A Formal and Constructive Theory of Computation

Final Bachelor Seminar Talk

Yannick Forster
Advisor: Prof. Dr. Gert Smolka

Saarland University
Programming Systems Lab

Idea

- A constructive development of basic computability theory.
- Proofs for the theorems of Rice and Scott
- Verified universal program and dovetailing self-interpreter
- Equivalence of Acceptability and Recursive Enumerability

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Possibilities

- Turing Machines
- `WHILE` language
- a variant of λ-calculus

DEFINITIONS

SYNTAX OF L

De Bruijn Terms:

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

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

$$I := \lambda x.x \qquad \omega := \lambda x.x\ x$$

$$K := \lambda xy.x \qquad \Omega := \omega\ \omega$$

SEMANTICS OF L

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

WHAT MAKES L GREAT

UNIFORM CONFLUENCE:

$$s \succ t_1 \rightarrow \ s \succ t_2 \rightarrow \ t_1 = t_2 \lor \exists t, t_1 \succ t \land t_2 \succ t$$

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:

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

[Dal Lago & Martini, 2008]

BOOLEANS AND NATURAL NUMBERS

SCOTT ENCODING:

$$true := \lambda\, x\, y.x$$
$$false := \lambda\, x\, y.y$$

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

if b then s else t $\Rightarrow \overline{b}\, s\, t$

match n with O => s | S n′ => t $\Rightarrow \overline{n}\, s(\lambda n'.t)$

SAARLAND UNIVERSITY
COMPUTER SCIENCE

VERIFICATION

EXAMPLE: ADDITION

$$Succ := \lambda\, n\, z\, s.\, s\, n$$

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

$$Succ\, \overline{n} \equiv \overline{Sn}$$

$$Add\, \overline{0}\, \overline{n} \equiv \overline{n}$$

$$Add\, \overline{Sm}\, \overline{n} \equiv Succ\, \overline{m+n}$$

$$Add\, \overline{m}\, \overline{n} \equiv \overline{m+n}$$

[Curry, Hindley, Seldin, 1972]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

SCOTT ENCODING FOR TERMS

$$\ulcorner n \urcorner := \lambda\, v\, a\, l.\, v\, \overline{n}$$

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

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

[Mogensen, 1990], [Stump, 2009]

DECIDABILITY

A predicate $P : \mathbf{T} \to \mathbf{Prop}$ is *L-decidable* if there is a procedure u s.t.:

$$\forall s : \mathbf{T}, Ps \wedge u \ulcorner s \urcorner \equiv \mathit{true} \vee \neg Ps \wedge u \ulcorner s \urcorner \equiv \mathit{false}$$

A predicate P is *L-acceptable* if there is a procedure u s.t:

$$\forall s : \mathbf{T}, Ps \leftrightarrow u \ulcorner s \urcorner \text{ converges}$$

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

Some Undecidable Problems

- $\boldsymbol{\lambda}s.\ \mathbf{P}s \land \neg\pi\ s\ s$
- $\boldsymbol{\lambda}s.\ \mathbf{C}s \land s$ converges
- $\boldsymbol{\lambda}s.\ \mathbf{P}s \land \forall t, \pi\ s\ t$

Rice and Scott

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 predicate P is L-undecidable if it satisfies the following conditions:

1. P is only satisfied by combinators: $P \subseteq \mathbf{C}$.
2. P is closed under reduction equivalence: For equivalent combinators s and t it holds that $P\ s \ \rightarrow \ P\ t$.
3. P is nontrivial: There are combinators s_1 and s_2 with $P\ s_1$ and $\neg(P\ s_2)$.

[Barendregt, 1985]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

RICE'S THEOREM

Let P be a predicate such that:

1. P is only satisfied by procedures: $P \subseteq \mathbf{P}$.
2. P is extensional: If s_1 and s_2 are procedures such that $\forall t. \pi\ s_1\ t \leftrightarrow \pi\ s_2\ t$, then $P\ s_1 \rightarrow P\ s_2$.
3. P is nontrivial: There are procedures s_1 and s_2 with $P\ s_1$ and $\neg P\ s_2$.

Then we have the following:

1. If $P(\lambda\Omega)$, then P is not L-acceptable
2. If $\neg P(\lambda\Omega)$, then $\overline{P}$ is not L-acceptable

[Rice, 1953]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Verified Interpreters

What is needed

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

Evaluation combinator

Idea: Linear search over all evaluation depths

Eval′ $\overline{n}$ $\ulcorner s \urcorner$ = case Eva $\overline{n}$ $\ulcorner s \urcorner$ of Some s ⇒ s | None ⇒ Eval′ $\overline{Sn}$ $\ulcorner s \urcorner$

Important: Eval′ $\overline{n}$ $\ulcorner s \urcorner$ converges iff. s converges

How to prove this?

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 $n + 1$.

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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Correctness

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.

AD-Theorem

Theorem: A propositionally decidable predicate is decidable if it is acceptable and co-acceptable.

Proof: Let u and v be deciders for P and $\overline{P}$. Run them in parallel using *Por*, because P is propositionally decidable either u or v will converge.

Recursive Enumerability

Recursive Enumerability

A predicate $P : \mathbf{T} \to \mathbf{Prop}$ is called *L-enumerable* if there is a combinator F with:

1. $\forall n. F\,\overline{n} \equiv None \lor \exists s. F\,\overline{n} \equiv Some\,\ulcorner s \urcorner \land Ps$
2. $\forall s. Ps \to \exists n. F\,\overline{n} \equiv Some\,\ulcorner s \urcorner$

L-enumerability is equivalent to *L*-Acceptability

RE IMPLIES L-ACCEPTABILITY

Construct an acceptor u that for input s performs linear search over the domain of F converging iff s occured.

L-Acceptability implies RE

"Recursive enumeration of all halting Turing machines" by Jochen Burghardt - Own work.

Saarland University
Computer Science

L-Acceptability implies RE

- Enumerate all terms (surjection is sufficient)
- Execute every term for every index (surjection is sufficient)

Construct bijections: $\mathbb{N} \cong \mathbb{N} \times \mathbb{N} \cong \mathbf{T} \times \mathbb{N}$

Enough:

- $\mathbb{N} \rightarrow \mathbf{T}$
- $\mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$

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

Find duplicate-free lists T_n such that: $\forall s \in \mathbf{T}. s \in T_{|s|}$ and $\forall n \in \mathbb{N}. |T_n| > n$

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

Idea:

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

All in all under 100 lines for proofs.

CONCLUSION

CONCLUSION

- Elegant constructive formalization of computability theory possible
- Careful formulation of correctness criteria
- Intuitionistic reformulation of Rice's Theorem
- Intuitionistic refinement of AD-Theorem
- Compact proofs (< 1500 lines)

BONUS

L-DECIDABILITY IMPLIES DECIDABILITY IN COQ

Theorem:

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

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

Decidability in Coq does not imply *L*-decidability

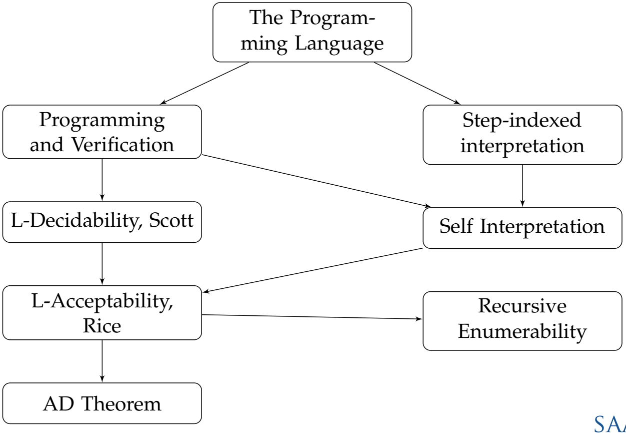
Theorem:

$$(\forall P, \mathit{dec}\ P \rightarrow\ \mathit{l\text{-}dec}\ P) \rightarrow\ \neg \mathit{SXM}$$

Proof:

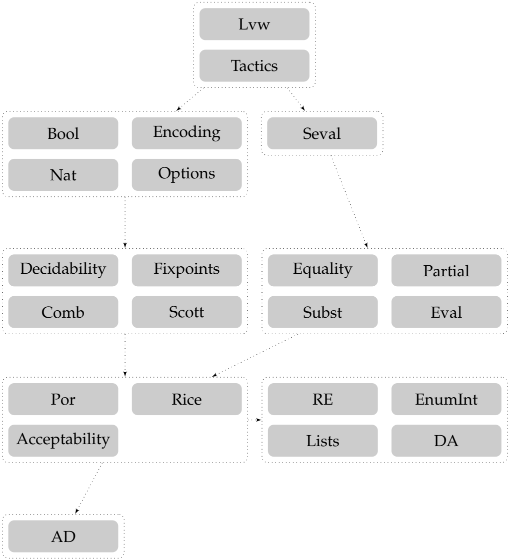
Assume $\forall P, \mathit{dec}\ P \rightarrow\ \mathit{l\text{-}dec}\ P$ and SXM. Then the self-halting problem would be L-decidable. Contradiction.

Overview



SAARLAND UNIVERSITY
COMPUTER SCIENCE

Coq-Development



SAARLAND UNIVERSITY
COMPUTER SCIENCE

Lines of proofs

File	Spec.	Proofs
Acceptability.v	12	50
AD.v	2	16
Bool.v	20	12
Lvw.v	186	119
Computability.v	30	16
DA.v	16	71
Decidability.v	14	18
Encoding.v	48	72
Equality.v	49	17
Eval.v	73	73
Fixpoints.v	4	20
EnumInt.v	64	51
Enum.v	51	113
Lists.v	84	97
MoreAcc.v	4	62
Nat.v	39	36
Options.v	15	29
Partial.v	84	11
Por.v	55	48
Proc.v	43	100
RE.v	21	114
Rice.v	65	92
Scott.v	10	54
Seval.v	43	79
Size.v	6	13
Subst.v	6	13
Tactics.v	46	12
MoreAcc.v	4	62
In Total	1094	1470

Correctness of *Por*

Classically: One of the following holds:

1. s converges and $Por\,\ulcorner s\urcorner\,\ulcorner t\urcorner \equiv \mathit{true}$.
2. or t converges and $Por\,\ulcorner s\urcorner\,\ulcorner t\urcorner \equiv \mathit{false}$.
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 $Por\,\ulcorner s\urcorner\,\ulcorner t\urcorner \equiv \mathit{true}$ and s converges or $Por\,\ulcorner s\urcorner\,\ulcorner t\urcorner \equiv \mathit{false}$ and t converges.

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]

SAARLAND UNIVERSITY
COMPUTER SCIENCE

List Encoding

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

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. $s \Downarrow\ \leftrightarrow Eval\,\ulcorner s \urcorner \Downarrow$

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 $k_2 < k$.

□

Internalized List library

Internalized functions:

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

100 lines of proofs