Weak Call-By-Value Lambda-Calculus as a Model of Computation

Initial Bachelor Seminar Talk

Yannick Forster
Advisor: Prof. Dr. Gert Smolka

Saarland University
Programming Systems Lab

Saarland University
Computer Science

Idea

- A constructive development of basic Computability Theory.
- Self Interpreter using Scott-Encoding
- Proofs for the Theorems of Rice and Scott

Content

THE CALCULUS λ_{vw}

DEFINITIONS

De Brujn Terms:

$$s, t \ ::= n \mid s\ t \mid \lambda s \quad (n \in \mathbb{N})$$

Substitution:

$$\begin{aligned} n_u^k &= \text{if } n = k \text{ then } u \text{ else } n \\ (st)_u^k &= s_u^k t_u^k \\ (\lambda s)_u^k &= s_u^{Sk} \end{aligned}$$

Definitions

Closedness:

$$s \text{ closed} := \forall k\, u. s_u^k = s$$

Some definitions:

- "Combinator" := closed term
- "Procedure" := closed abstraction

REDUCTION

$$\frac{}{(\lambda s)(\lambda t) \succ s^0_{\lambda t}} \qquad \frac{s \succ s'}{st \succ s't} \qquad \frac{t \succ t'}{st \succ st'}$$

Define $\equiv$ as the reflexive, transitive, symmetric closure of $\succ$.

Important: $s \equiv s' \to t \equiv t' \to st \equiv s't'$

[Plotkin, 1975], [Niehren, 1996], [Dal Lago & Martini, 2008]

WHAT MAKES λ_{vw} GREAT

UNIFORM CONFLUENCE:

$$s \succ t_1 \to s \succ t_2 \to t_1 = t_2 \vee \exists t, t_1 \succ t \wedge t_2 \succ t$$

PARAMETRIC CONFLUENCE (holds in general for uniform confluent $\succ$):
If $s \succ^m t_1$ and $s \succ^n t_2$ then there are $k \leq n, l \leq m$ and u such that:

$$t_1 \succ^k u \ \wedge \ t_2 \succ^l u \ \wedge \ m + k = n + l$$

[Dal Lago & Martini, 2008]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Recursion combinator

There is R such that:

$$R\ s \equiv \lambda x.\ s^{0}_{R\ s}\ x$$

[Dal Lago & Martini, 2008]

Datatypes

Natural numbers

Scott encoding:

$$\bar{0} = \lambda\, z\, s.z$$
$$\bar{Sn} = \lambda\, z\, s.s\, \bar{n}$$

Addition

$$Succ := \lambda\, n\, z\, s.\; s\, n$$
$$Add := R(\lambda Anm.n\; m\; (\lambda n'.Succ\; (A\; n'\; m)))$$

then

$$Succ\; \bar{n} \equiv \bar{Sn}$$
$$Add\; \bar{0}\; \bar{m} \equiv \bar{m}$$
$$Add\; \overline{Sn}\; \bar{m} \equiv Succ\; \overline{n+m}$$

[Curry, Hindley, Seldin, 1972]

Scott encoding for terms

$$\ulcorner n \urcorner := \lambda\, v\, a\, l.\, v\, (\bar{n})$$

$$\ulcorner st \urcorner := \lambda\, v\, a\, l.\, a\, \ulcorner s \urcorner\, \ulcorner t \urcorner$$

$$\ulcorner \lambda s \urcorner := \lambda\, v\, a\, l.\, l\, \ulcorner s \urcorner$$

[Mogensen, 1990], [Stump, 2009]

Self Interpretation

What is needed

- Internalized equality of natural numbers
- Internalized substitution
- Step-indexed evaluation
- Encoding for Some/None
- Internalized step-indexed evaluation

Evaluation combinator

$$\text{eval}' \ \bar{n} \ \ulcorner s \urcorner = \text{case eva } \bar{n} \text{ s of Some s} \Rightarrow \text{s} \mid \text{None} \Rightarrow \text{eval}' \ \overline{Sn} \ \ulcorner s \urcorner$$

Lambda lifting needed

Important: eval′ $\bar{n}$ $\ulcorner s \urcorner$ converges iff. s converges

Decidability

Decidability

A set M is *decidable* if there is a procedure u (the decider) s.t.:

$$\forall s, s \in M \land u \ulcorner s \urcorner \equiv true \lor s \notin M \land u \ulcorner s \urcorner \equiv \text{false}$$

A set M is *semi-decidable* if there is a procedure u (the acceptor) s.t:

$$\forall s, s \in M \Longleftrightarrow u \ulcorner s \urcorner \text{ converges}$$

The acceptance set $\mathcal{A}u$ of a procedure u is defined as:

$$\{s$$

$\mid u \ulcorner s \urcorner$ converges$\}$
For $s \in \mathcal{A}u$ one could also write $\pi u s$.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Undecidable Problems

- $\{u \in \mathcal{P} \mid \neg\pi\ u\ u\}$
- $\{s \in \mathcal{P} \mid \forall t, \pi\ s\ t\}, \{s \in \mathcal{P} \mid \exists t, \pi\ s\ t\}, \{s \in \mathcal{P} \mid \forall t, \neg\pi\ s\ t\}$
- for $t \in \mathcal{C}$, $\{s \mid s \equiv t\}$
- $\{\ulcorner st \urcorner \mid s \equiv t\}$
- $\{s \in \mathcal{C} \mid s \text{ converges}\}$

Theorems

RICE'S THEOREM

If M is a set of procedures as follows:

- M is closed under $\mathcal{A}$-equivalence: If $s \in M$ and t is a procedure such that $\mathcal{A}t = \mathcal{A}s$, then $t \in M$.
- M is nontrivial: There is a procedure in M and there is a procedure not in M.

Then M is not decidable.

Follows directly from:
If M is a set of procedures as above:

- If $\lambda\Omega \in M$, then M is not semi-decidable
- If $\lambda\Omega \notin M$, then $\bar{M}$ is not semi-decidable

[Rice, 1953], [Barendregt, 1985]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Scott's Theorem

Second Fixpoint Theorem:
For every combinator s there exists a combinator t such that $s \ulcorner t \urcorner \equiv t$.

Scott's Theorem:
A set M of combinators is undecidable if it satisfies the following conditions:

- M is closed under reduction equivalence: If $s \in M$ and t is a combinator such that $t \equiv s$, then $t \in M$.
- M is nontrivial: There is a combinator in M and there is a combinator not in M.

SAARLAND UNIVERSITY
COMPUTER SCIENCE

[Barendregt, 1985]

Further Directions

- Formalize abstract programming systems and show that λ_{vw} yields a model
- Show the computational equivalence of λ_{vw} , call-by-value combinatory logic and Imp
- Show that computability in λ_{vw} implies computability in Coq. Is a constructive proof possible?

Scott Encoding

In a dataype with constructors $c_1, \ldots, c_n$ and a k-ary constructor c_i an element $c_i x_1 \ldots x_k$ is represented as

$$\lambda\, c_1 \ldots\; c_n.\, c_i\; x_1 \ldots x_k$$

Such a term yields a match construct:

$$
\begin{array}{l}
\texttt{match} \;\; t \;\; \texttt{with} \\
\quad \mid \; c_1 x_1 \ldots x_{k_1} \;\; \texttt{=>} \;\; f_1\; x_1 \ldots x_{k_1} \\
\quad \mid \; \ldots \\
\quad \mid \; c_n x_1 \ldots x_{k_n} \;\; \texttt{=>} \;\; f_n\; x_1 \ldots x_{k_n} \\
\texttt{end}
\end{array}
$$

is simply done with $t\, f_1\; \ldots\; f_n$

COMBINATORS I

Equality for Natural Numbers:

$$EqN\ \overline{0}\ \overline{0} \equiv true \qquad EqN\ \overline{Sm}\ \overline{0} \equiv false$$

$$EqN\ \overline{0}\ \overline{Sn} \equiv false \qquad EqN\ \overline{Sm}\ \overline{Sn} \equiv EqN\ \overline{m}\ \overline{n}$$

Substitution:

$$Subst\ \ulcorner n \urcorner\ \overline{k}\ \ulcorner u \urcorner \equiv EqN\ \overline{n}\ \overline{k}\ \ulcorner u \urcorner\ (Var\ \overline{n})$$

$$Subst\ \ulcorner st \urcorner\ \overline{k}\ \ulcorner u \urcorner \equiv App\ (Subst\ \ulcorner s \urcorner\ \overline{k}\ \ulcorner u \urcorner)\ (Subst\ \ulcorner t \urcorner\ \overline{k}\ \ulcorner u \urcorner)$$

$$Subst\ \ulcorner \lambda s \urcorner\ \overline{k}\ \ulcorner u \urcorner \equiv Lam\ (Subst\ \ulcorner s \urcorner\ (Succ\ \overline{k})\ \ulcorner u \urcorner)$$

Combinators II

Evaluation:

$$
\begin{aligned}
Eva\ \overline{k}\ \ulcorner n \urcorner &\equiv \ulcorner \bot \urcorner \\
Eva\ \overline{k}\ \ulcorner \lambda s \urcorner &\equiv Some\ (Lam\ \ulcorner s \urcorner) \\
Eva\ \overline{0}\ \ulcorner st \urcorner &\equiv \ulcorner \bot \urcorner \\
Eva\ \overline{Sn}\ \ulcorner st \urcorner &\equiv Eva\ \overline{n}\ \ulcorner s \urcorner \\
&\quad (\lambda x.\ Eva\ \overline{n}\ \ulcorner t \urcorner \\
&\qquad (\lambda y.\ x\ (\lambda \ulcorner \bot \urcorner) \\
&\qquad\qquad (\lambda\lambda \ulcorner \bot \urcorner) \\
&\qquad\qquad (\lambda z.\ Subst\ z\ \overline{0}\ y))) \\
&\qquad \ulcorner \bot \urcorner \\
&\quad \ulcorner \bot \urcorner
\end{aligned}
$$

SAARLAND UNIVERSITY
COMPUTER SCIENCE

COMBINATORS III

Encoding:

$$
\begin{aligned}
P\,\overline{0} &\equiv \ulcorner 0 \urcorner \\
P\,\overline{Sn} &\equiv Lam\,(Lam\,(App\,\ulcorner 0 \urcorner\,(P\,\overline{n}))) \\
Q\,\ulcorner n \urcorner &\equiv Lam\,(Lam\,(Lam\,(App\,\ulcorner 2 \urcorner\,(P\,\overline{n})))) \\
Q\,\ulcorner st \urcorner &\equiv Lam\,(Lam\,(Lam\,(App\,(App\,\ulcorner 1 \urcorner\,(Q\,\ulcorner s \urcorner))\,(Q\,\ulcorner t \urcorner)))) \\
Q\,\ulcorner \lambda s \urcorner &\equiv Lam\,(Lam\,(Lam\,(App\,\ulcorner 0 \urcorner\,(Q\,\ulcorner s \urcorner))))
\end{aligned}
$$

Second Fixpoint Theorem

$$A := \lambda z.\ s(App\, z\, (Q z))$$

$$t := A \ulcorner A \urcorner$$

$$t \succ s(App \ulcorner A \urcorner (Q \ulcorner A \urcorner)) \equiv s(App \ulcorner A \urcorner \ulcorner \ulcorner A \urcorner \urcorner) \equiv s \ulcorner A \ulcorner A \urcorner \urcorner \equiv s \ulcorner t \urcorner$$

SAARLAND UNIVERSITY

COMPUTER SCIENCE