SK}} X$:

- $\underline{s}$ is SKv-redex
  $\Rightarrow s = (\lambda x.s')u$, where u procedure
  - successor of $\underline{s}$ unique: X
  - $\underline{s} = \underline{(\lambda x.s')u} \succ_{\mathrm{SK}}^{+} \underline{(s'^{x}_{u})}$

  $\Rightarrow t := s'^{x}_{u}$ has claimed properties
- $\underline{s}$ not SKv-redex
  $\Rightarrow s = s_1 s_2$, redex contained in $\underline{s_1}$ or $\underline{s_2}$
  $\Rightarrow$ claim holds by inductive hypothesis

Completeness on normalizing terms(2)

$$s \text{ closed} \land \mathcal{C}\, s \succ_{\mathrm{SK}} X \Longrightarrow \exists t, X \succ_{\mathrm{SK}}^{*} \mathcal{C}\, t \land s \succ_{\mathsf{L}} t$$

Generalizes to reduction chains:

$$s \text{ closed} \land \mathcal{C}\, s \succ_{\mathrm{SK}}^{*} X \Longrightarrow \exists t, X \succ_{\mathrm{SK}}^{*} \mathcal{C}\, t \land s \succ_{\mathsf{L}}^{*} t$$

Proof

Induction on length of $\underline{s} \succ_{\mathrm{SK}}^{k} X$:

- $k = 0$: trivial
- $k = 1 + k'$: extend, uniform confluence and inductive hypothesis

$$\begin{tikzcd}
\underline{s} \arrow[r] & Y \arrow[r, "k'"] \arrow[d, "*"] & X \arrow[d, "*"] & \\
& \underline{u} \arrow[r, "\leq k'"] & X' \arrow[r, "*"] & \underline{t}
\end{tikzcd}
\qquad
\begin{tikzcd}
s \arrow[dr] & & \\
& u \arrow[r, "*"] & t
\end{tikzcd}$$

Completeness on normalizing terms (3)

$$s \text{ closed} \wedge \mathcal{C}\, s \succ_{\mathrm{SK}}^{*} X \Longrightarrow \exists t, X \succ_{\mathrm{SK}}^{*} \mathcal{C}\, t \wedge s \succ_{\mathsf{L}}^{*} t$$

For normalizing, closed s:

$$\mathcal{C}\, s \Downarrow X \Longrightarrow s \Downarrow \mathcal{C}^{-1}\, X$$

Combined with soundness:

$$\mathcal{C}\, s \Downarrow \mathcal{C}\, t \Longleftrightarrow s \Downarrow t$$

This is satisfying for a term having a normal form, but we can do better!

Completeness on arbitrary terms

We want: $\mathcal{C}\, s \succ_{\mathrm{SK}}^{*} \mathcal{C}\, t \Rightarrow s \equiv_{\mathsf{L}} t$.
We study the $\mathcal{C}$-image of (closed) β-redexes, depending on the body:

Assume an abstraction s and $\underline{(\lambda x.s)t} \succ_{\mathrm{SK}}^{*} \underline{u}$. Then $(\lambda x.s)t \equiv_{\mathsf{L}} u$

Proof

We have $(\lambda x.s)t \Downarrow s^x_t$.
We use completeness on normalizing terms and confluence of SKv.

For other bodies:

Assume a non-abstraction s and $\underline{(\lambda x.s)t} \succ_{\mathrm{SK}} Y$.
Then there is a closed s' with $\underline{s'} = Y$ and $(\lambda x.s)t \equiv_{\mathsf{L}} s'$

Proof

By exhausting case distinction on s (variable and closed or non-closed application).

Completeness on arbitrary terms (2)

$$s \text{ closed} \wedge \mathcal{C}\, s \succ_{\mathrm{SK}}^{k} \mathcal{C}\, t \Longrightarrow s \equiv_{\mathsf{L}} t$$

Idea: Decompose $X_1 X_2 \succ_{\mathrm{SK}}^{k} Y_1 Y_2$:

$$X_i \succ_{\mathrm{SK}}^{k_i} Y_i \;\bigvee\; X_i \succ_{\mathrm{SK}}^{k} Z_i \wedge \underbrace{Z_1 Z_2}_{\textit{redex}} \succ_{\mathrm{SK}} Y \wedge Y \succ_{\mathrm{SK}}^{k'} Y_1 Y_2$$

Proof by Lexicographic induction on (k, s)

$s = \lambda x.s'$: $\underline{s} = \underline{t} \Rightarrow s = t$ by injectivity.

$t = \lambda x.t'$: completeness on normalizing terms.

Decompose $\underline{s} = \underline{s_1}\ \underline{s_2} \succ_{\mathrm{SK}}^{k} \underline{t_1}\ \underline{t_2} = \underline{t}$:

- $\underline{s_i} \succ_{\mathrm{SK}}^{k_i} \underline{t_i}$ with $k_i \leq k$: inductive hypothesis $\Rightarrow s_i \equiv_{\mathsf{L}} u_i$
- $\underline{s_i} \succ_{\mathrm{SK}}^{k_i} \underline{u_i}$; $\underline{u_1}\ \underline{u_2} \succ_{\mathrm{SK}} Y$; $Y \succ_{\mathrm{SK}}^{k'} \underline{t}$; $k_1 + k_2 + 1 + k' = k$; u_i procedures. By inductive hypothesis: $s_i \equiv_{\mathsf{L}} u_i$.
  - Body of u_1 abstraction: $u_1 u_2 \equiv_{\mathsf{L}} t$. So $s_1 s_2 \equiv_{\mathsf{L}} u_1 u_2 \equiv_{\mathsf{L}} t$
  - Body of u_1 non-abstraction: $\exists s'$ such that $Y = \underline{s'}$ and $u_1 u_2 \equiv_{\mathsf{L}} s'$. So $s_1 s_2 \equiv_{\mathsf{L}} u_1 u_2 \equiv_{\mathsf{L}} s' \equiv_{\mathsf{L}} t$.

Summary

$\mathcal{C}$ compiles L into SKv and is fully compatible with term equivalence, evaluation and normality on closed terms s,t:

$$s \equiv_{\mathsf{L}} t \iff \mathcal{C}\, s \equiv_{\mathrm{SK}} \mathcal{C}\, t$$

$$s \Downarrow t \iff \mathcal{C}\, s \Downarrow \mathcal{C}\, t$$

$$s \text{ normal} \iff \mathcal{C}\, s \text{ normal}$$

So the whole semantic structure of L can be embedded in SKv and also be pulled back to L using $\mathcal{C}^{-1}$.

LC: L with closures

$$p, q, r \ ::= \ x \ \mid \ s[\sigma] \ \mid \ p{\cdot}q \qquad (x \in \mathbb{N}, s \in \mathsf{L}, \sigma : \mathbb{N} \to \mathsf{LC})$$

Intuition: Carry out substitution as deep as needed

$$\frac{}{x[\sigma] \succ_{\mathsf{LC}} \sigma(x)} \qquad \frac{}{(\lambda x.s)[\sigma]{\cdot}(\lambda y.t)[\tau] \succ_{\mathsf{LC}} s[x \mapsto (\lambda y.t)[\tau]\,, \sigma]\,\sigma}$$

$$\frac{}{st[\sigma] \succ_{\mathsf{LC}} s[\sigma]{\cdot}t[\sigma]} \qquad \frac{p \succ_{\mathsf{LC}} p'}{p{\cdot}q \succ_{\mathsf{LC}} p'{\cdot}q} \qquad \frac{q \succ_{\mathsf{LC}} q'}{p{\cdot}q \succ_{\mathsf{LC}} p{\cdot}q'}$$

- call-by-value $\Rightarrow$ uniformly confluent
- admissible terms: derivable from closed L-terms in empty context.
- $\lceil\cdot\rceil : L_C \to L$ substitutes using the environments
- Simulation Lemma: $\lceil p\rceil \Downarrow t \iff \exists q, p \Downarrow q \land t = \lceil q\rceil$.
- We also have a complete interpreter for LC.

Completeness on arbitrary terms

J. Roger Hindley and Jonathan P. Seldin.
Introduction to Combinators and λ–Calculus,
Cambridge University Press, 1986.

Yannick Forster
A Formal and Constructive Theory of Computation
Bachelor thesis, Saarland University, 2014

Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy.
Explicit substitutions
J. Funct. Program.,1(4):375–416, 1991

Jean-Jacques Lévy and Luc Maranget.
Explicit substitutionsand programming languages
19th FSTTCS, p. 181–200, 1999