Program Equivalence is Coinductive

Dirk Pattinson    Lutz Schröder

The Australian National University
Friedrich-Alexander Universität Erlangen-Nürnberg

July 2016
Q. How do we understand programming languages?

A. PL = denotational + operational + axiomatic

Q. How do we understand models of computation? Turing / counter machines and the like?

- operational semantics – is a given
- denotational semantics – somewhat clear?
- axiomatic semantics – this paper
Example: Turing Machines

**Question.** What operations do we need to model Turing machines?
Operations for Turing Machines

**Tape Movement.** left (l), right (r)

**Tape Manipulation.** read (rd), write (wr)

**Arities** for a tape alphabet $A$
- left/right: unary
- writing: $A$ unary operations $wr_a$
- reading: one $A$-ary operation

$$rd(P_1, \ldots, P_n) \triangleq \text{case symbol under tape head of}$$

\[
\begin{align*}
    a_1 & \mapsto P_1 \\
    a_2 & \mapsto P_2 \\
    \vdots & \\
    a_n & \mapsto P_n
\end{align*}
\]
Equations for Tape Operations

Left/Right Cancel Out

\[ l \cdot r \cdot x = x \quad r \cdot l \cdot x = x \]

Pseudo-Branching has no Effect

\[ \text{rd}(x, \ldots, x) = x \]

Read/Write Interaction

\[ \text{wr}_{a_i} \cdot \text{rd}(x_1, \ldots, x_n) = \text{wr}_{a_i} \cdot x_i \]

Read/Write/Move Interaction

\[ \text{wr}_{a_i} \cdot m^k \cdot \text{rd}(x_1, \ldots, x_n) = m^k \cdot \text{rd}(m^{-k} \cdot \text{wr}_{a_i} \cdot m^k \cdot x_1, \ldots, m^{-k} \cdot \text{wr}_{a_i} \cdot m^k \cdot x_n) \]

where \( k \in \mathbb{Z} \setminus \{0\} \), \( m^+i = l^i \), \( m^-i = r^i \) for \( i \geq 0 \).
Counter Machines a la Minsky

**Operations.**

- increment (inc), unary
- clear (clr), unary
- jump/decrement (jd), binary

\[
\text{jd}(P_1, P_2) \equiv \begin{cases} P_1 & \text{if the counter is zero} \\ P_2 & \text{otherwise} \end{cases}
\]

**Equations.**

\[
\text{clr.jd}(x, y) = \text{clr}.x \quad \text{inc.jd}(x, y) = y
\]
Question. What’s the “right” interpretation of these algebraic theories?
Comodels for Turing Machines

Comodel Interpretation of $l, r, rd, wr_a$:

- carrier set $T = \text{two-sided infinite tape}$
- operations $l, r, wr_a : T \to T$
- reading: $rd : T \to T + \cdots + T$ (for each letter of the alphabet)

Informal Interpretation.

- $T = \text{tapes together with head position}$
- $l, r, wr_a$ manipulate the tape state
- $rd$ gives new state plus tape symbol

Concrete Example for alphabet $A = \{a_1, \ldots, a_n\}$ and $\alpha \in T$:

- $T = \mathbb{Z} \to A = \text{two-way infinite tape with head at position 0}$
- $l(\alpha)(n) = \alpha(n - 1)$ \hspace{1cm} $rd(\alpha) = \text{inj}_i(\alpha)$ where $\alpha(0) = a_i$

- $wr_a(\alpha)(n) = \begin{cases} a & n = 0 \\ \alpha(n) & n \neq 0 \end{cases}$
Comodels in Universal Algebraic Terms

**Defn.** Given signature $\Sigma = \text{function symbols} + \text{arities}$:

$\Sigma$-comodels are pairs $(A, \cdot)$ where

- $A$ is a (carrier) set
- $\cdot f : A \to n \cdot A$ for $n$-ary $f \in \Sigma$

**Term Interpretation** $[t] : A \to V \cdot A$ for a set $V$ of variables:

- $[v](c) = (v, c)$
- $[f(t_1, \ldots, t_n)] = [[t_1], \ldots [t_n]] \circ [f]$

- variables are coproduct injections
- terms interpreted “the other way around”

**Trivial Theorem.** The category of $\Sigma, E$-comodels is isomorphic to the category of comodels for the Lawvere theory generated by $\Sigma$ and $E$. 
Algebraic Machines

Computational Models.

\[(X, \xi : X \to T_\Sigma(X + 1))\]

- \(X\) is a (finite) state set
- \(\xi\) describes operational behaviour.
- \(* \in 1\) is termination.

**Operational Semantics.** given \(\Sigma\)-comodel \((C, \langle \cdot \rangle)\)

\[
(f(t_1, \ldots, t_n), c) \to (t_i, c') \iff \langle f \rangle(c) = \text{inj}_i(c')
\]

\[(x, c) \to (\xi(x), c)\]

**Examples.**

\[
(wr_a.t, c) \to (t, \langle wr_a \rangle(c))
\]

\[
(rd(t_1, t_2)) \to (t_i, c') \quad (\langle rd \rangle(c) = \text{inj}_i(c'))
\]

That is, \(t_i\) is chosen according to the symbol under the head, and \(c'\) is the post state after reading.
Comparision to Standard Model

**Trivial Theorem.** Standard Turing machines are equivalent to algebraic machines, for the comodel $\mathbb{Z} \to A$ of two-way infinite tapes. An analogous statement holds for counter machines.

**Why comodels?**
- Equational theory of computation
- Universal properties via final comodels
- Uniform treatment of more than one model
**Final Comodels**

**Observation.** For $E = \emptyset$, $\Sigma$-comodels are coalgebras for

$$FX = \prod_{f \in \Sigma} \text{arity}(f) \cdot X$$

and final comodels do exist.

**Lemma.** Final $\Sigma, E$-comodel exist, and are sub-comodels of the final $\Sigma$-comodel that satisfy all equations derivable from $E$.

**Lemma.** Final $\Sigma, E$-comodels are compact, with the subspace topology inherited from the final $\Sigma$-comodel.
Examples of Final Comodels

For Turing Machines over alphabet $A$.
- two-way infinite tapes, i.e. $\mathbb{Z} \to A$ are final.

For Counter machines
- Counters with infinity, i.e. $\mathbb{N} \cup \{\infty\}$ are final

(Both with respect to the “natural” equations.)
**Observation.** Equational Reasoning is incomplete over comodels. (E.g. nullary operations force comodels to be trivial.)

**Soundness is trivial.**
(And follows as $\Sigma, E$-comodels are comodels for a Lawvere theory.)
Completeness

**Linear Terms** are terms built from unary function symbols only.

**Example.**

\[ l.l.wr_a.l.wr_a.x \quad \text{inc.clr.inc.inc.x} \]

**Linear Normal Forms** for \( \Sigma \) and \( E \):

- every linear term is provably equal to a normal form term
- for normal form terms, syntactic identity and semantic equality are equivalent.

**Example.** Both Turing and counter machines have linear normal forms, e.g. \( \text{clr.inc...inc.x} \) and \( \text{inc....inc.x} \)

**Trivial Lemma.** Completeness for linear terms follows from the existence of linear normal forms.
General Terms and Splittings

**Splittings** for a signature $\Sigma$ and equations $\mathcal{E}$ are finite sets $S$ of linear terms such that

$$\{ s.t = s.u \mid s \in S \}$$

is an admissible rule.

A splitting $S$ is **reductive** for a general term $t$ if

$s.t$ is linear or has smaller branching than $c$

That is, we can successively reduce to linear terms.

**Example** for counter machines: $\{\text{inc}.x, \text{clr}.x\}$

**Equational Completeness** for a fixed comodel $C$ holds provided

- we have linear normal forms
- every term has a reductive splitting.
Turing machines and counter machines

Summary. Both Turing and counter machines have

- linear normal forms
- reductive splittings

over the final comodel.

Corollary. Equational reasoning is complete over the final comodel (under above assumptions).

Remark.

- also gives completeness for counter machines with standard counters
- but this non-final comodel fails compactness.
Reasoning about Computation

**Computational Simulation** for an algebraic machine 
\((X, \xi : X \to T_{\Sigma}(X + 1))\) and a comodel \(C\) and \(s, t \in T_{\Sigma}(X)\):

\[ t \leq s \iff \forall c, c' \in C : (t, c) \xrightarrow{*} (\ast, c') \implies (s, c) \xrightarrow{*} (c', c) \]

That is, \(s\) terminates whenever \(t\) does, with the same value.
Calculus for Computational Simulation

Equational Reasoning.

\[
(\mu) \frac{t \leq t}{t \leq t} \quad (\mu) \frac{E \vdash s = s' \quad s' \leq t'}{t \leq t'}
\]

Split

\[
(\mu) \frac{s.t \leq s.t' \mid s \in S}{t \leq t'} \quad (S \text{ reductive splitting})
\]

Unfolding.

\[
(\mu) \frac{s \leq t_{\xi}}{s \leq t} \quad (s \in T_{\Sigma}(\{\ast\})) \quad (\nu) \frac{s_{\xi} \leq t}{s \leq t} \quad (s \in T_{\Sigma}(X))
\]

Entailment. Inductive / Coinductive calculus

\[
E \vdash \leq = \nu R. \mu S.
\]

\[
\{s \leq t \mid s \leq t \text{ conc of } \mu\text{-rule with prems in } S\}
\]

\[
\{s \leq t \mid s \leq t \text{ conc of } \nu\text{-rule with prems in } R\}
\]
Soundness and Completeness

Assumptions.
- $C$ compact comodel and $E$ complete over $C$
- every $t \in T_\Sigma$ has reductive splitting
- splittings are full: every $c \in C$ is of the form $\langle s \rangle (c')$ for some $c' \in C$
- linear terms have normal forms

Soundness. We have $C \vDash s \leq t$ whenever $E \vdash s \leq t$
- follows from splittings being full and completeness of equational reasoning

Completeness. We have $E \vdash s \leq t$ whenever $C \vDash s \leq t$
- follows from reductiveness of splittings
- and compactness of $C$
Summary

Comodels for Turing machines and counter machines
- final comodels have complete equational axiomatisation
- proof uses reductive splittings

Computational Simulation for Turing and counter machines
- is sound and complete over final comodel
- proof uses reductive splittings
- compactness is crucial: finite unfolding on the right.
And thus the Turing Machine was born.