Program Equivalence is Coinductive

Dirk Pattinson
The Australian National University

Lutz Schröder *
Friedrich-Alexander-Universität Erlangen-Nürnberg

Abstract

We describe computational models, notably Turing and counter machines, as state transition systems with side effects. Side effects are expressed via an algebraic signature and interpreted over comodels for that signature: comodels describe the memory model while the transition system captures the control structure. Equational reasoning over comodels is known to be subtle. We identify a criterion on equational theories and classes of comodels that guarantees completeness, over the given class of comodels, of the standard equational calculus, and show that this criterion is satisfied in our leading examples. Based on a complete equational axiomatization of the memory (co)model, we then give a complete inductive-coinductive calculus for simulation between states, where a state simulates another if it has at least the same terminating computations, with the same cumulative effect on global state. Extensional equivalence of computations can then be expressed as mutual simulation. The crucial use of coinduction is to deal with non-termination of the simulated computation where the coinductive rule permits infinite unfolding.

Categories and Subject Descriptors F.1.1 [*Computation by Abstract Devices*]: Models of Computation—Bounded-action devices; F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs—Logics of programs

Keywords Models of computation, comodels, algebraic theories, non-wellfounded proofs, program equivalence

1. Introduction

Most state-based models of computation split the state into two components: a finite-state part associated with a control unit, and an infinite-state part thought of as modelling the memory. Turing machines, random access machines, and counter machines all follow this pattern, with the infinite memory modelled as one or more infinite tapes; as infinitely many registers each storing a natural number; and as a finite number of counters each storing a natural number, respectively. This paper provides a generic framework for reasoning about program equivalence in such computational models, and capitalizes on the distinction between control and memory. To our best knowledge, equational calculi for program equivalence have not previously been studied even for basic and widely used models such as Turing machines.

The first step in this program is to unify the various notions of memory model. We use the, by now fairly established [1, 13, 15, 16, 18, 17], device of modelling side effects as algebraic operations that are interpreted in so-called *comodels*. These are models C of algebraic theories in Set^{op}, the dual of the category of sets, and thus interpret n-ary algebraic operations as maps of the type

$$C \to n \cdot C$$

where $n \cdot C$ denotes the n-fold *co*-product of C. The intuition is that operations may change the state, i.e. produce a new state in C, and at the same time branch to one of n alternatives. E.g. the theory of Turing machines over a binary alphabet $\{0, 1\}$ will include a binary operation rd for reading from the current tape position and a unary operation l for moving the head to the left, and the term $\mathsf{rd}(\mathsf{l}(x), \mathsf{l}(\mathsf{l}(x)))$ will describe a computation that, depending on whether the current tape symbol is 0 or 1, will move the head one or two cells to the left, in both cases ending up in a yet unspecified control state currently named x. In contrast to standard universal algebra, the evaluation order of terms is from *left to right*, following the definition of term interpretation over comodels.

The equations of the theory then specify the behaviour of the memory model, prescribing, e.g., that reading from a cell that has just been written will produce the value written to the cell. A well-known problem with equational reasoning over comodels is that standard equational deduction in general fails to be complete over comodels. For example, the commutativity axiom $x + y = y + x$ is inconsistent over comodels, in the sense that it forces validity of all equations. In earlier work [14], we have given an extended equational deduction system incorporating non-standard rules that ensures completeness over comodels for unrestricted equational theories. Here, we complement this result with a criterion on equational theories and their intended classes of comodels that ensures completeness of the standard system, without additional rules. Roughly speaking, the criterion demands that the theory has enough write operations and is readily verified in our running examples, viz, Turing machines and counter machines.

Given an algebraic signature Σ describing the operations of the memory model, computational models, i.e. state machines that manipulate memory, then arise as coalgebras

$$X \to T_\Sigma(X + 1)$$

where $1 = \{\mathsf{halt}\}$ signifies termination, and $T_\Sigma(X + 1)$ is the set of Σ-terms over variables $X + 1$. We think of the states in X as labels, to each of which the coalgebra associates a command, possibly branching, that may cause a side effect and then branch to a new label or terminate successfully. The interpretation of Σ is given by a comodel C, which determines a transformation of $T_\Sigma(X+1)$ into the state monad for C, and hence an operational semantics for the language.

* Work forms part of the DFG project SCHR 1118/8-1 (HighMoon)

To reason about program equivalence formally, we define a system that axiomatizes *simulation* of control states, where a control state x simulates a control state y if whenever y terminates from some memory state then x does as well, with the same final memory state. Two states are equivalent iff they simulate each other. Our system is based on a complete equational axiomatization of the memory model developed in the first step, and incorporates inductive and coinductive rules, where inductive rules can only be applied in a well-founded manner while coinductive rules can be applied indefinitely, accounting for non-termination of the simulated program. We prove soundness and completeness of the deduction system, and obtain a generic complete coinductive verification system for program equivalence in state-based computational models.

Related Work Work on the coalgebraic Chomsky hierarchy [7] uses a similar type functor for coalgebras, geared towards automata instead of machines. Specifically, a T*-automaton* for a monad T is defined to be a coalgebra for $B \times T^A$ where B is the output alphabet and A the input alphabet. So far, work on T-automata is focussed on expressivity rather than on formal reasoning, as we pursue here. As far as we are aware, existing research on the equational logic of program equivalence has focussed mostly on abstracted formalisms rather than Turing complete computational models, such as Kleene algebra with tests [11] and the metalanguage of control and effects, a combination of Kleene algebra with Moggi's computational metalanguage [8]. On the side of functional computational models such as the λ-calculus, equational reasoning is more established, and often employs bisimulation-style methods (e.g. [2]).

Regarding research on comodels, we have already mentioned [18] where the theory of arrays is developed in terms of comodels, and the use of comodels in the semantics of programming languages [16, 17, 1]. None of these papers is concerned with axiomatic semantics, i.e. the equational logic of comodels. This has been addressed in [14] where we give a generic sound and complete calculus for equality over comodels for unrestricted algebraic theories. This calculus includes non-standard rules that are specific to comodels, while we use standard equational reasoning in this paper, and completeness hinges on additional properties of the algebraic theory. The model/comodel duality is investigated in [10, 9] on the basis of clones and establishes, in our terminology, a dual equivalence between categories of comodels and certain topological spaces, but does not investigate the logical aspects.

Non-wellfounded calculi have been used in [5, 4] to formalize arguments by infinite descent for inductive definitions where proofs are finite, possibly cyclic graphs subject to an external well-formedness condition. Our soundness proof can be cast as an instance of the generic soundness criterion (without becoming much shorter) while there appears to be no generic completeness result. In fact, such a result would necessarily need to admit general non-wellfounded proofs instead of finite cyclic proofs, as simple computability arguments show that our system becomes incomplete when restricted to finite cyclic proofs. Infinite equational deductions appear in the framework of *infinitary equational reasoning* [6], which focuses on equality of infinite computations with different objectives, and e.g. does not identify all non-terminating computations as we do. Some details of our soundness proof exhibit phenomena that bear a resemblence to step-indexed logical relations [3] (Remark 7.8).

2. Preliminaries and Notation

Universal Algebra. We use standard notions from universal algebra such as signatures, equations, and terms, and write $T_\Sigma(X)$ for the set of terms built from function symbols in Σ and variables in X. If $\mathcal{E}$ is a set of Σ-equations, we write $\mathcal{E} \vdash s = t$ if $s = t$ is derivable from $\mathcal{E}$ in equational logic. To make terms more readable, we use dot-notation and write $f.t$ instead of $f(t)$ if f is a unary function symbol, similarly for unary terms. The n*-fold copower* of A is $n \cdot A = A + \cdots + A$ (the n-fold coproduct), and in general $V \cdot A = \sum_{v \in V} A$ denotes the V-fold copower of an object A. We write $\mathsf{inj}_v : A \to V \cdot A$ for the v-th coproduct injection, $v \in V$. In the category of sets, copowers are isomorphic to cartesian products, i.e. $A \cdot B \cong A \times B$.

Comodels. A *comodel* for a signature Σ is a pair $(C, (\!|\cdot|\!))$ where C is a set (of *states*) and $(\!|f|\!) : C \to n \cdot C$ for n-ary $f \in \Sigma$. Intuitively, every n-ary function symbol is interpreted as an n-way case distrinction $C \to n \cdot C$ that may additionally change the state. A morphism of comodels $\phi : (C, (\!|\cdot|\!)_C) \to (D, (\!|\cdot|\!)_D)$ is a function $\phi : C \to D$ on the underlying sets that satisfies $[\phi, \dots, \phi] \circ (\!|f|\!)_C = (\!|f|\!)_D \circ \phi$ for all $f \in \Sigma$.

Given a comodel $(C, (\!|\cdot|\!))$, the *interpretation* $(\!|t|\!) : C \to X \cdot C$ of a term $t \in T_\Sigma(X)$ over C is defined inductively by

$$(\!|x|\!) = \mathsf{inj}_x$$
$$(\!|f(t_1, \dots, t_n)|\!) = [(\!|t_1|\!), \dots, (\!|t_n|\!)] \circ (\!|f|\!)$$

where $[\dots]$ denotes co-tupling. A comodel $(C, (\!|\cdot|\!))$ *satisfies* an equation $s = t$, in symbols $C \vDash s = t$, if $(\!|s|\!) = (\!|t|\!)$, and a comodel state $c \in C$ satisfies $s = t$, written $C, c \vDash s = t$, if $(\!|s|\!)(c) = (\!|t|\!)(c)$. We write $C \vDash \mathcal{E}$ if $C \vDash s = t$ for all equations $s = t \in \mathcal{E}$, and then call C a $\Sigma, \mathcal{E}$*-comodel*.

It is easy to see that comodels without equations, i.e. $\Sigma, \varnothing$-comodels, are in 1-1 correspondence with coalgebras for the functor $F_\Sigma X = \prod_{f \in \Sigma} \mathsf{ar}(f) \cdot X$; cf. Lemma 4.4. In the literature, comodels are usually introduced as models of a Lawvere theory in the category $\mathsf{Set}^{\mathsf{op}}$, the dual of the category of sets and maps. Here, we use comodels for a signature and equations, as this gives us concrete syntax on which we base the calculi that we will introduce later. Our notion of $\Sigma, \mathcal{E}$-comodel is straightforwardly equivalent with comodels for the Lawvere theory generated by Σ and $\mathcal{E}$. As a consequence, equational reasoning is sound over comodels:

Lemma 2.1. *Let C be a $\Sigma, \mathcal{E}$-comodel. Then $C \vDash s = t$ whenever $\mathcal{E} \vdash s = t$.*

On the other hand, as pointed out by Plotkin and Power [16], equational reasoning is *not* in general complete over comodels:

Remark 2.2. If Σ contains a nullary function symbol f and C is a Σ-comodel, then $C = \varnothing$ as $(\!|f|\!) : C \to 0 \cdot C = \varnothing$, hence $C \vDash s = t$ for all terms $s, t \in T_\Sigma(X)$ but of course not in general $\mathcal{E} \vdash s = t$ for all terms $s, t \in T_\Sigma(X)$. Disallowing nullary function symbols does not remedy this situation: every comodel C for a theory $\mathcal{E}$ with a commutative binary operator will similarly satisfy $C = \varnothing$, so that $\mathcal{E}$ semantically entails all equations over comodels. To see this, let $C, c \vDash f(x, y) = f(y, x)$ for distinct variables x, y, and w.l.o.g. let $(\!|f(x, y)|\!)(c) = \mathsf{inj}_x(c')$; then $(\!|f(y, x)|\!)(c) = \mathsf{inj}_y(c')$ and hence $\mathsf{inj}_x(c') = \mathsf{inj}_y(c')$, contradicting $x \neq y$.

One of the contributions of this paper is a criterion that guarantees completeness of equational logic interpreted over comodels.

3. Examples: Turing Machines and Counter Machines

Many models of computation comprise a notion of global state, such as tape content or counter values. We model global state as comodels for a signature that describes how it can be manipulated, constrained by equations. Our running examples are comodels for Turing and counter machines.

Definition 3.1. Fix an alphabet $\mathcal{A} = \{a_1, \dots, a_n\}$. The signature of Turing comodels with tape alphabet $\mathcal{A}$ is $\Sigma_\mathsf{T} = \{\mathsf{l}/1, \mathsf{r}/1, \mathsf{rd}/n\} \cup \{\mathsf{wr}_a/1 \mid a \in \mathcal{A}\}$ with arities as indicated. The operations l and r

move the head to the left/right, respectively, rd is an n-fold case distinction depending on the symbol currently under the head, and wr_a writes the symbol a onto the tape. The presentation of the equational theory of Turing machines makes use of the following derived operations. For $z \in \mathbb{Z}$, we write $\mathsf{m}^z \equiv \mathsf{r}^z$ if $z \geq 0$ and $\mathsf{m}^z \equiv \mathsf{l}^{-z}$ if $z \leq 0$ for the derived operation that moves the head z cells to the right (if z is positive) or $-z$ cells to the left (if z is negative). The operation $\mathsf{wr}_a^z \equiv \mathsf{m}^z.\mathsf{wr}_a.\mathsf{m}^{-z}$ writes a symbol $a \in \mathcal{A}$ to the cell that is z cells to the right of the head. Finally, $\mathsf{rd}^z(t_0, \dots, t_n) \equiv \mathsf{m}^z.\mathsf{rd}(\mathsf{m}^{-z}.t_0, \dots, \mathsf{m}^{-z}.t_n)$ branches according to the symbol z cells to the right of the head without changing the head position. The equational theory $\mathcal{E}_\mathsf{T}$ of Turing comodels is given by the following equations, where the mnemonics c and a stand for *commute* and *absorb*, respectively.

$$
\begin{aligned}
(\mathsf{m}/\mathsf{m}/\mathsf{a}) \quad & \mathsf{m}^y.\mathsf{m}^z.x = \mathsf{m}^{y+z}.x \\
(\mathsf{m}/\mathsf{wr}/\mathsf{c}) \quad & \mathsf{m}^z.\mathsf{wr}_a^y.x = \mathsf{wr}_a^{z+y}.\mathsf{m}^z.x \\
(\mathsf{m}/\mathsf{rd}/\mathsf{c}) \quad & \mathsf{m}^z.\mathsf{rd}^y(x_1, \dots, x_n) = \mathsf{rd}^{y+z}(\mathsf{m}^z.x_1, \dots, \mathsf{m}^z.x_n) \\
(\mathsf{wr}/\mathsf{wr}/\mathsf{c}) \quad & \mathsf{wr}_a^z.\mathsf{wr}_b^y.x = \mathsf{wr}_b^y.\mathsf{wr}_a^z.x \\
(\mathsf{wr}/\mathsf{wr}/\mathsf{a}) \quad & \mathsf{wr}_a^z.\mathsf{wr}_b^z.z = \mathsf{wr}_b^z.x \\
(\mathsf{wr}/\mathsf{rd}/\mathsf{a}) \quad & \mathsf{wr}_{a_i}^z.\mathsf{rd}^z(x_1, \dots, x_n) = \mathsf{wr}_{a_i}^z.x_i \\
(\mathsf{wr}/\mathsf{rd}/\mathsf{c}) \quad & \mathsf{wr}_a^z.\mathsf{rd}^y(x_1, \dots, x_n) = \mathsf{rd}^y(\mathsf{wr}_a^z.x_1, \dots, \mathsf{wr}_a^z.x_n) \\
(\mathsf{rd}/\mathsf{a}) \quad & \mathsf{rd}^z(x, \dots, x) = x \\
(\mathsf{rd}/\mathsf{wr}/\mathsf{a}) \quad & \mathsf{rd}^z(\mathsf{wr}_{a_1}^z.x_1, \dots, wr_{a_n}^z.x_n) = \mathsf{rd}^z(x_1, \dots, x_n)
\end{aligned}
$$

where $y \neq z$ are distinct in $(\mathsf{wr}/\mathsf{wr}/\mathsf{c})$ and $(\mathsf{wr}/\mathsf{rd}/\mathsf{c})$. We refer to $\Sigma_\mathsf{T}, \mathcal{E}_\mathsf{T}$-comodels as *Turing comodels*.

Using the signature Σ_T of Turing comodels, the transition function of a deterministic (standard) Turing machine with state set X can be expressed as coalgebra $\sigma : X \to T_\Sigma(X + 1)$ where $1 = \{\mathsf{halt}\}$ indicates termination and $\sigma(x)$ are tape operations to be performed in state $x \in X$.

The following is an example of a concretely given Turing comodel, in fact the final comodel.

Lemma 3.2. *Let $F = \mathbb{Z} \to \mathcal{A}$ be the set of functions from the integers to the (fixed) alphabet $\mathcal{A}$. We interpret $f \in F$ as a configuration of a two-way infinite Turing tape, with head at position 0 and tape content, z cells to the right of the head, given by $f(z)$. (We use z cells to the right synonymously with $-z$ cells to the left.) We equip F with a Σ_T-comodel structure by means of the following operations:*

$$
\begin{aligned}
(\!|\mathsf{l}|\!)(s)(n) &= s(n-1) & (\!|\mathsf{wr}_a|\!)(s)(0) &= a \\
(\!|\mathsf{r}|\!)(s)(n) &= s(n+1) & (\!|\mathsf{wr}_a|\!)(s)(m) &= s(m)
\end{aligned}
$$

$$
(\!|\mathsf{rd}|\!)(s) = \mathsf{inj}_i s
$$

where $s(0) = a_i$ in the last clause and $m \neq 0$. Then $(F, (\!|\cdot|\!))$ satisfies the equations $\mathcal{E}_\mathsf{T}$ so that $(F, (\!|\cdot|\!))$ is a Turing comodel. Moreover, $(F, (\!|\cdot|\!))$ is the final Turing comodel.

Proof. It is routine to check that $(F, (\!|\cdot|\!))$ satisfies all equations in $\mathcal{E}_\mathsf{T}$. We show that it is the final Turing comodel. Suppose that $(C, (\!|\cdot|\!)_C)$ is a Turing comodel. Consider the function

$$
\mathsf{hd} : C \to \Sigma \text{ given by } \mathsf{hd} = [k_0, \dots, k_{n-1}] \circ (\!|\mathsf{rd}|\!)_C
$$

where $k_i : C \to \Sigma$ is the constant function $k_i(x) = a_i$. Intuitively, hd computes the symbol under the head of a Turing machine. Define a function

$$
u : C \to F \text{ by } u(c)(z) = \mathsf{hd} \circ (\!|\mathsf{m}^z|\!)_C(c).
$$

It is easy to see that u is a morphism of comodels, and indeed the only one. □

Our second main example are counter machines in the style of Minsky [12], extended with operations that clear counters.

Definition 3.3. We assume a fixed number c of counters. The signature of counter machines is $\Sigma_\mathsf{C} = \bigcup\{\mathsf{clr}_n/1, \mathsf{inc}_n/1, \mathsf{jd}_n/2 \mid 1 \leq n \leq \mathsf{c}\}$ with arities as indicated. The operation clr_n clears the n-th counter by replacing its content with 0; inc_n increments the n-th counter; and jd_n branches to its first argument if the n-th counter is zero, and otherwise decrements the n-th counter and branches to its second argument. The algebraic theory $\mathcal{E}_\mathsf{C}$ of counter machines comprises the following equations, with mnemonics a, c as above.

$$
\begin{aligned}
(\mathsf{inc}/\mathsf{inc}/\mathsf{c}) \quad & \mathsf{inc}_n.\mathsf{inc}_m.x = \mathsf{inc}_m.\mathsf{inc}_n.x \\
(\mathsf{inc}/\mathsf{clr}/\mathsf{a}) \quad & \mathsf{inc}_n.\mathsf{clr}_n.x = \mathsf{clr}_n.x \\
(\mathsf{inc}/\mathsf{clr}/\mathsf{c}) \quad & \mathsf{inc}_n.\mathsf{clr}_m.x = \mathsf{clr}_m.\mathsf{inc}_n.x \qquad (m \neq n) \\
(\mathsf{inc}/\mathsf{jd}/\mathsf{a}) \quad & \mathsf{inc}_n.\mathsf{jd}_n(x, y) = y \\
(\mathsf{inc}/\mathsf{jd}/\mathsf{c}) \quad & \mathsf{inc}_n.\mathsf{jd}_m(x, y) = \mathsf{jd}_m(\mathsf{inc}_n.x, \mathsf{inc}_n.y) \qquad (n \neq m) \\
(\mathsf{clr}/\mathsf{clr}/\mathsf{c}) \quad & \mathsf{clr}_n.\mathsf{clr}_m.x = \mathsf{clr}_m.\mathsf{clr}_n.x \\
(\mathsf{clr}/\mathsf{clr}/\mathsf{a}) \quad & \mathsf{clr}_n.\mathsf{clr}_n.x = \mathsf{clr}_n.x \\
(\mathsf{clr}/\mathsf{jd}/\mathsf{a}) \quad & \mathsf{clr}_n.\mathsf{jd}_n(x, y) = \mathsf{clr}_n.x \\
(\mathsf{clr}/\mathsf{jd}/\mathsf{c}) \quad & \mathsf{clr}_n.\mathsf{jd}_m(x, y) = \mathsf{jd}_m(\mathsf{clr}_n.x, \mathsf{clr}_n.y) \qquad (n \neq m) \\
(\mathsf{jd}/\mathsf{clr}/\mathsf{a}) \quad & \mathsf{jd}_n(\mathsf{clr}_n.x, y) = \mathsf{jd}(x, y) \\
(\mathsf{jd}/\mathsf{clr}/\mathsf{inc}) \quad & \mathsf{jd}_n(\mathsf{clr}_n.x, \mathsf{inc}_n.x) = x
\end{aligned}
$$

and $1 \leq i \leq \mathsf{c}$. A $\Sigma_\mathsf{C}, \mathcal{E}_\mathsf{C}$-comodel is called a *counter comodel*.

As for Turing machines, we have a final counter comodel. It deviates slightly from the standard comodel in that it allows for infinite numbers to be stored in (uninitialized) counters; see Remark 7.16 on how verification over the standard comodel is encoded into verification over the final one.

Lemma 3.4. *Let $C = \{(n_i)_{1 \leq i \leq \mathsf{c}} \mid n_i \in \mathbf{N} \cup \{\infty\}\}$ be the set of c-tuples of extended natural numbers. We equip C with a Σ_C-comodel structure by means of the following operations where $\infty + 1 = \infty = \infty - 1$.*

$$
\begin{aligned}
(\!|\mathsf{clr}_i|\!)(s_1, \dots, s_\mathsf{c}) &= (s_1, \dots, s_{i-1}, 0, s_{i+1}, \dots s_\mathsf{c}) \\
(\!|\mathsf{inc}_i|\!)(s_1, \dots, s_\mathsf{c}) &= (s_1, \dots, s_{i-1}, s_i + 1, s_{i+1}, \dots, s_\mathsf{c}) \\
(\!|\mathsf{jd}_i(s_1, \dots, s_n)|\!) &= \mathsf{inj}_0(s_1, \dots, s_\mathsf{c}) && \text{if } s_i = 0 \\
(\!|\mathsf{jd}_i(s_1, \dots, s_\mathsf{c})|\!) &= \mathsf{inj}_1(s_1, \dots, s_{i-1}, s_i - 1, \\
& && s_{i+1}, \dots, s_\mathsf{c}) \qquad \text{if } s_i > 0
\end{aligned}
$$

Then $(C, (\!|\cdot|\!))$ is a counter comodel, that is, satisfies all equations in $\mathcal{E}_\mathsf{C}$, and is in fact the final counter comodel.

Proof. It is easy to see that C satisfies all equations in $\mathcal{E}_\mathsf{C}$. To see that C is final, let $(D, (\!|\cdot|\!)_D)$ be a counter comodel. Consider the constant functions $n, z : D \to \{0, *\}$ given by $n(\cdot) = *$ and $z(\cdot) = 0$ so that $[z, n] : D + D \to \{0, *\}$ maps elements of the left component of $D + D$ to 0 and elements of the right to $*$. For $1 \leq i \leq \mathsf{c}$ let

$$
f_0^i = [z, n] \circ (\!|\mathsf{jd}_i|\!)_D \qquad f_{n+1}^i = [z, f_n^i] \circ (\!|\mathsf{jd}_i|\!)_D
$$

so that $f_0^i(d) = 0$ if and only if $(\!|\mathsf{jd}_i|\!)(d)$ delivers a value in the left component of $D + D$ that we interpret as the i-th counter of D having reached zero. Iterating jd_i, we have $f_n^i = 0$ if the i-th counter decrements to 0 in n or fewer steps. This allows us to define

$$
u_i : D \to \mathbf{N} \cup \{\infty\}, u_i(d) = \min\{n \in \mathbf{N} \mid f_n^i(d) = 0\}
$$

where $\min \varnothing = \infty$. Combining the u_i gives a comodel morphism

$$
u = \langle u_1, \dots, u_\mathsf{c} \rangle : D \to C
$$

and one shows that this is indeed the only possible one. □

As for Turing machines, we can express any counter machine in the standard sense as a coalgebra $X \to T_{\Sigma_\mathsf{C}}(X+1)$ for a finite set X of states: a counter machine program $0 : i_0; 1 : i_1; \dots n : \imath_n$ with line numbers $0, \dots, n$ and instruction $i_j \in \{\mathsf{inc}, \mathsf{clr}, \mathsf{jd}, \mathsf{halt}\}$ is the evident transition system $\{0, \dots, n\} \to T_{\Sigma_\mathsf{C}}(\{0, \dots, n\} \cup \{\mathsf{halt}\})$ so that general counter machines are coalgebras $X \to T_{\Sigma_\mathsf{C}}(X+1)$.

The main goal of our paper is to define the operational semantics of machine models that arise in this form with the aid of comodels, and then give a sound and complete logical description of extensional equality of computations.

4. Final Comodels and Their Topology

Both algebraic theories presented in the previous section did permit a final comodel. It is known that this holds in general [17], and we show that in addition, this final comodel is topologically compact, and its operations are continuous functions. Compactness will be crucial for completeness of the calculus for comodel-based machines, as it provides a semantic handle on termination. For the whole section, we fix a finite signature Σ and a set $\mathcal{E}$ of Σ-equations.

We first note that validity of equations can be enforced by restricting to those comodel states that satify all derivable equations.

Lemma 4.1. *Let C be a Σ-comodel and let $C_\mathcal{E} = \{c \in C \mid C, c \vDash s = t \text{ for all } s, t \text{ with } \mathcal{E} \vdash s = t\}$. Then $C_\mathcal{E}$ is a sub-comodel of C and a $\Sigma, \mathcal{E}$-comodel.*

Proof. We show that $C_\mathcal{E}$ is a sub-comodel. Let $c \in C_\mathcal{E}$ and $f \in \Sigma$ with $(\!|f|\!)_C(c) = \mathsf{inj}_i(c')$. We need to show that $c' \in C_\mathcal{E}$. So let $\mathcal{E} \vdash s = t$. By congruence, $\mathcal{E} \vdash f(s, \dots, s) = f(t, \dots, t)$ so that $c \vDash f(s, \dots, s) = f(t, \dots, t)$. Then $(\!|s|\!)(c') = [(\!|s|\!), \dots, (\!|s|\!)] \circ \mathsf{inj}_i(c') = [(\!|s|\!), \dots, (\!|s|\!)] \circ (\!|f|\!)(c) = (\!|f(s, \dots, s)|\!)(c) = (\!|f(t, \dots, t)|\!)(c) = [(\!|t|\!), \dots, (\!|t|\!)] \circ (\!|f|\!)(c) = [(\!|t|\!), \dots, (\!|t|\!)] \circ \mathsf{inj}_i(c') = (\!|t|\!)(c')$ so that $c' \vDash s = t$. It is clear by construction that $C_\mathcal{E}$ is a $\Sigma, \mathcal{E}$-comodel. □

Moreover, the image of a comodel morphism is always a sub-comodel.

Lemma 4.2. *Let $f : C \to D$ be a morphism of Σ-comodels, with C a $\Sigma, \mathcal{E}$-comodel. Then $\mathfrak{I}(f) = \{f(c) \mid c \in C\}$ is a $\Sigma, \mathcal{E}$-comodel.*

We now show that $\Sigma, \mathcal{E}$-comodels always exist, and can be carved out of a suitable final coalgebra. This final coalgebra always exists in our setup:

Lemma 4.3. *[19] Let the category $\mathbf{C}$ have and the functor $F : \mathbf{C} \to \mathbf{C}$ preserve inverse limits of ω-cochains. Then the final F-coalgebra is (Z, ζ) where Z is the limit of the sequence $(F^n 1)_{n \in \mathbf{N}}$ in Set with connecting maps $f_0 = ! : F1 \to 1$ and $f^{n+1} = Ff_n$, and ζ is the unique map satisfying $F\pi_n\zeta = \pi_{n+1}$ where $\pi_n : Z \to F^n 1$ are the limit projections.*

As indicated above, we obtain from Σ a functor F_Σ on Set by $F_\Sigma X = \prod_{f \in \Sigma} \mathsf{ar}(f) \cdot X$; we denote the projections $F_\Sigma X \to \mathsf{ar}(f) \cdot X$ by π_f. In the same way, we define the functor $\bar{F}_\Sigma$ on the category Top of topological spaces and continuous maps. We record formally that F_Σ-coalgebras are essentially Σ-comodels:

Lemma 4.4. *The categories of F_Σ-coalgebras and Σ-comodels are isomorphic via the functor that sends an F_Σ-coalgebra (C, γ) to the comodel on C with structure given by $(\!|f|\!) = \pi_f \circ \gamma$.*

Both F_Σ and $\bar{F}_\Sigma$ are isomorphic to products of a constant functor (for $\prod_{f \in \Sigma} \mathsf{ar}(f)$, equipped with the discrete topology in the case of $\bar{F}_\Sigma$) and a power functor $(\cdot)^k$ (where $k = |\Sigma|$), and hence preserve all limits. The final coalgebras of F_Σ and $\bar{F}_\Sigma$ are therefore described by Lemma 4.3. Note that $\bar{F}^n_\Sigma 1$ is just $F^n_\Sigma 1$ equipped with the discrete topology. Moreover, the forgetful functor $\mathsf{Top} \to \mathsf{Set}$ preserves all limits, so if (Z, ζ) is the final F_Σ-coalgebra then the underlying space $\bar{Z}$ of the final $\bar{F}_\Sigma$-coalgebra has the form $\bar{Z} = (Z, \tau)$ where τ is the coarsest topology making all projections $\pi_n : Z \to F^n 1$ continuous. Moreover, by the description of the final coalgebra structure in Lemma 4.3, ζ is also the structure map of the final $\bar{F}_\Sigma$-coalgebra. Explicitly:

Lemma 4.5. *1. The topology of $\bar{Z}$ is generated by the subbase $\{\pi_n^{-1}(o) \mid o \subseteq F^n 1, n \in \mathbf{N}\}$.*
2. The map $\zeta : \bar{Z} \to \bar{F}_\Sigma \bar{Z}$ is continuous.
3. For every term s, the interpretation $(\!|s|\!)$ in $\bar{Z}$ is continuous.
4. $\bar{Z}$ is a Stone space.

Proof. The first item is immediate from the description of τ as the coarsest topology on Z making the π_n continuous. The second has been argued above, and implies continuity of $(\!|f|\!)$ for $f \in \Sigma$ by Lemma 4.4 and the fact that the projections π_f are continuous. The third item then follows by induction on s. The fourth follows from the fact that (by Tychonoff's theorem and preservation of compactness and zero-dimensionality under taking closed subspaces) the category of Stone spaces is closed under limits in Top. □

Lemma 4.6. *$Z_\mathcal{E} = \{z \in Z \mid Z, z \vDash s = t \text{ for all } s, t \text{ with } \mathcal{E} \vdash s = t\}$ carries the structure of a $\Sigma, \mathcal{E}$-comodel, and is indeed the final $\Sigma, \mathcal{E}$-comodel.*

Proof. The fact that $Z_\mathcal{E}$ is a sub-comodel of Z has been established in Lemma 4.1. We need to show that it is final. Let C be a $\Sigma, \mathcal{E}$-comodel, and let $u : C \to Z$ be the unique morphism of Σ-comodels induced by finality of Z. Then $\mathfrak{I}(u) \subseteq Z$ is a sub-comodel of Z and a $\Sigma, \mathcal{E}$-comodel by Lemma 4.2. This gives a morphism $C \to Z_\mathcal{E}$ by construction of $Z_\mathcal{E}$. Now assume that we have two such morphisms $u, v : C \to Z_\mathcal{E}$. By extending them to morphisms with codomain Z, we have two morphisms into the final Σ-comodel Z so that $u = v$. □

Corollary 4.7. *The final $\Sigma, \mathcal{E}$-comodel $Z_\mathcal{E}$ carries the structure of a Stone space.*

Proof. By Lemma 4.6 and Lemma 4.5, $Z_\mathcal{E}$ is a subspace of the Stone space $\bar{Z}$ determined by a set of equations between continuous functions; since Z is Hausdorff, such a subspace is closed, and therefore again a Stone space. □

5. Equational Completeness over Comodels

The completeness of the announced characterization of equality of computable functions has two main parts: a static component that amounts to soundness and completeness of the equational theory of a comodel, and a dynamic component to deal with recursion. In this section, we develop the static component and present a general criterion for the completeness of equational logic over comodels. In particular, this criterion applies to the equational axiomatization of Turing and counter comodels, introduced in Definitions 3.1 and Definition 3.3, respectively. The proof proceeds in several steps, and we begin with a stratification of terms according to the amount of branching (function symbols of arity > 1).

Definition 5.1. Let Σ be a signature. The set $T^n_\Sigma(V)$ of *degree-n terms* over Σ and a set V of variables is given by $T^{-1}_\Sigma(V) = \varnothing$ and

$$T^n_\Sigma(V) \ni t ::= x \mid f.t \mid g(u_1, \dots, u_k) \qquad (n \geq 0)$$

where $x \in V$, $f \in \Sigma$ is unary and $t \in T^n_\Sigma(V)$, and $u_1, \dots, u_n \in T^{n-1}_\Sigma(V)$, and $g \in \Sigma$. Terms in $T^0_\Sigma(V)$ are called *linear Σ-terms*.

If C is a class of comodels for Σ and a set $\mathcal{E}$ of Σ-equations, we say that *linear terms have normal forms* over $\mathcal{E}$ and C if there exists a set $\mathsf{Nf}(V) \subseteq T^0_\Sigma(V)$ (of *normal forms*) such that

1. For all $s \in T^0_\Sigma(V)$ there exists $t \in \mathsf{Nf}(V)$ such that $\mathcal{E} \vdash s = t$
2. For all $s, t \in \mathsf{Nf}(V)$ we have $\mathsf{C} \vDash s = t$ iff $s \equiv t$ (where $\equiv$ denotes syntactic equality).

That is, the nesting depth of function symbols of arith >1 (that we think of as branching points in a program) in a degree-n term is at most n (and linear terms don't contain any function symbols of arity >1). Linear terms have normal forms if 1) every linear term is provably equal to a normal form, and 2) normal forms are semantically equivalent only if they are syntactically identical.

It is evident that the above requirements entail equational completeness for linear terms, and that every term has a degree.

Lemma 5.2. *Let $\mathcal{E}$ be a set of equations over a signature Σ, and let C be a class of $\Sigma, \mathcal{E}$-comodels such that linear terms have normal forms over $\mathcal{E}$ and C. Then $\mathsf{C} \vDash s = t \iff \mathcal{E} \vdash s = t$ for all $s, t \in T^0_\Sigma(V)$.*

Lemma 5.3. *Let Σ be an algebraic signature. Then $T_\Sigma(V) = \bigcup\{T^n_\Sigma(V) \mid n \in \mathbf{N}\}$.*

Crucially, the equational axiomatization of both Turing and counter comodels permits normal forms. We begin with Turing comodels:

Lemma 5.4. *Linear Σ_T-terms have normal forms over $\mathcal{E}_\mathsf{T}$ and the (one-element) model class $\{F\}$ consisting of the final Turing comodel.*

Proof. Given a set V of variables, the set $\mathsf{Nf}(V)$ consists of all terms of the form

$$\mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_i}_{a_i} \cdot \mathsf{m}^j \cdot x$$

where $i \geq 0$, $n_1 < \cdots < n_i$, and $j \in \mathbb{Z}$. It is immediate that the second requirement for normal form terms holds. We now show that every linear term s is provably equal to a (linear) term in normal form, by induction on the term structure. If s is a variable, then s is in normal form already. If $s = \mathsf{l}.s'$ we have a normal form term $t' = \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_i}_{a_i} \cdot \mathsf{m}^j \cdot x$ such that $E_l \vdash s' = t'$. Then

$$\begin{aligned} \mathsf{l}.s' &= \mathsf{wr}^{n_1+1}_{a_1} \ldots . \mathsf{wr}^{n_i+1}_{a_n} . \mathsf{l}.\mathsf{m}^j \cdot x && \text{(using } (\mathsf{m}/\mathsf{wr}/\mathsf{c})) \\ &= \mathsf{wr}^{n_1+1}_{a_1} \ldots . \mathsf{wr}^{n_i+1}_{a_n} . \mathsf{m}^{j-1} \cdot x && \text{(using } (\mathsf{m}/\mathsf{m}/\mathsf{a})) \end{aligned}$$

which is in normal form. The case for $s = \mathsf{r}.s'$ is similar. Now suppose that $s = \mathsf{wr}^n_a.s'$ and let $t' = \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_i}_{a_i} \cdot \mathsf{m}^j \cdot x$ be a normal form term such that $E_l \vdash s' = t'$. First suppose that $n_1 < \cdots < n_\alpha < n < n_\beta < \cdots < n_i$. Then

$$\begin{aligned} \mathsf{wr}^n_a.s' &= wr^n_a . \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_i}_{a_i} \cdot \mathsf{m}^j \cdot x \\ &= \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_\alpha}_{a_\alpha} . \mathsf{wr}^n_a . \mathsf{wr}^{n_\beta}_{a_\beta} \ldots . \mathsf{wr}^{n_i}_{a_i} . \mathsf{m}^j \cdot x \end{aligned}$$

which is in normal form, using $(\mathsf{wr}/\mathsf{wr}/\mathsf{c})$ in the last step. Now suppose that $n_1 < \cdots < n = n_\alpha < n_\beta < \cdots < n_i$. Then

$$\begin{aligned} \mathsf{wr}^n_a.s' &= wr^n_a . \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^{n_i}_{a_i} \cdot \mathsf{m}^j \cdot x \\ &= \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^n_a . \mathsf{wr}^{n_\alpha}_{a_\alpha} . wr^{n_\beta}_{a\beta} \ldots . \mathsf{wr}^{n_i}_{a_i} . \mathsf{m}^j \cdot x \\ &= \mathsf{wr}^{n_1}_{a_1} \ldots . \mathsf{wr}^n_a . \mathsf{wr}^{n_\beta}_{a_\beta} \ldots . \mathsf{wr}^{n_i}_{a_i} . \mathsf{m}^j \cdot x \end{aligned}$$

which is in normal form, using $(\mathsf{wr}/\mathsf{wr}/\mathsf{c})$ in the second and $(\mathsf{wr}/\mathsf{wr}/\mathsf{a})$ in the third step. □

The same requirement is satisfied for counter comodels:

Lemma 5.5. *The equational axiomatization $\mathcal{E}_\mathsf{C}$ has linear normal forms over the (one-element) model class $\{C\}$ consisting of the final counter comodel.*

Proof. Given a set V of variables, we take the set $\mathsf{Nf}(V)$ to consist of all terms of the form

$$\mathsf{clr}_{i_1} \ldots . \mathsf{clr} i_n . \mathsf{inc}^{k_1}_{j_1} \ldots . \mathsf{inc}^{k_l}_{j_l} . x$$

where $1 \leq i_1 < \cdots < i_n \leq n$, $k_1, \ldots, k_n > 0$ and $1 \leq j_1 < \cdots < j_l \leq n$. It is immediate to see that two terms in normal form are semantically equivalent if and only if they are syntactically equal. To see that every term is provably equal to a normal form term, note that

- occurrences of clr can be permuted (or eliminated) using $(\mathsf{clr}/\mathsf{clr}/\mathsf{c})$, $(\mathsf{inc}/\mathsf{clr}/\mathsf{a})$ and $(\mathsf{inc}/\mathsf{clr}/\mathsf{c})$
- duplicate occurrences of clr reduce to single occurrences using $(\mathsf{clr}/\mathsf{clr}/\mathsf{a})$
- occurrences of inc permute using $(\mathsf{inc}/\mathsf{inc}/\mathsf{c})$

which is readily formalized as an inductive proof on the term structure. □

For the general completeness proof, we need to impose an additional condition that will allow us to inductively reduce the branching degree of a term.

Definition 5.6. Let Σ be a signature and $\mathcal{E}$ a set of Σ-equations. A *splitting* is a finite set S of linear Σ-terms. We say that S is *admissible* over $\mathcal{E}$ if the rule

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

is admissible in $\mathcal{E}\vdash$, for all $t, u \in T_\Sigma(X)$. A linear term s is *reductive* for a degree-1 term $t \in T^1_\Sigma(X)$ if there is a linear term $t' \in T^0_\Sigma(X)$ such that $\mathcal{E} \vdash s.t = t'$; the splitting S is *reductive for* t if all $s \in S$ are reductive for t.

That is, to show that $u = v$, it suffices to show that $s.u = s.v$, for all $s \in S$, and being reductive ensures that prefixing has the additional effect of reducing the branching degree.

One standard method to establish admissibility of a rule is by showing that it is derivable. One has to be slightly careful here because in presence of the substitution rule, derivability is not stable under substitution into the assumptions (e.g. given the theory of groups, we would not dream of assuming that a formula derived under commutativity as an additional axiom would still be derivable when specialized to two elements, assuming only commutation of those elements). For clarity, we therefore define a specific (complete) notion of equational derivation:

Definition 5.7. Given a set $\mathcal{A}$ of equations, an equation $s = t$ is *strictly derivable* from $\mathcal{A}$ under $\mathcal{E}$ if $s = t$ is derivable by reflexivity, symmetry, transitivity, and congruence from equations in $\mathcal{A}$ and substitution instances of equations in $\mathcal{E}$.

Lemma 5.8. *Let C be a class of comodels, and let $\mathcal{E}$ be a set of Σ-equations such that $C \vDash \mathcal{E}$, and let*

$$\frac{s_1 = t_1 \quad \ldots \quad s_n = t_n}{s_0 = t_0}$$

where $s_0, t_0, \ldots, s_n, t_n \in T_\Sigma(V)$, be a strictly derivable rule. Then this rule is admissible in $\mathcal{E}\vdash$, that is, for all substitutions σ, if $\mathcal{E} \vdash s_i\sigma = t_i\sigma$ for $i = 1, \ldots n$, then $\mathcal{E} \vdash s_0\sigma = t_0\sigma$.

Proof. For admissibility, note that we can substitute σ into the strict derivation of $s_0 = t_0$ from $\mathcal{A} = \{s_i = t_i \mid i = 1, \ldots, n\}$ to obtain a (strict) derivation of $s\sigma = t\sigma$ from $\mathcal{A}\sigma$. This is possible because all rules (reflexivity, symmetry, transitivity, congruence) and the set of instances of $\mathcal{E}$ are stable under substitution. It follows that $\mathcal{E} \vdash s_0\sigma = t_0\sigma$. □

We now establish the requisite conditions in our running examples:

Lemma 5.9. *The splittings $S_i = \{\mathsf{clr}_i.x, \mathsf{inc}_i.x\}$, for $1 \le i \le \mathsf{c}$, are admissible over the theory $\mathcal{E}_\mathsf{C}$ of counter machines. Moreover, for every degree-1 term t there exists $1 \le i \le \mathsf{c}$ such that S_i is reductive for t.*

Proof. To see that some S_i is reductive, note that every term t of degree 1 is provably equivalent to a linear term or a term of the from $\mathsf{jd}_i(t_1, t_2)$ with t_1, t_2 linear, using equations $(\mathsf{inc/jd}/\cdot)$ and $(\mathsf{clr/jd}/\cdot)$. Hence, for $s \in S_i$ we have that $s.t$ is either linear or of the form $s.\mathsf{jd}_i(t_1, t_2)$, which is again linear modulo equational reasoning using $(\mathsf{inc/jd/a})$ and $(\mathsf{clr/jd/a})$, respectively.

To see that S_i is admissible, we show that $s = t$ is strictly derivable from $\mathsf{clr}_i.t = \mathsf{clr}_i.s$ and $\mathsf{inc}_i.t = \mathsf{inc}_i.s$ under $\mathcal{E}_\mathsf{C}$: By congruence, we obtain $\mathsf{jd}_i(\mathsf{clr}_i.t, \mathsf{inc}_i.t) = \mathsf{jd}_i(\mathsf{clr}_i.s, \mathsf{inc}_i.s)$, and hence $t = s$ using $(\mathsf{jd/clr/inc})$. □

Lemma 5.10. *The splittings $S_k = \{\mathsf{wr}^k_{a_1}.x, \dots, \mathsf{wr}^k_{a_n}.x\}$, for $k \in \mathbb{Z}$, are admissible over the theory $\mathcal{E}_\mathsf{T}$ of Turing machines. Moreover, for every degree-1 term t there exists $k \in \mathbb{Z}$ such that S_k is reductive for t.*

Proof. To see that S_k is admissible, we show that $s = t$ is strictly derivable from the assumptions $\mathsf{wr}^k_{a_i}.s = \mathsf{wr}^k_{a_i}.t$, $i = 1, \dots, n$, under $\mathcal{E}_\mathsf{T}$. By congruence, we first derive $\mathsf{rd}^k(\mathsf{wr}^k_{a_1}.s, \dots, \mathsf{wr}^k_{a_n}.s) = \mathsf{rd}^k(\mathsf{wr}^k_{a_1}.t, \dots, \mathsf{wr}^k_{a_n}.t)$, and then obtain $\mathsf{rd}^k(s, \dots, s) = \mathsf{rd}^k(t, \dots, t)$ using $(\mathsf{rd/wr/a})$. Finally, we obtain $s = t$ using $(\mathsf{rd/a})$.

Now let u be a degree-1 term over $\mathcal{E}_\mathsf{T}$. By $(\mathsf{wr/rd}/\cdot)$ and $(\mathsf{m/rd})$, u is provably equivalent to either a linear term (in which case every S_k is reductive for u) or to a term of the form $\mathsf{rd}^k(u_1, \dots, u_n)$ with $u_1, \dots, u_n$ linear. In the latter case, S_k is reductive for u since $\mathcal{E}_\mathsf{T} \vdash \mathsf{wr}^k_{a_i}.\mathsf{rd}^k(u_1, \dots, u_n) = \mathsf{wr}^k_{a_i}.u_i$ for $i = 1, \dots, n$ by $(\mathsf{wr/rd/a})$. □

Equational completeness now takes the following form.

Theorem 5.11. *Let C be a class of $\Sigma, \mathcal{E}$-comodels, and suppose that*

1. *linear terms have normal forms over C and $\mathcal{E}$*
2. *every degree-1 term has a reductive admissible splitting over $\mathcal{E}$.*

Then $\mathcal{E} \vdash s = t$ whenever $\mathsf{C} \vDash s = t$.

Proof. We show by induction on $n + m$ that $\mathcal{E} \vdash s = t$ for terms $s \in T^n_\Sigma(X)$, $t \in T^m_\Sigma(X)$ such that $\mathsf{C} \vDash s = t$. For $n + m = 0$, both s and t are linear, and the claim follows from Lemma 5.2.

So let $n + m > 0$. Then w.l.o.g. $m > 0$, so $t = t'\sigma$ for some degree-1 term t' and some substitution σ such that $\sigma(x) \in T^{m-1}_\Sigma(X)$ for all variables x. By assumption, t' has a reductive admissible splitting S. Let $u \in S$; since S is admissible, it suffices to show that $\mathcal{E} \vdash u.s = u.t$. Since u is reductive for t', $u.t$ is provably equivalent to a degree-$(m-1)$ term, and since u is linear, $u.s$ has degree n, so $\mathcal{E} \vdash u.s = u.t$ by induction. □

As a consequence, we obtain soundness and completeness for both Turing and counter machine comodels:

Theorem 5.12. *1. The equational logic generated by $\mathcal{E}_\mathsf{T}$ is sound and complete over the final Turing comodel F*
2. The equational logic generated by $\mathcal{E}_\mathsf{C}$ is sound and complete over the final counter comodel S.

Proof. By Lemmas 5.10 and 5.9, respectively, Theorem 5.11 applies in both cases. □

6. Interlude: Monads and Completeness

In this section, which is not needed for the remainder of the paper, we give an alternative characterization of equational completeness and relate comodels with algebras for the state monad.

Observation 6.1. Let Σ be a signature, and $C = (C, (\!|\cdot|\!))$ a Σ-comodel. Then C induces, for every set X, a Σ-algebra structure $[\![\cdot]\!]$ on $S_C X = (X \times C)^C$ by

$$[\![f]\!](\phi_1, \dots, \phi_n) = [\phi_1, \dots, \phi_n] \circ (\!|f|\!)$$

for n-ary $f \in \Sigma$.

Lemma 6.2. *Let C be a comodel and $S_C X$ the Σ-algebra induced by C as per Observation 6.1.*

1. *For a Σ-term t over variables $x_1, \dots, x_k$, and for $\phi_1, \dots, \phi_k \in S_C X = (C \times X)^C$ we have*
$$[\![t]\!](\phi_1, \dots, \phi_k) = [\phi_1, \dots, \phi_k] \circ (\!|t|\!). \quad (1)$$
2. *The Σ-algebra $SX = (((X \times C)^C, [\![\cdot]\!])$ satisfies all equations that are satisfied by $(C, (\!|\cdot|\!))$.*

We put $MX = UF$ where $U : \mathsf{Alg}(\Sigma, \mathcal{E}) \to \mathsf{Set}$ is the forgetful and F its left adjoint; M extends to a monad in the standard way. Initiality of MX then yields a morphism of monads $M \to S_C$, where the monad structure on the functor S_C is that of the state monad:

Corollary 6.3. *Let C be a $\Sigma, \mathcal{E}$-comodel. The unique morphisms ν^C_X of M-algebras that satisfy $\nu_X \circ \eta^M_X = \eta^{S_C}_X$ form a monad morphism $\nu^C : M \to S_C$.*

Incidentally, it follows that C induces an operational semantics for machines, which coincides with the one we introduce directly in the next section. The announced characterization of equational completeness can now be given as follows:

Theorem 6.4. *Let C be a $\Sigma, \mathcal{E}$-comodel. Then $\mathcal{E}$ is complete for C if and only if all components of ν^C are injections.*

Proof. By definition, $\nu^C_X(s) = [\![s]\!]_{S_C X}$ where $[\![\cdot]\!]_{S_C X}$ denotes term evaluation under the valuation $x \mapsto \lambda a.\,(a, x)$. By Lemma 6.2.1, we thus have $\nu^C_X(s) = [\lambda a.\,(a, x)]_{x \in X}(\!|s|\!)$. Now the map $[\lambda a.\,(a, x)]_{x \in X} : X \cdot C \to C \times X$ is bijective, so

$$\nu^C_X(s) = \nu^C_X(t) \quad \text{iff} \quad (\!|s|\!)^X = (\!|t|\!)^X \quad (2)$$

for terms s, t over variables from X. Injectivity of ν^C says that $\mathcal{E} \vdash s = t$ is equivalent to the left-hand side of (2), and completeness of $\mathcal{E}$ for C says that $\mathcal{E} \vdash s = t$ is equivalent to the right-hand side; this proves the equivalence claimed in the statement. □

The informal intepretation of this last theorem is as follows: every comodel C induces an algebraic structure on the side effect monad (with global state C). Having "enough" equations (i.e. completeness over the given comodel C) is then equivalent to this translation being faithful (ν^C having injective components).

7. From Comodels to Machines

We conceptualize both Turing and counter machines as deterministic transition systems with side effects. The transition structure is excessively simple: every state $x \in X$ is mapped to a term $\sigma(x) \in T_\Sigma(X + 1)$ where $1 = \{\mathsf{halt}\}$ deonotes successful termination. We write $x \to t$ to indicate $\sigma(x) = t$. For the whole section, we fix an algebraic signature Σ.

Definition 7.1. A Σ-*machine* is a pair (X, σ) where X is a set (of states) and $\sigma : X \to T_\Sigma(X + 1)$.

In other words, a Σ-machine is a $T_\Sigma(\_ + 1)$-coalgebra. We often denote Σ-machines as lists of transitions $x \to \sigma(x)$. Transitions of a Σ-machine produce purely syntactic (side) effects that we interpret by means of Σ-comodels.

Definition 7.2. Let Σ be a signature, $(C, (\!|\cdot|\!))$ a Σ-comodel, and (X, σ) a Σ-machine. The *operational semantics* of (X, σ) over C is given by binary relations $\to^0_C, \to^1_C$ on $T_\Sigma(X+1) \times C$ defined by

$$(x, c) \to^1_C (\sigma(x), c) \qquad \text{if } x \in X$$
$$(f(t_1, \ldots, t_n), c) \to^0_C (t_i, c') \qquad \text{if } (\!|f|\!)(c) = \mathsf{inj}_i(c').$$

We write $(t, c) \to^{<n}_C (t', c')$ if $(t, c) \to^{i_1}_C \cdots \to^{i_k}_C (t', c')$ and $i_1 + \cdots + i_k < n$, and $\to^*_C = \bigcup\{\to^{<n}_C \mid n \in \mathbf{N}\}$. We elide the subscript C if the comodel C is clear from the context. Given Σ-machines (X, σ), (Y, τ) and terms $s \in T_\Sigma(X+1)$, $t \in T_\Sigma(Y+1)$, we say that t *simulates* s *over* C, in symbols $C \vDash s \preceq t$ if for all $c, c' \in C$ we have that $(s, c) \to^*_C (\mathsf{halt}, c')$ implies $(t, c) \to^*_C (\mathsf{halt}, c')$. Moreover, we write $C \vDash_n s \preceq t$ if for all $c, c' \in C$, we have that $(s, c) \to^{<n} (\mathsf{halt}, c')$ implies $(t, c) \to^* (\mathsf{halt}, c')$, and refer to $\vDash_n$ as n*-step* or *step-indexed* simulation.

That is, if s simulates t then s terminates when run from any comodel state from which t terminates, and produces the same cumulative side effect. This allows us to formulate state equivalence as two-way simulation: if s and t simulate each other, then they terminate from the same comodel states, and produce the same cumulative side effect. (In particular, any two non-terminating computations, such as x and y given by $x \to \mathsf{l}.x$ and $y \to \mathsf{r}.y$, are equivalent.) The step-indexed version $\vDash_n s \preceq t$ is central to the soundness proof. Notice that it bounds the number of evaluation steps only for the left-hand term s; this is similar to the clause for computation types in step-indexed logical relations [3]. By definition, the index on $\to^{<n}$ and $\vDash_n$ counts only state transitions, not comodel operations.

We now introduce the simulation calculus that precisely characterizes simulation between two Σ-machines (X, σ) and (Y, τ), i.e. derives $s \preceq t$ whenever $C \vDash s \preceq t$, where $t \in T_\Sigma(X+1)$ and $s \in T_\Sigma(Y+1)$ are terms over X and Y, respectively. The calculus is built on top of an equational axiomatization of comodels, and parametrized over a collection of splittings. As already in the case of equational completeness (Theorem 5.11), soundness and completeness will rely on suitable properties of the splittings, to be discussed later.

Definition 7.3. Let $\mathcal{E}$ be a set of Σ-equations and (X, σ) and (Y, τ) be Σ-machines. The *simulation calculus*, parametrized by a collection $\mathcal{S}$ of *splittings* (i.e. finite sets of linear terms), comprises the following rules:

$$(\mathsf{ax})\frac{}{s \preceq s} \qquad (\mathsf{rw})\frac{\mathcal{E} \vdash s = s' \quad s' \preceq t' \quad \mathcal{E} \vdash t' = t}{s \preceq t}$$
$$(\mathsf{split})\frac{\{s.u \preceq s.v \mid s \in S\}}{u \preceq v}$$
$$(\mu)\frac{s \preceq t\tau}{s \preceq t} \qquad (\nu)\frac{s\sigma \preceq t}{s \preceq t}(s \in T_\Sigma(X))$$

where $S \in \mathcal{S}$ is a splitting in the second rule. The rules (ax), (rw), (split) and (μ) may be applied *inductively* whereas (ν) is applied coinductively. More formally, $\mathcal{E}, \mathcal{S} \vdash s \preceq t$ if $s \preceq t$ is an element of the mixed fixpoint

$$\nu R.\mu U.C_i(U) \cup C_c(R)$$

where $C_i(U)$ denotes the set of conclusions of the inductive rules (ax), (rw), (split), (μ), with premises in U, and $C_c(R)$ denotes the set of conclusions of the coinductive rule (ν) applied to premises in R. In case the splittings $\mathcal{S}$ and the equations $\mathcal{E}$ are clear from the context, we write $\vdash s \preceq t$ for $\mathcal{E}, \mathcal{S} \vdash s \preceq t$.

Notice that the side condition of the (ν)-rule essentially says that s will not terminate successfully before further unfolding. The rule

$$(\mathsf{imp})\frac{\mathcal{E} \vdash s = s'}{s \preceq s'}$$

(for *import*) is derivable using (ax) and (rw).

Remark 7.4. The above system admits, via its coinductive rule (ν), infinite proofs. It is clear that we cannot in general hope to find a finite representation format that fits all proofs, e.g. as rational trees: If one had any proof system for equality (or simulation) of Turing machines with finitely represented proofs whose correctness is decidable, then this would make the problem of whether a program p diverges on a given finite input w recursively enumerable, as p diverges on w iff $\mathsf{wr}_w.p \preceq y$ where wr_w writes the input w on the tape and y is some diverging program, say $y \to \mathsf{r}.y$.

We now turn to the main results of this paper, soundness and completeness of the simulation calculus, before illustrating the calculus with several examples. The following technical lemma that picks apart the stratification of the operational semantics is needed in various places.

Lemma 7.5. *Let $(C, (\!|\cdot|\!))$ be a Σ-comodel and (X, σ) a Σ-machine.*

1. *$(s, c) \to^{<k+1}_C (s', c')$ iff $(s\sigma, c) \to^{<k}_C (s', c')$ for all $c, c' \in C$ and all $s, s' \in T_\Sigma(X+1)$.*
2. *$(s, c) \to^{<n}_C (\mathsf{halt}, c')$ iff $(s', c) \to^{<n}_C (\mathsf{halt}, c')$ for all $c, c' \in C$ and $s \in T_\Sigma(X+1)$ provided $C \vDash s = s'$.*

The proof of the first item is immediate by induction on the structure of s, and omitted, the second is straightforward using $(s, c) \to^{<1} (x, c')$ iff $(\!|s|\!)(c) = \mathsf{inj}_x(c')$. As indicated, the soundness of the calculus hinges on a semantic property of the splittings:

Definition 7.6. Let $(C, (\!|\cdot|\!))$ be a Σ-comodel, and let $S \subseteq T^0_\Sigma(V)$ be a set of linear terms. Then a splitting S is *full* over C if for every $c \in C$ there exists $s(x) \in S$ and $c' \in C$ such that $(\!|s|\!)(c') = (\!|x|\!)(c)$.

Soundness then takes the following form:

Theorem 7.7 (Soundness)**.** *Let $(C, (\!|\cdot|\!))$ be a $\Sigma, \mathcal{E}$-comodel and (X, σ), (Y, τ) be Σ-machines.*

If every splitting $S \in \mathcal{S}$ is full over C, then $C \vDash s \preceq t$ whenever $\mathcal{E}, \mathcal{S} \vdash s \preceq t$, for all $s \in T_\Sigma(X+1)$ and all $t \in T_\Sigma(Y+1)$.

Proof. For $n \in \mathbf{N}$, put

$$\vdash_{n+1} = \mu U.C_i(U) \cup C_c(\{s \preceq t \mid \vdash_n s \preceq t\})$$

and $\vdash_0 s \preceq t$ for all $s, t \in T_\Sigma(X)$. Then $\vdash s \preceq t$ if $\vdash_n s \preceq t$ for all $n \in \omega$. It suffices to show that $\vdash_n \subseteq \vDash_n$ for all $n \in \mathbf{N}$ (cf. Definition 7.2). For $n = 0$ there is nothing to show. So suppose that $n > 0$. We proceed by induction on the derivation of $\vdash_n s \preceq t$ where we treat the individual rules in turn.

- (ν): Suppose that $\vdash_{n-1} s\sigma \preceq t$ so that by induction, $\vDash_{n-1} s\sigma \preceq t$. Then $\vDash_n s \preceq t$ by Lemma 7.5.1.
- (μ): Suppose that $\vdash_n s \preceq t\sigma$ so that by induction, $\vDash_n s \preceq t\sigma$. Then $\vDash_n s \preceq t$ by Lemma 7.5.1.
- (ax): Trivial.
- (rw): Suppose that $\mathcal{E} \vdash s = s'$, $\mathcal{E} \vdash t = t'$, and $\vdash_n s' \preceq t'$. We have $\vDash_n s' \preceq t'$ by induction, and need to show that $\vDash_n s \preceq t$. This follows from Lemma 7.5.2.
- (split): Suppose that $\vdash_n s.u \preceq s.v$ for all $s \in S$ where $S \in \mathcal{S}$ is a splitting. To see that $\vDash_n u \preceq v$ let $c \in C$ and assume that

$(u,c) \to^{<n} (\mathsf{halt}, c'')$. Since S is full, there exist $c' \in C$ and $s(x) \in S$ such that $(\!|s|\!)(c') = (\!|x|\!)(c)$ whence

$$(s.u, c') \to^0 (u,c) \to^{<n} (\mathsf{halt}, c'').$$

By induction, this gives

$$(s.v, c') \to^* (\mathsf{halt}, c'')$$

which implies that $(v,c) \to^* (\mathsf{halt}, c'')$ as required. □

Remark 7.8. Soundness of coinductive proof systems breaks more easily than for standard well-founded proof systems. E.g. it is by no means the case that the system will remain sound when extended with semantically sound inductive rules, i.e. rules whose conclusion is valid whenever the premises are valid. As a simple counterexample, consider the inverse rule of (ν), which is clearly semantically sound in this sense. Adding this rule to the system as an inductive rule, however, will make the system inconsistent in the sense that all inequalities $s \preceq t$ become provable, via coinductive proofs that keep reducing $s\sigma \preceq t$ and $s \preceq t$ to each other using (ν) and its inverse.

The key point in the above soundness proof is that all inductive rules are sound for the step-indexed notion of satisfaction, $\vDash_n$, and we can soundly add any rule to the system as an inductive rule if it is sound for $\vDash_n$. This holds in particular for the *congruence rule*

$$(\mathsf{cg})\ \frac{s_1 \preceq t_1 \quad \dots \quad s_n \preceq t_n}{f(s_1,\dots,s_n) \preceq f(t_1,\dots,t_n)}.$$

One rule that clearly *fails* to be sound for $\vDash_n$ is transitivity of $\preceq$ (if $\vDash_n s \preceq t$ and $\vDash_n t \preceq u$ then $\vDash_n s \preceq u$ may still fail, say when s terminates in less than n steps, t terminates in more than n steps, and u diverges). Interestingly, more or less the same problem appears in a quite different context, the failure of the transitivity proof for the Appel-McAllester model of recursive types as discussed in [3]. Of course, our step-indexed simulation relation is nevertheless transitive, but it remains an open question whether transitivity can be soundly added to our system as an inductive rule.

The crucial step in the completeness proof that we are about to give is the ability to inductively reduce the branching degree of terms so that we get to apply completeness for linear terms at some point. This is as in the proof of equational completeness for comodels (Theorem 5.11) and indeed the general completeness theorem makes the same assumptions, but additionally requires that the comodel in question be compact. The latter is automatic for final comodels (Corollary 4.7).

Theorem 7.9 (Completeness)**.** *Let $(C, (\!|\cdot|\!))$ be a $\Sigma,\mathcal{E}$-comodel that carries a compact topology for which all $(\!|f|\!)$, $f \in \Sigma$ are continuous, and let (X,σ), (Y,τ) be Σ-machines. Assume that*

- *linear Σ-terms have normal forms over $\mathcal{E}$ and C;*
- *every splitting $S \in \mathcal{S}$ is admissible over $\mathcal{E}$;*
- *every degree-1 term has a reductive splitting $S \in \mathcal{S}$.*

Then $\mathcal{E},\mathcal{S} \vdash s \preceq t$ whenever $C \vDash s \preceq t$, for all $s \in T_\Sigma(X+1)$ and $t \in T_\Sigma(Y+1)$.

Proof. As $\mathcal{E},\mathcal{S} \vdash s \preceq t$ is a greatest fixpoint, it suffices to show that

$$V \subseteq \mu U.(C_i(U) \cup C_c(V))$$

where $V = \{s \preceq t \mid C \vDash s \preceq t\}$ is the set of all semantically simulating pairs of terms. Let $s \preceq t \in V$ where $s \in T^n_\Sigma(X+1)$ and $t \in T^m_\Sigma(Y+1)$; we proceed by induction on $n+m$. If $n+m=0$, then s and t are linear, so we have either $s \in T^0_\Sigma(X)$ or $s \in T^0_\Sigma(1)$.

In case $s \in T^0_\Sigma(X)$, the assumption $C \vDash s \preceq t$ implies that $C \vDash s\sigma \preceq t$, so $s \preceq t \in C_c(V) \subseteq \mu U.C_i(U) \cup C_c(V)$ as required.

In case $s \in T^0_\Sigma(1)$ we claim that there exists n such that $C \vDash s = t\sigma^n$ where $t\sigma^n$ is the n-fold application of σ to t, defined formally bty $t\sigma^0 = t$ and $t\sigma^{n+1} = (t\sigma^n)\sigma$.

To see this, put $o_k = \{c \in C \mid C, c \vDash s = t\sigma^k\}$. As $C \vDash s \preceq t$ (and s contains $\mathsf{halt} \in 1$ as the only variable) and the interpretation of terms is continuous, $C = \bigcup_k o_k$ is an open cover of C, and since C is compact, it follows that $C \vDash s = t\sigma^k$ for some k. This proves the claim. Given $C \vDash s = t\sigma^k$, we obtain that $s \preceq t\sigma^k \in \mu U.C_i(U) \cup C_c(V)$ using (rw) and (ax) and therefore $s \preceq t \in \mu U.C_i(U) \cup C_c(V)$ using (μ) k times.

This finishes the proof for the base case $n+m=0$. If $n+m>0$, then either s or t has strictly positive degree and hence decomposes into a degree-1 term and a substitution as in the proof of Theorem 7.7. The former has a reductive splitting $S \in \mathcal{S}$, and it suffices by (split) to show that $u.s \preceq u.v \in \mu U.C_i(U) \cup C_c(V)$ for all $u \in S$. Since $C \vDash s \preceq t$, we have $C \vDash u.s \preceq u.t$. Using (rw) we may assume that the sum of the degrees of each $u.s$ and $u.t$ is smaller than that of s and t, and apply the inductive hypothesis. □

Remark 7.10. Observe that from the above proof, one extracts an easy argument that shows that if C is compact, then every program that terminates on all inputs runs in constant time. This is does not mean that the expressivity of the formalism is unexpectedly restricted but is instead based on the fact that typical compact comodels, in particular final ones, allow for infinite input, e.g. tapes filled with infinite input strings, and any program that terminates on all inputs will have to terminate also on infinite inputs. Typical verification goals will compare a program to a simpler (maybe less efficient) program that serves as the specification. As a simple example, consider programs $binc$ and $uinc$, over the usual alphabet $\{0,1,B\}$ with B read as a blank symbol and 0 and 1 as bits, that increment a binary number and a unary number, respectively, as well as a binary-to-unary conversion program $b2u$. Then the inequality $b2u.uinc \preceq binc.b2u$ serves as a specification of $binc$. Both sides involve programs that may fail to terminate on some inputs, namely if the input is an infinite bit string; still, the specification is the same as if all programs were total.

It is routine to verify that our requirements for soundness and completeness are met in our running examples:

Example 7.11. The simulation calculus, instantiated with the splittings used in the respective proofs of equational completeness, is sound and complete for both counter machines and Turing machines, over the final comodels F (for Turing machines, Lemma 3.2) and C (for counter machines, Lemma 3.4); that is:

1. If $\mathcal{S}_\mathsf{T} = \{S_k \mid k \in \mathbb{Z}\}$ is the set of splittings from Lemma 5.10, then $\mathcal{E}_\mathsf{T},\mathcal{S}_\mathsf{T} \vdash s \preceq t \iff F \vDash s \preceq t$.
2. If $\mathcal{S}_\mathsf{C} = \{S\}$ and S is the set of splittings from Lemma 5.9, then $\mathcal{E}_\mathsf{C},\mathcal{S}_\mathsf{C} \vdash s \preceq t \iff C \vDash s \preceq t$.

Proof. By soundness and completeness of the generic simulation calculus, once we have convinced ourselves (for soundness) that the splittings given are indeed full, which is however evident. □

To facilitate examples, we fix the following terminology.

Definition 7.12. A *coinductive set* is a set R of inequalities such that

$$R \subseteq \mu P.\,(C_i(P) \cup C_c(P) \cup C_c(\mu U.\,(C_i(U) \cup R))).$$

In words, every inequality in a coinductive set R has a well-founded proof, using all rules, from conclusions of the coinductive rule (ν) whose premises can be proved using the inductive rules from elements of R.

Lemma 7.13. *Let R be a coinductive set. Then $\mathcal{E} \vdash R$.*

Proof. We show that $\bar{R} \coloneqq \mu U.\,(C_i(U) \cup C_c(U) \cup R)$ is a postfixpoint of the functional defining derivability, i.e. that

$$\bar{R} \subseteq \mu P.C_i(P) \cup C_c(\bar{R}).$$

The right hand side is closed under C_i by its own definition, and closed under C_c by the definition of $\bar{R}$. Thus, it suffices to show $R \subseteq \mu P.C_i(P) \cup C_c(\bar{R})$. Since R is a coinductive set, it suffices to show

$$\mu P.\,(C_i(P) \cup C_c(P) \cup C_c(\mu U.\,(C_i(U) \cup R))) \subseteq \mu P.C_i(P) \cup C_c(\bar{R}).$$

Again because the right hand side is closed under C_i and C_c, it suffices to show

$$C_c(\mu U.\,(C_i(U) \cup R)) \subseteq \mu P.C_i(P) \cup C_c(\bar{R}).$$

But in fact the left-hand side is even contained in $C_c(\bar{R})$. □

The following examples illustrate the calculus for Turing machines over the two-symbol alphabet $\{0, 1\}$.

Example 7.14. 1. For the nonterminating program $y \to \mathsf{r}.y$ we can prove $y \preceq \mathsf{halt}$ by indefinite application of the (ν)-rule: repeated backwards application of (ν) starting from $y \preceq \mathsf{halt}$ successively reduces the goal to inequalities $\mathsf{r}.y \preceq \mathsf{halt}$, $\mathsf{r}.\mathsf{r}.y \preceq \mathsf{halt}$, ... The proof tree fails to be rational, which is unsurprising as the program y goes through infinitely many tape states (counting the head position as part of the tape state).

2. Contrastingly, the nonterminating program $y \to \mathsf{r}.z$, $z \to \mathsf{l}.y$ only visits finitely many tape states. And indeed, we do find a rational proof tree for $y \preceq \mathsf{halt}$ (formally, $\{y \preceq \mathsf{halt}\}$ is a coinductive set):

$$\cfrac{\cfrac{\cfrac{\vdots}{y \preceq \mathsf{halt}}\,(\mathsf{rw})}{\mathsf{l}.\mathsf{r}.y \preceq \mathsf{halt}}\,2\times(\nu)}{y \preceq \mathsf{halt}}$$

3. On the other hand, we cannot prove the invalid inequality $\mathsf{halt} \preceq y$ for $y \to \mathsf{r}.y$: backwards application of the inductive rules (including (μ)) will always yield goals whose left hand sides contain halt, so (ν) never becomes applicable due to its side condition $s \in T_\Sigma(X)$.

4. Equivalence of isomorphic systems is proved quickly by means of (ν), (μ), and (cg) (see Remark 7.8). E.g. for $x \to \mathsf{r}.x$ and $y \to \mathsf{r}.y$, to show $x \preceq y$ we first apply (ν) and then (μ), obtaining the goal $\mathsf{r}.x \preceq \mathsf{r}.y$, and then apply (cg), obtaining the original goal $x \preceq y$; since we have applied (ν) in between, this constitutes a coinductive proof in our system.

5. As an example that terminates in some cases and diverges in others, consider the Turing program $x \to \mathsf{rd}(\mathsf{halt}, x)$, which stops if the current tape cell contains 0 and otherwise diverges. Again, we show $x \preceq \mathsf{halt}$:

$$\cfrac{\cfrac{\Pi\qquad\cfrac{\cfrac{\vdots}{x \preceq \mathsf{halt}}\,(\mathsf{cg})}{\mathsf{wr}_1.x \preceq \mathsf{wr}_1.\mathsf{halt}}\,(\mathsf{rw})}{\mathsf{rd}(\mathsf{halt}, x) \preceq \mathsf{wr}_1.\mathsf{halt}}\,(\mathsf{split})}{\mathsf{rd}(\mathsf{halt}, x) \preceq \mathsf{halt}}\,(nu)$$

where (split) is applied with $S = \{\mathsf{wr}_0, \mathsf{wr}_1\}$ and Π represents a proof of $\mathsf{wr}_0.\mathsf{rd}(\mathsf{halt}, x) \preceq \mathsf{wr}_0.\mathsf{halt}$ using (rw) and (ax). Formally, we have shown that $\{x \preceq \mathsf{halt}\}$ is a coinductive set. Again, the proof tree is rational in this case, matching the observation that x visits only finitely many memory states (just one, in fact).

6. Consider the Turing program

$$x \to \mathsf{r}.\mathsf{rd}(\mathsf{l}.y, x) \quad y \to \mathsf{rd}(\mathsf{halt}, \mathsf{l}.y);$$

that is, y searches the tape leftwards for the first occurrence of 0 (possibly at the current tape position), and x searches the tape for the first occurrence of 0 strictly to the right of the current tape position and then moves one cell to the left and calls y. We show $x \preceq y$ by establishing that

$$R = \{\mathsf{wr}_{1^n}.\mathsf{r}^n.x \preceq \mathsf{wr}_{1^n}.y \mid n \geq 0\}$$

is a coinductive set, where we write wr_{1^n} for the program $\mathsf{wr}_1^0.\ldots.\mathsf{wr}_1^{n-1}.z$ that writes the string 1^n consisting of n 1's on the tape and leaves the head position unchanged: For $n \geq 0$, we have

$$\cfrac{\Pi\qquad\cfrac{\cfrac{\vdots}{\mathsf{wr}_{1^{n+1}}.\mathsf{r}^{n+1}.x \preceq \mathsf{wr}_{1^{n+1}}.y}\,(\nu,\mathsf{rw})}{\mathsf{wr}_1^n.\mathsf{wr}_{1^n}.\mathsf{r}^n.x \preceq \mathsf{wr}_1^n.\mathsf{wr}_{1^n}.y}}{\mathsf{wr}_{1^n}.\mathsf{r}^n.x \preceq \mathsf{wr}_{1^n}.y}\,(\mathsf{split})$$

where (split) is applied with $S = \{\mathsf{wr}_0^n, \mathsf{wr}_1^n\}$ and Π is the proof

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\mathsf{wr}_0^n.\mathsf{wr}_{1^n}.y \preceq \mathsf{wr}_0^n.\mathsf{wr}_{1^n}.y}\,(\mathsf{ax})}{\vdots}}{\mathsf{wr}_0^n.\mathsf{wr}_{1^n}.\mathsf{r}^{n-1}.y \preceq \mathsf{wr}_0^n.\mathsf{wr}_{1^n}.y}\,(\mathsf{rw})}{\mathsf{wr}_0^n.\mathsf{wr}_{1^n}.\mathsf{r}^n.\mathsf{l}.y \preceq \mathsf{wr}_0^n.\mathsf{wr}_{1^n}.y}\,(\nu,\mathsf{rw})}{\mathsf{wr}_0^n.\mathsf{wr}_{1^n}.\mathsf{r}^n.x \preceq \mathsf{wr}_0^n.\mathsf{wr}_{1^n}.y}$$

with the ellipsis indicating repeated application of (ν) and rewriting.

We complement these examples with a derivation in the simulation calculus for counter machines.

Example 7.15. We consider machines with two counters, 0 and 1. Let the state x have the transition

$$x \to \mathsf{jd}_1(\mathsf{halt}, \mathsf{inc}_0.x).$$

That is, x adds the content of counter 1 to counter 0 by repeatedly incrementing counter 0 and decrementing counter 1, eventually leaving counter 1 being 0. Let the state y have the transition

$$y \to \mathsf{jd}_0(x, \mathsf{inc}_1.y);$$

so y first adds the content of counter 0 to the content of counter 1 and then calls x to move counter 1 to counter 0, leaving counter 1 being 0.

When these programs are interpreted over the final counter comodel (Lemma 3.4), counters store a finite value $n \in \mathbf{N}$ or ∞. Accordingly, we have different sources of non-termination:

- If counter 1 contains ∞, then x will fail to terminate. In this case, state y will also fail to terminate: even if counter 0 is finite, y will eventually call x and diverge.
- If counter 0 contains ∞, then y will fail to terminate, whereas x will terminate as long as counter 1 is finite.

It is thus not true that $x \preceq y$; we show $y \preceq x$. To this end, we establish that the set

$$R = \{\mathsf{inc}_1^n.y \preceq \mathsf{inc}_0^n.x \mid n \geq 0\},$$

which contains our goal as the case for $n = 0$, is coinductive.

So let $n \geq 0$. We have the derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{clr}_0.\mathsf{inc}_0^n.x \preceq \mathsf{clr}_0.\mathsf{inc}_0^n.x}(\mathsf{ax})}}{\vdots}(\mathsf{rw},\nu)}{\mathsf{clr}_0.\mathsf{inc}_1^{n-1}.\mathsf{inc}_0.x \preceq \mathsf{clr}_0.\mathsf{inc}_0^n.x}(\mathsf{rw},\nu)}{\mathsf{clr}_0.\mathsf{inc}_1^n.x \preceq \mathsf{clr}_0.\mathsf{inc}_0^n.x} \quad \mathsf{inc}_1^{n+1}.y \preceq \mathsf{inc}_0^{n+1}.x}(\mathsf{split},\mathsf{rw})}{\mathsf{jd}_0(\mathsf{inc}_1^n.x, \mathsf{inc}_1^{n+1}.y) \preceq \mathsf{inc}_0^n.x}(\mathsf{rw})}{\mathsf{inc}_1^n.\mathsf{jd}_0(x, \mathsf{inc}_1.y) \preceq \mathsf{inc}_0^n.x}(\nu)}{\mathsf{inc}_1^n.y \preceq \mathsf{inc}_0^n.x}(\nu)$$

where (split) is applied with $S = \{\mathsf{clr}_0, \mathsf{inc}_0\}$ and ellipsis indicates repeated unfolding on the left (ν) and equational rewriting (rw). The right branch ends in an element of R with an interceding application of (ν), showing that R is coinductive and finishing the proof.

Remark 7.16. The above example shows that the use of the final counter comodel spoils the expected equivalence of the states x, y, allowing for simulation only in one direction. One can easily encode simulation w.r.t. the standard counter comodel with counters $1, \ldots, n$ storing only finite values into simulation w.r.t. the final comodel, for simplicity with an additional counter $n+1$: let p be a program that adds the counters $i = 1, \ldots, n$ to the counter $n+1$ by iterated decrementation of i and incrementation of $n+1$. Then for counter machine terms s, t using only the counters $1, \ldots, n$, we have $s \preceq t$ w.r.t. the standard comodel iff $p.s \preceq p.t$ w.r.t. the final comodel.

8. Conclusions

We have formalized a generic notion of state machines that produce algebraic effects; as runnning examples, we have discussed two computational models, Turing machines and counter machines, in these terms. Our main result is a sound and complete calculus for computational simulation, i.e. termination from at least the same starting memory states, with the same cumulative effect. We have argued that this cannot be achieved by a purely inductive system (as otherwise equality of recursive functions would become recursively enumerable, violating, e.g., Rice's theorem) and have given a mixed coinductive/inductive calculus instead.

We believe that that this is the first attempt at a logical treatment of extensional equality of programs in state-based computational models. Many questions are still open, in particular:

- Under what conditions does a simulation $s \preceq t$ have a rational proof, i.e. a finitely representable non-wellfounded proof?
- Can Σ-machines be equipped with a meaningful notion of *polynomial equivalence* and can this be axiomatized logically?
- Can the approach presented here be extended to a form of mathematical axiomatic semantics for programming languages including structured control constructs (e.g. loop constructs), analogous to Turi and Plotkin's mathematical operational semantics [20]?

We leave these questions for further work.

Acknowledgments The authors wish to thank James Brotherston for useful discussions, and Erwin R. Catesbeiana for favourable remarks on empty comodels.

References

[1] F. Abou-Saleh and D. Pattinson. Comodels and effects in mathematical operational semantics. In *Foundations of Software Science and Computation Structures, FoSSaCS 2013*, vol. 7794 of *LNCS*, pp. 129–144. Springer, 2013.

[2] S. Abramsky and L. Ong. Full abstraction in the lazy lambda calculus. *Inf. Comput.*, 105:159–267, 1993.

[3] A. Ahmed. Step-indexed syntactic logical relations for recursive and quantified types. In *European Symposium on Programming, ESOP 2006*, vol. 3924 of *LNCS*, pp. 69–83. Springer, 2006.

[4] J. Brotherston, N. Gorogiannis, and R. Petersen. A generic cyclic theorem prover. In *Programming Languages and Systems, APLAS 2012*, vol. 7705 of *LNCS*, pp. 350–367. Springer, 2012.

[5] J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21:1177–1216, 2011.

[6] J. Endrullis, H. Hansen, D. Hendriks, A. Polonsky, and A. Silva. A coinductive framework for infinitary rewriting and equational reasoning. In *Rewriting Techniques and Applications, RTA 2015*, vol. 36 of *LIPIcs*, pp. 143–159. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[7] S. Goncharov, S. Milius, and A. Silva. Towards a coalgebraic Chomsky hierarchy. In *Theoretical Computer Science, IFIP TCS 2014*, vol. 8705 of *LNCS*, pp. 265–280. Springer, 2014.

[8] S. Goncharov, L. Schröder, and T. Mossakowski. Kleene monads: Handling iteration in a framework of generic effects. In *Algebra and Coalgebra in Computer Science, CALCO 2009*, vol. 5728 of *LNCS*, pp. 18–33. Springer, 2009.

[9] S. Kerkhoff. A general duality theory for clones. *Int. J. Algebra Comput.*, 23:457–502, 2013.

[10] S. Kerkhoff. Dualizing clones as models of lawvere theories. In *Algebra, Coalgebra and Topology, WACT 2013*, vol. 303 of *ENTCS*, pp. 79–105, 2014.

[11] D. Kozen. Kleene algebra with tests. *ACM TOPLAS*, 19:427–443, 1997.

[12] M. Minsky. Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Ann. Math.*, 74:437–455, 1961.

[13] R. Møgelberg and S. Staton. Linear usage of state. *Log. Meth. Comput. Sci.*, 10, 2014.

[14] D. Pattinson and L. Schröder. Sound and complete equational reasoning over comodels. In *Mathematical Foundations of Programming Semantics, MFPS 2015*, vol. 319 of *ENTCS*, pp. 315–331, 2015.

[15] G. Plotkin. Adequacy for algebraic effects with state. In *Algebra and Coalgebra in Computer Science, CALCO 2005*, vol. 3629 of *LNCS*, p. 51. Springer, 2005.

[16] G. Plotkin and J. Power. Tensors of comodels and models for operational semantics. In *Mathematical Foundations of Programming Semantics, MFPS 2008*, vol. 218 of *ENTCS*, pp. 295–311, 2008.

[17] J. Power. Semantics for local computational effects. In *Mathematical Foundations of Programming Semantics, MFPS 2006*, vol. 158 of *ENTCS*, pp. 355–371, 2006.

[18] J. Power and O. Shkaravska. From comodels to coalgebras: State and arrays. In *Coalgebraic Methods in Computer Science, CMCS 2004*, vol. 106 of *ENTCS*, pp. 297–314, 2004.

[19] J. Rutten. Universal Coalgebra: A theory of systems. *Theoret. Comput. Sci.*, 249(1):3–80, 2000.

[20] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS 1997*, pp. 280–291. IEEE, 1997.