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

Second 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.
- Verified self interpreter using Scott-encoding
- Proofs for the theorems of Rice and Scott

- Verified parallel executor (parallel or)
- Verified list library and internalized Goedel-numbering
- Well-known classical theorems, which are not provable intuitionistically

Review

DEFINITIONS

De Bruijn Terms:

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

Substitution:

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

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

REDUCTION

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 $\succ$.

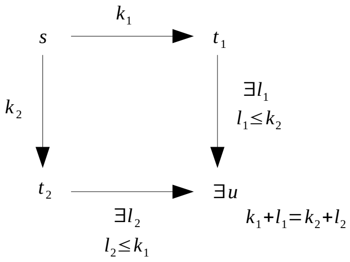
Important: $s \equiv s' \rightarrow \ t \equiv t' \rightarrow \ st \equiv s't'$

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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Last talk

- Uniform confluence
- Some undecidable problems
- Theorems of Rice and Scott
- Verified self interpreter



[Dal Lago & Martini, 2008]

Self-Interpreter

EVALUATION COMBINATOR

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

Lambda lifting needed
Important: $\text{Eval}'\ \bar{n}\ \ulcorner s \urcorner$ converges iff. s converges

How to prove this?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Define: $Eval := Eval' \; \overline{n}$
We need:

1. $s \Downarrow t \rightarrow \; Eval \ulcorner s \urcorner \equiv \ulcorner t \urcorner$
2. $Eval \ulcorner s \urcorner \equiv \ulcorner t \urcorner \rightarrow \; s \Downarrow t$
3. $Eval \ulcorner s \urcorner \Downarrow \rightarrow \exists v. Eval \ulcorner s \urcorner \equiv \ulcorner v \urcorner$
4. $s \Downarrow \Longleftrightarrow Eval \ulcorner s \urcorner \Downarrow$

Proofs:

1. Easy
2. ?
3. ?
4. Follows

$Eval\ \ulcorner s \urcorner \equiv \ulcorner t \urcorner \to\ s \Downarrow t$

Proof.
$Eval\ \ulcorner s \urcorner \equiv \ulcorner t \urcorner$, then $Eval'\ \overline{n}\ \ulcorner s \urcorner \succ^k \ulcorner t \urcorner$.
Complete induction over k.

- If $s \Downarrow^n t'$, then $Eval'\ \overline{n}\ \ulcorner s \urcorner \equiv \ulcorner t' \urcorner \equiv \ulcorner t \urcorner$ and thus $s \Downarrow t$
- If s does not converge in n steps, then
  $Eval'\ \overline{n}\ \ulcorner s \urcorner \succ^{k_1} Eval'\ \overline{Sn}\ \ulcorner s \urcorner \succ^{k_2} \ulcorner t \urcorner$
  where $k_1 > 0$ and $k = k_1 + k_2$.
  Inductive hypothesis yields result, because $k_2 < k$.

□

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Parallel-or

Parallel-Or

$$Por'\ \overline{n}\ \ulcorner s \urcorner\ \ulcorner t \urcorner$$

At recursion depth n:

- Execute s for n steps. Converges? Return true.
- Execute t for n steps. Converges? Return false.
- Start again at recursion depth $n + 1$.

$$Por := Por'\ \overline{0}$$

Correctness

Classically: One of the following holds:

1. s converges and $Por \ulcorner s \urcorner \ulcorner t \urcorner \equiv true$.
2. or t converges and $Por \ulcorner s \urcorner \ulcorner t \urcorner \equiv false$.
3. or the terms s, t, and $Por \ulcorner s \urcorner \ulcorner t \urcorner$ all diverge.

Intuitionistically:

1. If s converges or t converges, then $Por \ulcorner s \urcorner \ulcorner t \urcorner$ converges.
2. If $Por \ulcorner s \urcorner \ulcorner t \urcorner$ converges, then either $Por \ulcorner s \urcorner \ulcorner t \urcorner \equiv$ true and s converges or $Por \ulcorner s \urcorner \ulcorner t \urcorner \equiv$ false and t converges.

THEOREMS

Decidability

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

$$\forall s, s \in M \wedge u \ulcorner s \urcorner \equiv \mathit{true} \vee s \notin M \wedge u \ulcorner s \urcorner \equiv \text{false}$$

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

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

Short for u accepts s: $\pi\ u\ s$.

Post's Theorem

$$\forall M. acc\ M \rightarrow\ acc\ \overline{M} \rightarrow\ decidable\ M$$

Classically: Take acceptors u and v for M and $\overline{M}$, execute them on parallel on the input. One of them will converge.
Intuitionistically: $(\pi\ u\ s \Longleftrightarrow \neg\pi\ v\ s) \rightarrow\ \pi\ u\ s \vee \neg\pi\ v\ s$ does not hold.

[Bauer, 2006]

Some equivalent theorems

- *Post's Theorem*: $\forall M.acc\ M \rightarrow\ acc\ \overline{M} \rightarrow\ decidable\ M$
- *Convergence Theorem*: $\forall u \in \mathcal{C}.(\neg\neg u \Downarrow) \rightarrow\ u \Downarrow$
- *Markov's Rule for λ_{vw}*:
  $\forall M.decidable\ M \rightarrow\ (\neg\neg\exists t.t \in M) \rightarrow\ \exists t.t \in M$
- *Double Negation for Acceptable Sets*:
  $\forall M.acc\ M \rightarrow\ \forall t.(\neg t \notin M) \rightarrow\ t \in M$

[Bauer, 2006]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Existential quantification over decidable predicates is acceptable

For every natural number n, decide if the term s with Goedel number n fullfills the predicate.

- If yes: Halt
- If not: Proceed with $n + 1$

Goedel Numbering

Surjection $\mathbb{N}$ to $\mathbb{T}$ needed

Find duplicate-free lists T_n such that:

$$\frac{}{n \in T_n} \qquad \frac{s \in T_n \quad t \in T_n}{s\,t \in T_{n+1}} \qquad \frac{s \in T_n}{\lambda s \in T_{n+1}} \qquad T_{n+1} = T_n \mathbin{+\!\!+} B$$

Then: $\forall s \in \mathbb{T}. s \in T_{|s|}$ and $\forall n \in \mathbb{N}. |T_n| > n$

Goedel number of s: Position of s in $T_{|s|}$.
Inverse function: Element at position n in T_n

All in all under 100 lines for proofs.

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:

match t with
| $c_1 x_1 \ldots x_{k_1}$ => $f_1\; x_1 \ldots x_{k_1}$
| ...
| $c_n x_1 \ldots x_{k_n}$ => $f_n\; x_1 \ldots x_{k_n}$
end

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

[Curry, Hindley, Seldin, 1972]

Internalized List library

Internalized functions:

- *app*
- *map*
- Elementship
- *filter*
- Position

100 lines of proofs

CONCLUSION

- Great properties of λ_{vw} pay off
- Post's theorem does not hold intuitionistically
- Goedel numbering for arbitrary datatypes is easy
- Internalization is routine and probably automatable

- Decidablity and computability in λ_{vw} imply decidability and computability in Coq

Further Directions

- Formalize some enumeration theorems
  - Acceptable sets are recursively enumerable
  - Decidable sets are recursive
  - The set of acceptable predicates is recursively enumerable
- Self-interpreter for converging subset of $\mathbb{T}$ possible?
- Think about automated translation to λ_{vw}.

Bonus

$s \Downarrow \iff Eval \ulcorner s \urcorner \Downarrow$

Proof.
Left to right: Assume $s \Downarrow t$, then $Eval \ulcorner s \urcorner \equiv \ulcorner t \urcorner$.
Right to left: Assume $Eval \ulcorner s \urcorner \Downarrow$, then $\exists v. Eval \ulcorner s \urcorner \equiv \ulcorner v \urcorner$ and thus $s \Downarrow v$. $\square$

DECIDABILITY IN λ_{vw} IMPLIES DECIDABILITY IN COQ

$$(\forall s. u \ulcorner s \urcorner \equiv \text{true} \land Ps \lor u \ulcorner s \urcorner \equiv \text{false} \land \neg Ps) \to \forall s. dec(Ps)$$

Proof.
Want to show: $dec\ (Ps)$.
We know: $\exists n, u \ulcorner s \urcorner \Downarrow^n \text{true} \lor u \ulcorner s \urcorner \Downarrow^n \text{false}$ With constructive choice we get this n.
Then: Decide if $eva\ n\ u = true$ or $eva\ n\ u = false$. If the first, then Ps, else $\neg Ps$. □

Computability in λ_{vw} implies computability in Coq

$$(\forall m. \exists n. u\ \overline{m} \Downarrow \overline{n} \wedge p\ m\ n) \rightarrow \{ f \mid \forall m. p\ m\ (fm) \}$$

List Encoding

$$\ulcorner \ulcorner nil \urcorner \urcorner = \lambda nc.n$$

$$\ulcorner \ulcorner x :: xs \urcorner \urcorner = \lambda nc.c \ulcorner x \urcorner \ulcorner \ulcorner xs \urcorner \urcorner$$