Weak Call-by-Value Lambda Calculus as a Model of Computation in Coq

ITP 2017

Yannick Forster         Gert Smolka

Saarland University,
Programming Systems Lab
Related Work

- Michael Norrish
  *Mechanised computability theory*
  ITP 2011

- J. Xu, X. Zhang and C. Urban
  *Mechanising Turing Machines and computability theory in Isabelle/HOL*
  ITP 2013

- Andrea Asperti and Wilmer Ricciotti
  *A formalization of multi-tape Turing machines*
  TCS 2015

- Andrej Bauer
  *First steps in synthetic computability theory*
  ENTCS 2006
1.7. **Theorem** (Rice’s theorem)  
Suppose that $B \subseteq C_1$, and $B \neq \emptyset, C_1$. Then the problem ‘$\phi_x \in B$’ is undecidable.

**Proof.** From the algebra of decidability (theorem 2-4.7) we know that ‘$\phi_x \in B$’ is decidable iff ‘$\phi_x \in C_1 \setminus B$’ is decidable; so we may assume without any loss of generality that the function $f_\emptyset$ that is nowhere defined does not belong to $B$ (if not, prove the result for $C_1 \setminus B$).

**Kozen: Automata and Computability:**

*Proof of Rice’s theorem.* Let $P$ be a nontrivial property of the r.e. sets. Assume without loss of generality that $P(\emptyset) = \bot$ (the argument is symmetric if $P(\emptyset) = \top$). Since $P$ is nontrivial, there must exist an r.e. set $A$ such that $P(A) = \top$. Let $K$ be a TM accepting $A$.

**Wikipedia:**

Let us now assume that $P(a)$ is an algorithm that decides some non-trivial property of $F_a$. Without loss of generality we may assume that $P(no-halt) = \text{"no"}$, with no-halt being the representation of an algorithm that never halts. If this is not true, then this holds for the negation of the property. Since $P$ decides a non-trivial
INTRODUCTION

INGREDIENTS

- Take terms $s, t, u$, call closed normal forms \textit{procedures},
- Take evaluation $s \triangleright t$ (functional, $t$ procedure),
- Define $E_s := \exists t. s \triangleright t$,
- Take procedures $T \neq F$ such that $Tst \triangleright s$ and $Fst \triangleright t$,
- Take retraction $\tilde{s}$ into procedures to encode terms,
- Do computability theory.
DEFINITIONS

\[ u \text{ decides } p \text{ if } \]
\[ \forall s. \ ps \land u\overline{s} \triangleright T \lor \neg ps \land u\overline{s} \triangleright F \]

\[ u \text{ recognises } p \text{ if } \]
\[ \forall s. \ ps \leftrightarrow \mathcal{E}(u\overline{s}) \]
**u decides p if**

\[
\forall s. \; ps \land us \triangleright T \; \lor \; \neg ps \land us \triangleright F
\]

**Fact**

\[
\lambda s. \neg (ss \triangleright T) \text{ is not decidable.}
\]

**Proof.**

\[ u \text{ decides } \lambda s. \neg (ss \triangleright T): \]

\[
\forall s. \; \neg (ss \triangleright T) \land us \triangleright T \; \lor \; \neg \neg (ss \triangleright T) \land us \triangleright F
\]

\[
\neg (uu \triangleright T) \land uu \triangleright T \; \lor \; \neg \neg (uu \triangleright T) \land uu \triangleright F
\]

Contradiction!
**Selected Results**

- **Self-interpreter.** There is a procedure $U$ such that for all terms $s, t$:
  1. If $s \triangleright t$, then $U \overline{s} \triangleright \overline{t}$.
  2. If $U \overline{s}$ evaluates, then $s$ evaluates.

- **Rice’s theorem.** Every nontrivial extensional class of procedures is undecidable.

- **Modesty.** $L$-decidable classes are functionally decidable.

- **Post’s Theorem.** A class is decidable if it is recognisable, corecognisable, and logically decidable.
SYNTAX OF $L$

De Bruijn Terms:

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

\[
\begin{align*}
I &= \lambda x.x & T &= \lambda xy.x & F &= \lambda xy.y & \omega &= \lambda x.xx & D &= \lambda x.\omega \omega \\
 0 &= \lambda 0 & 1 &= \lambda(\lambda 1) & 0 &= \lambda(\lambda 0) & 00 &= \lambda(00) & \omega &= \lambda(\omega \omega)
\end{align*}
\]

“Procedure” := closed abstraction
SEMANTICS OF $L$

Reduction:

\[
(\lambda s)(\lambda t) \ Leadsto s_\lambda^0 \quad \frac{s \ Leadsto s'}{st \ Leadsto s't} \quad \frac{t \ Leadsto t'}{st \ Leadsto st'}
\]

implemented using capturing single-point substitution

\[\equiv\] equivalence closure of $\leadsto$

$\triangleright$ big-step evaluation to abstraction

1. Equational reasoning: $s \equiv s' \rightarrow t \equiv t' \rightarrow st \equiv s't'$
2. Church Rosser: If $s \equiv t$, then $s \Rightarrow^* u$ and $t \Rightarrow^* u$ for some $u$.
3. Unique nfs: If $s \triangleright^m t$, $s \triangleright^n u$, then $t = u$, $m = n$.

[Plotkin, 1975], [Niehren, 1996], [Dal Lago & Martini, 2008]
SCOTT ENCODINGS AND RECURSION

**ENCODINGS**

T, F for booleans  
\( \hat{n} \) for natural numbers  
\( \bar{s} \) for terms

**SCOTT CONSTRUCTORS**

- Succ \( \hat{n} \equiv \hat{S}n \)
- A \( \bar{s} \bar{t} \equiv \bar{st} \)

**RECURSION COMBINATOR**

- \((\rho u)v \equiv u(\rho u)v\)

[Mogensen, 1990], [Jansen, 2013]
**VERIFICATION**

Functional specification:

\[ \forall mn. \ Add \hat{m} \hat{n} \equiv \hat{m} + \hat{n} \]

By induction from:

\[ Add \ 0 \hat{n} \equiv \hat{n} \]
\[ Add \hat{S}m \hat{n} \equiv \text{Succ}(Add \hat{m} \hat{n}) \]

\[ Add := \rho(\lambda mn. mn(\lambda m_0. \text{Succ}(am_0n))) \]

\[ Add \hat{m} \hat{n} \equiv Add \hat{n} \hat{m} \]
If \( u \) decides \( p \) and \( v \) decides \( q \)
then \( \lambda s.ps \land qs \) is decidable.

\[
\lambda x.ux(vx)F \text{ does the job}
\]
(Step-indexed) Interpreter

\[ \text{eval} : \mathbb{N} \rightarrow \mathbb{T} \rightarrow \mathbb{T}_\perp \]

\[ \text{eval } n \ k = \perp \]
\[ \text{eval } n \ (\lambda s) = [\lambda s] \]
\[ \text{eval } 0 \ (st) = \perp \]
\[ \text{eval } (Sn) \ (st) = \text{match eval } n \ s, \text{ eval } n \ t \text{ with} \]
\[ | [\lambda s], \ [t] \Rightarrow \text{eval } n \ s^0_i \]
\[ | \_ \_ \Rightarrow \perp \]

\[ s \triangleright t \iff \exists n. \text{eval } n \ s = [t] \]

\[ \text{E} \overset{\triangleright}{n} \bar{s} \equiv \text{eval } n \ s \]

If \( s \triangleright t \), then \( U \bar{s} \triangleright \bar{t} \).

If \( U \bar{s} \) evaluates, then \( s \) evaluates.
MINIMISATION AND INTERPRETER

If $s \triangleright t$, then $U\bar{s} \triangleright \bar{t}$.
If $U\bar{s}$ evaluates, then $s$ evaluates.

Theorem

There is a procedure $C$ such that for every unary $u$:

1. If $u$ is satisfiable, then $Cu \triangleright \tilde{n}$ for some $n$ satisfying $u$.
2. If $Cu$ evaluates, then $u$ is satisfiable.

$U := \lambda x. E(C(\lambda y. E y x (\lambda z. T) F)) x$
RICE IN REALITY

Kozen:

Proof of Rice’s theorem. Let $P$ be a nontrivial property of the r.e. sets. Assume without loss of generality that $P(\emptyset) = \perp$ (the argument is symmetric if $P(\emptyset) = \top$). Since $P$ is nontrivial, there must exist an r.e. set $A$ such that $P(A) = \top$. Let $K$ be a TM accepting $A$.

Wikipedia:

Let us now assume that $P(a)$ is an algorithm that decides some non-trivial property of $F_a$. Without loss of generality we may assume that $P(\text{no-halt}) = "no"$, with no-halt being the representation of an algorithm that never halts. If this is not true, then this holds for the negation of the property. Since $P$ decides a non-trivial
**Rice & Scott**

*Scott:* Every class $p$ satisfying the following conditions is undecidable.

1. There are closed terms $s_1$ and $s_2$ such that $ps_1$ and $\neg ps_2$.
2. If $s$ and $t$ are closed terms such that $s \equiv t$ and $ps$, then $pt$.

*Rice:* Every class $p$ satisfying the following conditions is undecidable.

1. There are procedures $s_1$ and $s_2$ such that $ps_1$ and $\neg ps_2$.
2. If $s$ and $t$ are procedures such that $\forall uv. \overrightarrow{su} \triangleright v \iff \overrightarrow{tu} \triangleright v$ and $ps$, then $pt$. ("$p$ is extensional")

[Barendregt, 1984]
**Rice’s Theorem**

**Fact**

The class of closed terms $s$ such that $\neg E(s\bar{s})$ is not recognisable.

**Lemma (Reduction)**

A class $p$ is unrecognisable if there exists a function $f$ such that:

1. $p(fs) \leftrightarrow \neg E(s\bar{s})$ for every closed terms $s$.
2. There is a procedure $\nu$ such that $\nu\bar{s} \equiv \bar{fs}$ for all $s$. 
**RICE’S THEOREM**

**Lemma**

Let $p$ be an extensional class such that $D$ is in $p$ and some procedure $N$ is not in $p$. Then $p$ is unrecognisable.

**Proof.**

- Define function $fs$ such that
  - $fs \approx D$ if $\neg\mathcal{E}(ss)$
  - $fs \approx N$ if $\mathcal{E}(ss)$

- $f := s \mapsto \lambda y. F(ss)Ny$
- $\nu := \lambda x. L(A(A(A F(Ax(Qx)))N)0)$

- $\nu s \equiv \bar{fs}$ and $p(fs) \leftrightarrow \neg\mathcal{E}(ss)$

- Reduction lemma
Rice’s theorem

**Lemma**

Let $p$ be an extensional class such that $D$ is in $p$ and some procedure $N$ is not in $p$. Then $p$ is unrecognisable.

**Theorem**

Every nontrivial extensional class of procedures is undecidable.

**Proof.**

If $u$ decides $p$ then $pD$ or $\neg pD$ and …
**COMPUTABLE NORMAL FORMS**

Lemma

There is a function of type $\forall s. (\exists t. s > t) \rightarrow \sum t. s > t$.

Proof.

- $(\exists t. s > t) \iff \exists n. \text{eval } n \ s \neq \bot$
- $\lambda n. \text{eval } n \ s \neq \bot$ is Coq-decidable
- Use constructive choice (constructive indefinite ground description) to obtain $n$ with $\text{eval } n \ s = \lfloor t \rfloor$
- $s > t$
Typing total $\lambda$-definable functions in Coq

If $u$ decides $p$ then there is $f$ with $fs = \text{true} \iff ps$
$\Rightarrow L$-decidability implies Coq-decidability

$$\forall u. (\forall n \exists m. u^n \triangleright m) \rightarrow \{f : \mathbb{N} \rightarrow \mathbb{N} \mid \forall s. u^s \triangleright fs\}$$

[Larchey-Wendling (2017)]
**Post’s Theorem**

**Theorem**

If $u$ recognises $p$ and $v$ recognises $\lambda s. \neg ps$, then $p$ is decidable if $\forall s. ps \lor \neg ps$.

Without restriction: equivalent to $\neg \neg E_s \rightarrow E_s$

[Bauer (2006)]
FURTHER RESULTS

- **Totality.** The class of total procedures is unrecognisable.

- **Parallel or.** There is procedure O such that:
  1. If $s$ or $t$ evaluates, then $O \bar{s} \bar{t}$ evaluates.
  2. If $O \bar{s} \bar{t}$ evaluates, then either $O \bar{s} \bar{t} \triangleright T$ and $E s$, or $O \bar{s} \bar{t} \triangleright F$ and $E t$.

- **Closure under union.** The union of recognisable languages is recognisable.

- **Scott’s theorem.** Every nontrivial class of closed terms closed under $\equiv$ is undecidable.

- **Enumerability.** A class is recognisable if and only if it is enumerable.
CONTRIBUTION

- Elegant model of computation, easy to reason about
- Constructive formalisation of basic computability theory, less than 2000 loc
- Self-Interpreter, Rice, Scott, Post, Totality
**Future Work**

- “L and Turing Machines can simulate each other with a polynomially bounded overhead in time and a constant-factor overhead in space.”
  
  [Dal Lago, Martini (2008)], [Forster, Kunze, Roth (LOLA 2017)]

- Connect L to other models such as recursive functions.
- Use L to show “real-word” problems undecidable (e.g. from logic)
- Do further computability theory in L (Turing degrees, Myhill isomorphism theorem)
- Automate correctness proofs including time complexity
  
  [Forster, Kunze (CoqWS 2016)]

https://www.ps.uni-saarland.de/extras/L-computability/
<table>
<thead>
<tr>
<th>What?</th>
<th>Lines</th>
<th>cumulated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition of $L$</td>
<td>400</td>
<td>400 loc</td>
</tr>
<tr>
<td>Rice’s theorem</td>
<td>500</td>
<td>900 loc</td>
</tr>
<tr>
<td>Step-indexed interpreter</td>
<td>500</td>
<td>900 loc</td>
</tr>
<tr>
<td>Full parallel interpreter</td>
<td>300</td>
<td>1200 loc</td>
</tr>
<tr>
<td>Enumerable $\leftrightarrow$ recognisable</td>
<td>600</td>
<td>1500 loc</td>
</tr>
</tbody>
</table>