

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

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

$$x_u^x := u$$

$$(st)_u^x := s_u^x t_u^x$$

$$y_u^x := y$$

$$(\lambda y.s)_u^x := \lambda y.(s_u^x)$$

$$\frac{s \gamma_L s'}{st \gamma_L s't}$$

$$\frac{s \gamma_L t'}{st \gamma_L st'}$$

$$\frac{}{(\lambda x.s)(\lambda y.t) \gamma_L s_{\lambda y.t}^x}$$

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

SK Combinatory Logic

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

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

- 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

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

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

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

- **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

- 1 Introduction
 - L: Weak Call-by-Value λ -Calculus
 - SK Combinatory Logic
- 2 SKv: Call-by-Value Combinatory Logic
 - Pseudo-Abstraction
- 3 Compiling L into SKv
 - Soundness
 - Left-Invertibility of Compilation
 - Completeness on normalizing terms
 - Completeness on arbitrary terms
- 4 LC: L with closures
- 5 Related Work

SKv: Call-by-Value Combinatory Logic

$$X, Y, Z ::= x \mid \mathbf{K} \mid \mathbf{S} \mid XY \quad (x \in \mathbb{N})$$
$$\text{Val} \ni X, Y ::= x \mid \mathbf{K} \mid \mathbf{K}X \mid \mathbf{S} \mid \mathbf{S}X \mid \mathbf{S}XY \quad (x \in \mathbb{N})$$

$$\frac{X \succ_{\text{SK}} X'}{XY \succ_{\text{SK}} X'Y}$$

$$\frac{Y \succ_{\text{SK}} Y'}{XY \succ_{\text{SK}} XY'}$$

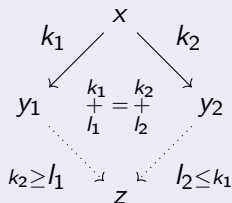
$$\frac{X, Y \in \text{Val}}{\mathbf{K}XY \succ_{\text{SK}} X}$$

$$\frac{X, Y, Z \in \text{Val}}{\mathbf{S}XYZ \succ_{\text{SK}} XZ(YZ)}$$

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

Uniform Confluence

Uniform Confluence



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

$$x_Z^x := u$$

$$y_Z^x := y$$

$$K_Z^x := K$$

$$S_Z^x := S$$

$$(XY)_Z^x := X_Z^x Y_Z^x$$

Substitutivity:

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

No similar lemma for values, e.g. $xK \notin Val$, but $(xK)_K^x = KK \in Val$.

X is a *maximal value* iff $X \in Val$, but $XY \notin Val$

So maximal values are values of form x , KX and SXY .

- If Y is a maximal value, then $X \in Val$ iff $X_Z^x \in Val$.

Pseudo-Abstraction

$$\begin{array}{ll} [x].x := \mathbf{I} & \\ [x].X := \mathbf{K}X & \text{if } x \notin \text{FV}(X) \wedge X \in \text{Val} \\ [x].(XY) := \mathbf{S}([x].X)([x].Y) & \text{otherwise} \end{array}$$

Similarities to L-abstractions:

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

Commutes with Substitution:

- Y maximal value $\wedge z \notin \text{FV}(Y) \wedge z \neq x \implies ([z].X)_Y^x = [z].(X_Y^x)$

Proof by Induction on X .

Crucial: $z \notin \text{FV}(X) \wedge X \in \text{Val} \iff z \notin \text{FV}(X_Y^x) \wedge X_Y^x \in \text{Val}$

Compiling L into SKv

$$\mathcal{C} x := x$$

$$\mathcal{C} (st) := (\mathcal{C} s)(\mathcal{C} t)$$

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

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 $\underline{s}_t^x = \underline{s}_t^x$

Soundness

If s is closed and $s \succ_L t$, then $\mathcal{C} s \succ_{SK}^+ \mathcal{C} t$.

Implies (for closed s):

$$s \succ_L^* t \Rightarrow \mathcal{C} s \succ_{SK}^* \mathcal{C} t \quad \text{and} \quad s \Downarrow t \Rightarrow \mathcal{C} s \Downarrow \mathcal{C} t$$

Left-Invertibility of Compilation

$$[x]^{-1}.(\mathbf{SKK}) := x$$

$$[x]^{-1}.(\mathbf{SXY}) := ([x]^{-1}.X)([x]^{-1}.Y)$$

$$[x]^{-1}.(\mathbf{KX}) := X$$

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

$$\mathcal{C}^{-1} x := x$$

$$\mathcal{C}^{-1} X := \lambda x.(\mathcal{C}^{-1} ([x]^{-1}.X)) \quad \text{if } X \in \text{Val}$$

$$\mathcal{C}^{-1} (XY) := (\mathcal{C}^{-1} X)(\mathcal{C}^{-1} Y)$$

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

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

Completeness

$\mathcal{C} s \gamma_{\text{SK}}^* \mathcal{C} t \implies s \gamma_{\text{L}}^* t$ does not hold:

Example

For a (reasonable) procedure u :

$(\lambda x.xx)u = \mathbf{SII}u \gamma_{\text{SK}} (\mathbf{I}u) (\mathbf{I}u) = ((\lambda x.x)u) ((\lambda x.x)u),$
but $(\lambda x.xx)u \not\gamma_{\text{L}}^* ((\lambda x.x)u) ((\lambda x.x)u)$

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

$$s \text{ closed} \wedge \mathcal{C} s \gamma_{\text{SK}} X \implies \exists t, X \gamma_{\text{SK}}^* \mathcal{C} t \wedge s \gamma_{\text{L}} t$$

Completeness on normalizing terms

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

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

Proof

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

- \underline{s} is SKv-redex
 $\implies s = (\lambda x. s')u$, where u procedure
 - ▶ successor of \underline{s} unique: X
 - ▶ $\underline{s} = \underline{(\lambda x. s')u} \succ_{\text{SK}}^+ \underline{(s'x)_u}$ $\implies t := s'x_u$ has claimed properties
- \underline{s} not SKv-redex
 $\implies s = s_1 s_2$, redex contained in $\underline{s_1}$ or $\underline{s_2}$
 \implies claim holds by inductive hypothesis

Completeness on normalizing terms(2)

$$s \text{ closed} \wedge \mathcal{C} s \succ_{\text{SK}} X \implies \exists t, X \succ_{\text{SK}}^* \mathcal{C} t \wedge s \succ_L t$$

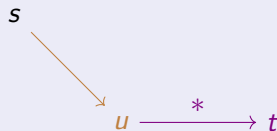
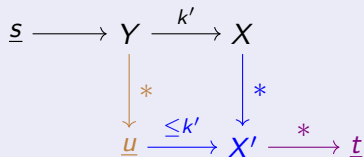
Generalizes to reduction chains:

$$s \text{ closed} \wedge \mathcal{C} s \succ_{\text{SK}}^* X \implies \exists t, X \succ_{\text{SK}}^* \mathcal{C} t \wedge s \succ_L^* t$$

Proof

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

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



Completeness on normalizing terms (3)

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

For normalizing, closed s :

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

Combined with soundness:

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

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

Completeness on arbitrary terms

We want: $\mathcal{C} s \succ_{\text{SK}}^* \mathcal{C} t \Rightarrow s \equiv_{\text{L}} t$.

We study the \mathcal{C} -image of (closed) β -redexes, depending on the body:

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

Proof

We have $(\lambda x.s)t \Downarrow s_t^x$.

We use completeness on normalizing terms and confluence of SKv.

For other bodies:

Assume a non-abstraction s and $(\lambda x.s)t \succ_{\text{SK}} Y$.

Then there is a closed s' with $\underline{s'} = Y$ and $(\lambda x.s)t \equiv_{\text{L}} s'$

Proof

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

Completeness on arbitrary terms (2)

$$s \text{ closed} \wedge C s \succ_{SK}^k C t \implies s \equiv_L t$$

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

$$X_i \succ_{SK}^{k_i} Y_i \vee X_i \succ_{SK}^k Z_i \wedge \underbrace{Z_1 Z_2}_{\text{redex}} \succ_{SK} Y \wedge Y \succ_{SK}^{k'} Y_1 Y_2$$

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

- $\underline{s_j} \succ_{SK}^{k_j} \underline{t_j}$ with $k_j \leq k$: inductive hypothesis $\Rightarrow s_j \equiv_L u_j$
- $\underline{s_j} \succ_{SK}^{k_j} \underline{u_j}$; $\underline{u_1} \underline{u_2} \succ_{SK} Y$; $Y \succ_{SK}^{k'} \underline{t}$; $k_1 + k_2 + 1 + k' = k$; u_j procedures.

By inductive hypothesis: $s_j \equiv_L u_j$.

- ▶ Body of u_1 abstraction: $u_1 u_2 \equiv_L t$. So $s_1 s_2 \equiv_L u_1 u_2 \equiv_L t$
- ▶ Body of u_1 non-abstraction: $\exists s'$ such that $Y = \underline{s'}$ and $u_1 u_2 \equiv_L s'$.
So $s_1 s_2 \equiv_L u_1 u_2 \equiv_L s' \equiv_L t$.

Summary

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

$$s \equiv_L t \iff \mathcal{C} s \equiv_{\text{SK}} \mathcal{C} t$$

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

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

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

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

Intuition: Carry out substitution as deep as needed

$$\frac{}{x[\sigma] \succ_{LC} \sigma(x)} \quad \frac{}{(\lambda x.s)[\sigma] \cdot (\lambda y.t)[\tau] \succ_{LC} s[x \mapsto (\lambda y.t)[\tau], \sigma] \sigma}$$
$$\frac{}{st[\sigma] \succ_{LC} s[\sigma] \cdot t[\sigma]} \quad \frac{p \succ_{LC} p'}{p \cdot q \succ_{LC} p' \cdot q} \quad \frac{q \succ_{LC} q'}{p \cdot q \succ_{LC} p \cdot q'}$$

- call-by-value \Rightarrow uniformly confluent
- admissible terms: derivable from closed L-terms in empty context.
- $[\cdot] : LC \rightarrow L$ substitutes using the environments
- Simulation Lemma: $[p] \Downarrow t \iff \exists q, p \Downarrow q \wedge t = [q]$.
- 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 substitutions and programming languages
19th FSTTCS, p. 181–200, 1999