

Incompleteness of Behavioral Logics

Samuel Buss¹

*Department of Mathematics
University of California, San Diego*

Grigore Roşu²

*Department of Computer Science & Engineering
University of California, San Diego*

Abstract

Incompleteness results for behavioral logics are investigated. We show that there is a basic finite behavioral specification for which the behavioral satisfaction problem is not recursively enumerable, which means that there are no automatic methods for proving all true statements; in particular, behavioral logics do not admit complete deduction systems. This holds for all of the behavioral logics of which we are aware. We also prove that the behavioral satisfaction problem is not co-recursively enumerable, which means that there is no automatic way to refute false statements in behavioral logics. In fact we show stronger results, that all behavioral logics are Π_2^0 -hard, and that, for some data algebras, the complexity of behavioral satisfaction is not even arithmetic; matching upper bounds are established for some behavioral logics. In addition, we show for the fixed-data case that if operations may have more than one hidden argument, then final models need not exist, so that the coalgebraic flavor of behavioral logic is lost.

1 Introduction

The results in this paper are a consequence of our and other scientists' effort to find complete deduction systems and Birkhoff-like axiomatizability results for various versions of behavioral logics [21,3,12,2,17,16]. The fact that equational reasoning was not strong enough to derive equalities provable by coinduction [18,10] and coinduction was not strong enough to prove equalities provable by

¹ Supported in part by NSF grant DMS-9803515 and by grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech republic)

² Also Fundamentals of Computing, Faculty of Mathematics, University of Bucharest, Romania.

circular coinduction [19], and the fact that the Birkhoff-like equational axiomatizability for the coalgebraic version of hidden algebra was not pure [16,14], made us believe that behavioral satisfaction might not admit a complete axiomatization and thus to motivate the present work.

We show the incompleteness of those approaches to behavioral specification and satisfaction currently in use which make a clear distinction between *visible* and *hidden* sorts, the equality being *strict* on the visible sorts and *behavioral* on the hidden sorts, that is, meant as “indistinguishability under experiments”. We generically call these logics *behavioral logics*. These include hidden algebra [6,7,9,8], coherent hidden algebra [5,4], observational logic [13,1] (the equational version) and other recent generalizations of hidden algebra [18,19]. We are aware that there is not a full consensus on which notation is best to deal with behavioral logics in general. However, we decided to take the distinction between visible and hidden sorts, and hence the notion of behavioral equivalence, as basics within our approach to incompleteness, because this is conceptually a very general framework capturing most of the situations of practical interest. By using such a general approach, it is possible to easily obtain incompleteness results for other approaches to behavioral logics as consequences of the results of the present paper.

In order to show that a logic does not admit a complete axiomatization it is necessary and sufficient to show that the satisfaction problem is not recursively enumerable. That is to say, there is no algorithm taking as input a specification and a sentence, and returning “Yes” if and only if the sentence is satisfied by all specification’s models, otherwise looping forever. The intuition behind this technique is that if a complete deduction system existed, then an algorithm would be to just generate and check all possible proofs. In this paper, we show a stronger result, that there are finite specifications in all behavioral logics above, for which the satisfaction problem is Π_2^0 -hard and thus is not recursively enumerable. Moreover, in many cases, the satisfaction problem for behavioral logic is in Π_2^0 , and thus the satisfaction problem can be Π_2^0 -complete. The fact that behavioral satisfaction can be Π_2^0 -hard means not only that there is no complete axiomatization for the behavioral satisfaction problem, but also that the complement of the behavioral satisfaction problem is not recursively enumerable. This means there is no way to algorithmically refute the sentences which are not behavioral consequences of some finite behavioral specifications. Notice that this result is even less intuitive than the incompleteness result, because for any fixed computable model (in which the interpretations of operations are computable functions), the satisfaction problem is co-r.e.

The fact that the behavioral logics we presented are incomplete does not mean that something is wrong with the hidden algebra notation or that the question we addressed is ill-posed. For an analogy, the standard first-order theory of the natural numbers is of course well-known to be incomplete, but it is nonetheless the correct first-order theory for the natural numbers. From

considerations such as Rice’s theorem, we should expect that any formal system which is strong enough to capture a significant amount of the behavior of computer systems is likely to be incomplete in some way; namely, it may lack expressive power, or may lack a complete axiomatization, or may admit unintended nonstandard models. Even if our results may seem negative, in fact they motivate work on new automatic proof techniques and algorithms to prove behavioral equivalences, such as Δ -coinduction [17] and/or circular coinduction [19], which may be applicable for larger classes of problems, in the same way in which induction is a very useful proof technique for natural numbers.

The outline of the paper is as follows. In Section 2, we review the basic definitions of two simple behavioral logics, with fixed data and loose data, respectively, which we show incomplete and from which all the other current behavioral logics are derived. We also point out how other behavioral logics generalize one of these two simple logics by relaxing their syntactic constraints. As a side point, we prove that when operations can have more than one hidden input, then there may be no final hidden algebra (in contrast to classical results on the existence of final algebras when operations have only a single hidden input). In Section 3, we review the Turing machines and introduce our notations and the TOTALITY problem, which is a classical example of a Π_2^0 -complete decision problem. Section 4 proves the main result of the paper, namely that behavioral satisfaction can be Π_2^0 -hard by a reduction from TOTALITY. Matching upper bounds are shown in Section 5 for two cases of behavioral logics, with fixed data and loose data, respectively. In Section 6 we show that the fixed-data hidden algebra logic for data algebra of the integers with successor (an algebra in which first-order validity is decidable), the behavioral satisfaction problem can be Π_1^1 -hard, so not even in the arithmetic hierarchy. Again, a matching upper bound is established.

2 Two Basic Behavioral Logics

The different approaches to behavioral specification and satisfaction can be classified in two broad categories, depending on whether a fixed data algebra is assumed for all models or not. In this section we introduce two very restrictive behavioral logics as references for the two categories. The other behavioral logics currently in use can be derived from one of these two basic logics, by relaxing their syntactic constraints. The fact that these two logics are syntactically very restrictive is a positive issue w.r.t. incompleteness, since their incompleteness implies the incompleteness of the other more relaxed behavioral logics. We have tried to prove our incompleteness results for the weakest (most restrictive) logics, so as to make our results as general as possible.

2.1 Fixed Data

Hidden algebra first appeared in [6] and was further investigated in [7,9,8] and many others. Here, we present an over-simplified version of hidden algebra logic and refer to it as *basic fixed-data hidden algebra* in the rest of the paper.

Definition 2.1 A *hidden signature* is a $\{v, h\}$ -sorted signature Σ , where v is the *visible sort* and h is the *hidden sort*, consisting of:

- two constants of visible sort, *true* and *false*, often called **the data**;
- one **attribute**, i.e., an operation $a : h \rightarrow v$;
- a finite set of **methods**, i.e., operations $m : h \rightarrow h$.

A *hidden Σ -algebra* is a Σ -algebra A such that $A_v = \{true, false\}$. A *morphism of hidden Σ -algebras* is a morphisms of Σ -algebras which is the identity on v .

Note that, in keeping with our desire to restrict the systems as much as possible, all operations (attributes and methods) are unary. The “fixed-data” terminology comes from the fact that all models have a fixed interpretation of the two visible constants. One important feature of basic fixed-data hidden algebra is its coalgebraic aspect:

Theorem 2.2 *The category of hidden algebras is isomorphic to a category of coalgebras over a polynomial functor; in particular, there exists a final hidden Σ -algebra.*

Behavioral equivalence can be defined in many equivalent ways. We prefer one based on contexts in the present paper:

Definition 2.3 *Given a hidden signature Σ and a special variable \star of sort h , a **context** c is a term $a(m_1(m_2(\dots(m_j(\star))\dots)))$, for some $j \geq 0$ (not necessarily distinct) methods m_1, m_2, \dots, m_j . Given a term t of sort h , i.e., a term $m'_1(m'_2(\dots(m'_i(x))\dots))$ over a variable x of sort h , we let $c[t]$ denote the term $a(m_1(m_2(\dots(m_j(m'_1(m'_2(\dots(m'_i(x))\dots))))\dots)))$.*

Contexts can be viewed as experiments. They consist of a series of methods changing the state followed by the attribute which “observes” the state. Notice that the contexts and the two visible constants are the only (modulo renaming of variables) terms of visible sort.

Intuitively, two terms are behaviorally equivalent iff they give the same results under all experiments. To make this formal, we next define what it means for a Σ -algebra to behaviorally satisfy an equation $(\forall x) t = t'$. We use \models to denote behavioral satisfaction and \models to denote ordinary equational satisfaction.

Definition 2.4 *Let A be a hidden Σ -algebra. Then A **behaviorally satisfies** a Σ -equation $(\forall x) t = t'$ of sort h , written $A \models_{\Sigma} (\forall x) t = t'$, if and only if, for all contexts c , $A \models_{\Sigma} (\forall x) c[t] = c[t']$. Behavioral satisfaction is ordinary*

satisfaction on equations of visible sort, i.e., if t and t' have visible sort, then $A \models_{\Sigma} (\forall x) t = t'$ iff $A \models_{\Sigma} (\forall x) t = t'$.

A **behavioral specification** is a pair (Σ, E) , where Σ is a hidden signature and E is a set of Σ -equations. A **model** of $\mathcal{B} = (\Sigma, E)$ is a hidden algebra A which behaviorally satisfies all equations in E , and in this case we may write $A \models \mathcal{B}$. Given a Σ -equation e , we may also write $\mathcal{B} \models^{fd} e$ whenever all models of \mathcal{B} behaviorally satisfy e .

Equational reasoning is sound for behavioral satisfaction [9], but not complete. We will show in the next section that actually there is no complete deduction system for behavioral satisfaction. More precisely, we show that for some behavioral specifications $\mathcal{B} = (\Sigma, E)$, the following problem called $BSAT_{\mathcal{B}}^{fd}$:

INPUT: A Σ -equation e ;
 OUTPUT: $\mathcal{B} \models^{fd} e$?

is not recursively enumerable even for finite E .

The rest of this subsection is dedicated to extensions of basic fixed-data hidden algebra, relaxing the constraints on hidden signatures. The incompleteness of basic fixed-data hidden algebra obviously yields the incompleteness of all these extended frameworks. The hasty reader may skip this part.

A first generalization is hidden algebra [6,7,9,8], where more visible and hidden sorts are allowed, and the only restriction on operations is to have *at most* one hidden argument. A fixed data algebra over the visible signature must be protected by all models.

A second generalization is to drop the restriction that the operations must have at most one hidden argument [18,10]. Thus, the declarational power of behavioral specifications is increased (for example, sets with union and intersection can now be naturally specified), still providing the proof techniques of hidden algebra, including coinduction, but the coalgebraic aspect is destroyed, in particular:

Proposition 2.5 *For some Σ with many-hidden-argument operations, there is no final hidden Σ -algebra.*

Proof. For example, let Σ contain only one operation $\pi : hh \rightarrow v$. Let D be the set $\{true, false\}$. Suppose that $\mathcal{F} = ((D, F_h), F_{\pi} : F_h \times F_h \rightarrow D)$ is a final hidden Σ -algebra. Since any set can be organized as a hidden Σ -algebra by interpreting π in an arbitrary way, one gets that there exists a function from any set to F_h , so F_h is nonempty. Let \mathcal{P} be the hidden Σ -algebra $((D, \mathcal{P}(F_h)), P_{\pi} : \mathcal{P}(F_h) \times \mathcal{P}(F_h) \rightarrow D)$, where $\mathcal{P}(F_h)$ is the power set of F_h , and $P_{\pi}(A, B)$ is *true* if $A = B$ and *false* otherwise. Then there exists a unique³ morphism of hidden Σ -algebras $\alpha : \mathcal{P} \rightarrow \mathcal{F}$; suppose that $\alpha = (1_v : D \rightarrow D, \alpha_h : \mathcal{P}(F_h) \rightarrow F_h)$. Notice that $A \neq B$ implies $\alpha_h(A) \neq \alpha_h(B)$,

³ The uniqueness is not important here.

since if $A \neq B$ and $\alpha_h(A) = \alpha_h(B)$ then:

$$\begin{aligned}
 false &= P_\pi(A, B) \\
 &= 1_v(P_\pi(A, B)) \\
 &= F_\pi(\alpha_h(A), \alpha_h(B)) \\
 &= F_\pi(\alpha_h(A), \alpha_h(A)) \\
 &= 1_v(P_\pi(A, A)) \\
 &= P_\pi(A, A) \\
 &= true.
 \end{aligned}$$

Consequently, $\alpha_h: \mathcal{P}(F_h) \rightarrow F_h$ is injective. But this is a contradiction, because $\mathcal{P}(F)$ always has a larger cardinal than F for any set F . \square

A third generalization of basic fixed-data hidden algebra is to allow only a subset of operations in Σ to be used in experiments [18,10,15,11]. Powerful proof techniques for behavioral satisfaction like coinduction are still sound, and the equational reasoning can be also adapted to this general framework. The interested reader is referred to the citations above.

2.2 Loose Data

There are approaches where the fixed data is not required, such as coherent hidden algebra [5,4], observational logic [13,1], and gcd hidden algebra [19]. We will show in the next section the incompleteness of an artificial restrictive framework briefly discussed below, called *basic loose-data hidden algebra*, which is a subcase⁴ of all these three approaches. Therefore, the three loose-data approaches above are also incomplete.

The basic loose-data hidden algebra logic differs from the basic fixed-data hidden algebra logic presented in the previous subsection in that the models are not required to have a fixed interpretation of the visible sort. The notion of “context” is exactly the same, and the “behavioral satisfaction” can be similarly defined for Σ -algebras as for hidden Σ -algebras.

Definition 2.6 *Given a behavioral specification $\mathcal{B} = (\Sigma, E)$ and a Σ -equation e , we write $\mathcal{B} \equiv^{ld} e$ whenever e is behaviorally satisfied by all Σ -algebras behaviorally satisfying E .*

Equational reasoning is also sound for this slightly different behavioral satisfaction [5], but not complete as we will shortly see. More precisely, we show that for some behavioral specifications $\mathcal{B} = (\Sigma, E)$, the following problem called $BSAT_{\mathcal{B}}^{ld}$:

⁴ Obtained by appropriate constraints on signatures.

INPUT: A Σ -equation e ;
 OUTPUT: $\mathcal{B} \equiv^{ld} e$?

is not recursively enumerable even for finite E . As a side point, this is achieved for a specification with only one visible constant, *true*. However, even *true* can be replaced by an attribute if the reader finds it inconvenient to have visible constants in the signature.

Definition 2.7 Given a Σ -equation e , say $(\forall x) t = t'$, let e^* be either the set $\{e\}$ if the sort of t and t' is visible, or the set of visible equations

$$\{(\forall x) c[t] = c[t'] \mid c \text{ is a context}\},$$

if the sort of t and t' is hidden. Given a set of equations E , let E^* be the set of visible equations $\bigcup_{e \in E} e^*$.

It is immediate from the definition of behavioral satisfaction that $A \equiv e$ is equivalent to $A \models e^*$. From this we get the following simple, but important fact:

Proposition 2.8 For any behavioral specification $\mathcal{B} = (\Sigma, E)$ and Σ -equation e , $\mathcal{B} \equiv^{ld} e$ iff $E^* \models e^*$.

Coherent hidden algebra extends this framework by allowing more visible and hidden sorts, the only restriction on operations being to have *at most* one hidden argument, and a subset of operations in Σ is allowed to define the behavioral equivalence, the “behavioral operations”. Observational logic and gcd hidden algebra extend it by allowing in addition even many-hidden-argument behavioral operations. Therefore, all these logics are incomplete.

3 Turing Machines and the Totality Problem

There are many equivalent definitions of Turing machines in the literature. We prefer one adapted from [20], and describe it informally in the sequel. The reader is assumed familiar with basics of Turing machines, the role of the following paragraphs being to establish our notations and conventions for the rest of the paper.

Consider a mechanical device which has associated with it a tape of infinite length in both directions, partitioned in spaces of equal size, called *cells*, which are able to hold either a “0” or an “1” and are rewritable. The device examines exactly one cell at any time, and can perform any of the following four operations (or *commands*):

- (i) Write an “1” in the current cell;
- (ii) Write a “0” in the current cell;
- (iii) Shift one cell to the right;
- (iv) Shift one cell to the left.

The device performs one operation per unit time, and this performance is called a *step*. Formally,

Definition 3.1 Let Q be a finite set of **internal states**, containing a **starting state** q_s and a **halting state** q_h . Let $B = \{0, 1\}$ be a set of **symbols** (or **bits**) and $C = \{0, 1, \rightarrow, \leftarrow\}$ be a set of **commands**. Then a (deterministic) Turing machine is a mapping from $Q \times B$ to $Q \times C$.

If the pair (q, b) is taken to (q', c) , then we sometimes write $(q, b) \rightarrow (q', c)$. We assume that the tape contains only 0's (or blanks) before the machine starts performing.

Definition 3.2 A **configuration** of a Turing machine is a triple consisting of an internal state and two infinite strings⁵, standing for the cells on the left and for the cells on the right, respectively. We let $(q, L|R)$ denote the configuration in which the machine is in state q , with left tape L and right tape R .

Given a configuration $(q, L|R)$, the content of the tape is LR , which is infinite at both ends. By convention, the current cell is the first cell of the right string. We also let $(q, L|R) \rightarrow (q', L'|R')$ denote the configuration transition under one of the four commands. Given a configuration in which the internal state is q and the examined cell contains b , and if $(q, b) \rightarrow (q', c)$, then exactly one of the following configuration transitions can take place:

- (i) $(q, L|bR) \rightarrow (q', L|cR)$ if $c = 0$ or $c = 1$;
- (ii) $(q, L|bR) \rightarrow (q', Lb|R)$ if $c = \rightarrow$;
- (iii) $(q, Lb'|bR) \rightarrow (q', L|b'bR)$ if $c = \leftarrow$.

The machine starts performing in the internal state q_s , so the initial configuration is $(q_s, \dots 0 \dots 0|0 \dots 0 \dots)$. Sometimes, we wish to run a Turing machine on a specific input, say $x = b_1 b_2 \dots b_n$. In this case, its initial configuration is $(q_s, \dots 0 \dots 0|b_1 b_2 \dots b_n 0 \dots 0 \dots)$.

Definition 3.3 A Turing machine **stops** when it first gets to its halting state, q_h .

Therefore, a Turing machine carries out a uniquely determined succession of steps, which may or may not terminate. It is well-known that Turing machines can compute exactly the partial recursive functions [20].

We claim that there are some Turing machines M , such that the following problem, called TOTALITY:

INPUT: An integer $k \geq 0$;

OUTPUT: Does M halt on all inputs $1^j 01^k$ for all $j \geq 0$?

is Π_2^0 -complete, where Π_2^0 is the class in the arithmetic hierarchy which extends both classes r.e. (recursively enumerable) and co-r.e., and contains predicates

⁵ Notice that the two infinite strings contain only 0's starting with a certain cell.

of the form $P(a) := (\forall x)(\exists y)R(a, x, y)$ where R is a primitive recursive predicate. It is obvious that **TOTALITY** is in Π_2^0 for any Turing machine M . To show that it is Π_2^0 -hard, we may choose M to be a universal Turing machine such that on input 1^j01^k , M computes $f_k(j)$, where f_k is the (partial) function computed by Turing machine with Gödel number k under some canonical assignment of Gödel numbers to Turing machines. By appropriately choosing conventions for Turing machines, $f_k(j)$ is defined if and only if the Turing machine numbered k halts on input j . Therefore, **TOTALITY**(k) has positive solution if and only if the function f_k is total. But the set $\{k \mid f_k \text{ is total}\}$ is Π_2^0 -complete [20]. It follows that **TOTALITY** is Π_2^0 -complete.

We henceforth fix some choice of M that makes the **TOTALITY** problem Π_2^0 -complete.

4 Behavioral Satisfaction is Π_2^0 -Hard

In this section we show that all versions of behavioral satisfaction discussed so far are Π_2^0 -hard, so in particular, the associated logics do not admit complete deduction systems.

The strategy used is the expected one: reduction from a Π_2^0 -complete problem to behavioral satisfaction. We chose **TOTALITY**, for a (fixed) Turing machine M which makes it Π_2^0 -complete. Next we define a behavioral specification like in Section 2. Let

- Σ be the following hidden signature:
 - v, h are a visible and a hidden sort, respectively;
 - $true, false$ are two visible constants⁶
- Attributes:
 - $WillStop: h \rightarrow v$;
- Methods:
 - $blank: h \rightarrow h$;
 - $q: h \rightarrow h$ for each state $q \in Q$;
 - $0_l, 1_l, 0_r,$ and 1_r , all of the form $h \rightarrow h$;
 - $always: h \rightarrow h$;
 - $More: h \rightarrow h$;
- E be the finite set of equations:
 - (1) $(\forall x) blank(m(x)) = blank(x)$ for all methods m ;
 - (2) $(\forall x) m(always(x)) = always(x)$ for all methods $m \neq blank$;
 - (3) $(\forall x) m(q(x)) = always(x)$ for any method $m \neq blank$ and state $q \in Q$, such that $m \neq More$ or $q \neq q_s$;
 - (4) $(\forall x) More(q_s(x)) = q_s(1_r(x))$;
 - (5) $(\forall x) b'_l(b_r(x)) = b_r(b'_l(x))$ for all $b, b' \in \{0, 1\}$;
 - (6) $(\forall x) 0_d(blank(x)) = blank(x)$ for any $d \in \{l, r\}$;
 - (7) $(\forall x) WillStop(always(x)) = true$;

⁶ In fact, only $true$ is needed for the loose-data case.

$$(8) (\forall x) WillStop(q_h(x)) = true;$$

Now, for every transition $(q, b) \rightarrow (q', c)$, add:

- (9a) $(\forall x) WillStop(q(b_r(x))) = WillStop(q'(c_r(x)))$ if $c = 0$ or $c = 1$,
- (9b) $(\forall x) WillStop(q(b_r(x))) = WillStop(q'(b_l(x)))$ if $c = \rightarrow$,
- (9c) $(\forall x) WillStop(q(b'_l(b_r(x)))) = WillStop(q'(b'_r(b_r(x))))$ for $b' = 0$ and $b' = 1$,
if $c = \leftarrow$.

Let \mathcal{B} be the behavioral specification (Σ, E) . Notice that \mathcal{B} is finite; more precisely, the number of operations in Σ is $O(|Q|)$, and the number of equations in E is $O(|Q|^2)$, where $|Q|$ is the number of internal states of M .

Lemma 4.1 *If M stops on an input $b_1 b_2 \dots b_n$, then*

$$B \equiv^{ld} (\forall x) WillStop(q_s(b_{1r}(b_{2r}(\dots(b_{nr}(blank(x))\dots)))) = true.$$

Proof. It can be easily seen that every configuration transition in M can be simulated by equational deduction using the equations (9a), (9b), and/or (9c); notice that the equation (5) may be needed to bring the operations b_r on top of the terms as arguments of q operations in order to be allowed to use the equations (9a), (9b), and (9c), and that the equation (6) can be used to generate more 0_l and 0_r operations when needed. Iterating this procedure, we get that \mathcal{B} satisfies the equation

$$(\forall x) WillStop(q_s(b_{1r}(b_{2r}(\dots(b_{nr}(blank(x))\dots)))) = WillStop(q_h(t(blank(x))))$$

for some appropriate sequence t of methods $b_d \in \{0_l, 1_l, 0_r, 1_r\}$. The rest follows by equation (8). \square

Let \mathcal{M} be the following Σ -algebra:

- $\mathcal{M}_v = \{true, false\}$;
- $\mathcal{M}_h = ((Q \cup \perp) \times String \times String) \cup \square$ where \square is a special new element;
- $\mathcal{M}_{WillStop}(\square) = true$ for any strings S and S' ;
- $\mathcal{M}_{WillStop}((\perp, S, S')) = true$;
- $\mathcal{M}_{WillStop}((q, S, S')) = true$ iff the Turing machine M halts when starts with the state $(q, \dots 0 \dots 0 S | S' 0 \dots 0 \dots)$; otherwise it is *false*;
- $\mathcal{M}_{blank}(X) = (\perp, \epsilon, \epsilon)$ for any element X in \mathcal{M}_h ;
- $\mathcal{M}_q(\square) = \square$ for any $q \in Q$;
- $\mathcal{M}_q((\perp, S, S')) = (q, S, S')$ for any $q \in Q$ and any strings S, S' ;
- $\mathcal{M}_q((q', S, S')) = \square$ for any $q, q' \in Q$ and any strings S, S' ;
- $\mathcal{M}_{b_d}(\square) = \square$ for any $b_d \in \{0_l, 1_l, 0_r, 1_r\}$;
- $\mathcal{M}_{b_d}((q, S, S')) = \square$ for any $q \in Q$, strings S, S' , and $b_d \in \{0_l, 1_l, 0_r, 1_r\}$;
- $\mathcal{M}_{b_l}((\perp, S, S')) = (\perp, Sb, S')$ for any $b \in \{0, 1\}$ and strings S, S' , such that $b \neq 0$ or $S \neq \epsilon$;

- $\mathcal{M}_{0_l}((\perp, \epsilon, S')) = (\perp, \epsilon, S')$ for any string S' ;
- $\mathcal{M}_{b_r}((\perp, S, S')) = (\perp, S, bS')$ for any $b \in \{0, 1\}$ and strings S, S' , such that $b \neq 0$ or $S' \neq \epsilon$;
- $\mathcal{M}_{0_r}((\perp, S, \epsilon)) = (\perp, S, \epsilon)$ for any string S ;
- $\mathcal{M}_{always}(X) = \square$;
- $\mathcal{M}_{More}(\square) = \square$;
- $\mathcal{M}_{More}((\perp, S, S')) = \square$ for all strings S, S' ;
- $\mathcal{M}_{More}((q, S, S')) = \square$ for all $q \in Q \Leftrightarrow \{q_s\}$ and strings S, S' ;
- $\mathcal{M}_{More}((q_s, S, S')) = (q_s, S, 1S')$ for all strings S, S' ;

Proposition 4.2 \mathcal{M} is a hidden Σ -algebra which behaviorally satisfies \mathcal{B} .

Proposition 4.3 Given $k \geq 0$, let e_k be the equation

$$(\forall x) q_s(0_r(1_r(1_r(\dots(1_r(blank(x))\dots)))))) = always(x),$$

where the method 1_r occurs k times. Then the following are equivalent:

- (i) $\text{TOTALITY}(k)$ is positive;
- (ii) $\mathcal{B} \equiv^{ld} e_k$;
- (iii) $\mathcal{B} \equiv^{fd} e_k$.

Proof. (i) \Rightarrow (ii). Let s_k be the term on the left-hand side of e_k . Suppose that there exists a context c such that \mathcal{B} does not (loosely) behaviorally satisfy the equation $(\forall x) c[s_k] = c[always(x)]$. Clearly, c must be a term of the form $WillStop(m_1(m_2(\dots(m_j(\star))\dots)))$, where m_1, m_2, \dots, m_j are methods.

If $blank$ is among m_1, m_2, \dots, m_j , then by iteratively using the equation (1), $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = c[always(x)]$, contradiction. Therefore, $blank$ does not occur in c . If $always$ is in c , then by (2), $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = WillStop(always(t))$ and $\mathcal{B} \equiv^{ld} (\forall x) c[always(x)] = WillStop(always(t))$ for some appropriate term t of sort h . Therefore, $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = c[always(x)]$, contradiction. Therefore, $always$ does not occur in c . By (2) and (7), one can immediately see that $\mathcal{B} \equiv^{ld} (\forall x) c[always(x)] = true$. If $m_j \neq More$, then by the equations (3), (2), and (7), $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = true$, contradiction. Therefore, m_j is $More$. For $i \leq j$, let s_k^i be $q_s(1_r(1_r(\dots(1_r(0_r(1_r(1_r(\dots(1_r(blank(x))\dots))))))))$, with $j \Leftrightarrow i$ occurrences of the operation 1_r , followed by an operation 0_r , and followed by k operations 1_r ; notice that $s_k^j = s_k$. If there is an index $i < j$ such that $m_i \neq More$ and m_{i+1}, \dots, m_j are all $More$, then by equation (4), $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = c_i[s_k^i]$, where c_i is $WillStop(m_1(m_2(\dots(m_i(\star))\dots)))$. Since $m_i \neq More$, by equations (3), (2), and (7) as above, $\mathcal{B} \equiv^{ld} (\forall x) c_i[s_k^i] = true$, that is, $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = true$, contradiction.

Therefore, c must have the form $WillStop(More(More(\dots(More(\star))\dots)))$, for $j \geq 0$ occurrences of $More$. Then by (4), $\mathcal{B} \equiv^{ld} (\forall x) c[s_k] = WillStop[s_k^0]$. Since $\text{TOTALITY}(k)$ is positive, the Turing machine M stops for any input $1^j 0 1^k$

with $j \geq 0$. Then by Lemma 4.1, $\mathcal{B} \models^{ld} (\forall x) WillStop[s_k^0] = true$. Consequently, $\mathcal{B} \models^{ld} (\forall x) c[s_k] = c[always(x)]$, contradiction again.

Hence, \mathcal{B} satisfies the equation $(\forall x) c[s_k] = c[always(x)]$ for any context c , that is, $\mathcal{B} \models^{ld} e_k$.

(ii) \Rightarrow (iii). Obvious, since any hidden Σ -algebra which is a model of \mathcal{B} is also a loose model of \mathcal{B} , with the same behavioral equivalence on it.

(iii) \Rightarrow (i). If $\mathcal{B} \models^{fd} e_k$ then by equational reasoning as above, \mathcal{B} behaviorally satisfies $(\forall x) WillStop[s_k^0] = true$ for all $j \geq 0$. By Proposition 4.2, \mathcal{M} satisfies the equation $(\forall x) WillStop[s_k^0] = true$, that is, $\mathcal{M}_{WillStop}((q_s, \epsilon, 1^j 01^k))$ is *true*, which means that the Turing machine M terminates on the input $1^j 01^k$. Hence, the answer of $TOTALITY(k)$ is positive. \square

The following is the main result of the paper, showing that the behavioral satisfaction and non-satisfaction problems cannot be mechanized in any algorithmical way.

Theorem 4.4 *In all versions of behavioral logics mentioned, there are behavioral specifications for which the behavioral satisfaction problem is Π_2^0 -hard. Therefore, these logics do not have complete deduction systems. Moreover, none of these logics admits algorithms to refute false statements.*

Proof. Since the problem $TOTALITY$ is Π_2^0 -complete, then by Proposition 4.3, $BSAT_{\mathcal{B}}^{fd}$ and $BSAT_{\mathcal{B}}^{ld}$ are Π_2^0 -hard. Since the other versions of behavioral logics are generalizations of either basic fixed-data hidden algebra or basic loose-data hidden algebra, they are also Π_2^0 -hard. Since the class Π_2^0 strictly includes both the set of r.e. predicates and the set of co-r.e. predicates, the behavioral satisfaction and behavioral non-satisfaction problems in all the mentioned logics are not r.e., so there is no algorithm to prove a true statement or to refute a false statement in behavioral logics. \square

5 Some Behavioral Logics are Π_2^0 -Complete

Once we know that all behavioral logics mentioned are Π_2^0 -hard, a natural next step is to find a place for these logics in the arithmetic hierarchy, if such a place exists. In this section, we show that some behavioral logics are Π_2^0 -complete, and in the next section we show that some fixed-data versions of hidden algebra are not even in the arithmetic hierarchy, they being Π_1^1 -hard. Our work in this and the next sections should be viewed only as a starting point toward stronger characterization results of behavioral logics.

As opposed to the previous sections, our goal now is to find as general behavioral logics as possible which are in the class Π_2^0 , since this would imply that all behavioral logics obtained as special instances of them are also in Π_2^0 .

The general idea is to choose those logics admitting complete deduction for visible equations. If this is the case, then the behavioral satisfaction problem

becomes of the form:

INPUT: A behavioral specification $\mathcal{B} = (\Sigma, E)$ and a Σ -equation $(\forall X) t = t'$;
 OUTPUT: Is it the case that for every context c , there exists a proof p , such that
 p proves $(\forall X) c[t] = c[t']$ in \mathcal{B} ?

which is a Π_2^0 statement.

5.1 Fixed Data

We analyze two versions of fixed-data behavioral logics which are Π_2^0 -complete, both special cases of hidden algebra [6,7,9,8].

The first behavioral logic we present considers that the data algebra is finite, and for this reason we call it *finite fixed-data hidden algebra*. It obviously includes the restrictive basic fixed-data hidden algebra logic we presented in Section 2.

Proposition 5.1 *If \mathcal{B} is any behavioral specification in the sense of finite fixed-data hidden algebra, then $BSAT_{\mathcal{B}}^{fd}$ is in Π_2^0 .*

Proof. Let φ be a first-order sentence that completely characterizes the finite data algebra D up to isomorphism. Then $\mathcal{B} \models^{fd} e$ holds if and only if $\{\varphi, E^*\} \vdash e^*$ (see Definition 2.7) where \vdash means provability in first-order logic with equality. This is a Π_2^0 statement. \square

Now we present another behavioral logic, which we call *flat fixed-data hidden algebra*, which is a special case of hidden algebra logic that generalizes Corradini’s coalgebraic equational logic [3].

Definition 5.2 *A **hidden signature** is a $V \cup H$ -sorted signature such that each operation either has exactly one argument of sort in H and zero or more arguments of sorts in V , or is a constant from a fixed set of constants D which is a recursively enumerable⁷ set, called **the data**. The sorts in V are called **visible** and the sorts in H are called **hidden**.*

Corradini’s approach is slightly more restrictive, in the sense that it does not admit visible sorts as arguments of operations. On the other hand, hidden algebra is more general, as it allows operations with visible arguments, hidden constants, and the data can be any visible-sorted algebra (as opposed to just a flat set).

The notions of behavioral specification and satisfaction defined in Section 2 easily translate to this slightly more general framework. The contexts are now visible terms having exactly one occurrence of a special hidden variable $\{\star\}$, such that all their visible proper subterms are constants in D .

Corradini’s deduction system, generating a relation \vdash^C , and its completeness theorem in [3] can be easily generalized to operations allowing visible parameters, thus obtaining the following:

⁷ D can be any set in [3].

Theorem 5.3 *Given a behavioral specification $\mathcal{B} = (\Sigma, E)$ and above and a Σ -equation of visible sort e , then $\mathcal{B} \models^{fd} e$ if and only if $E \vdash^C e$. Therefore, $BSAT_{\mathcal{B}}^{fd}$ is in Π_2^0 .*

Notice that Corradini’s framework allows only visible equations in E , but this is not an inconvenience since E can be replaced by an r.e. set of visible equations E^* (see Definition 2.7). In order to make this operation part of the deduction systems, we need to add a congruence inference rule to those in [3].

5.2 Loose Data

We claim that all versions of loose-data behavioral satisfactions for which the equational reasoning is sound are in Π_2^0 . These include observational (equational) logic, and coherent and gcd hidden algebra only for the case in which all operations are congruent [5,19]. The reason is that Proposition 2.8 is applicable, reducing behavioral satisfaction to equational standard satisfaction. We always assume that the set E of equations is r.e.

Proposition 5.4 *$BSAT_{\mathcal{B}}^{ld}$ is in Π_2^0 for any \mathcal{B} as above.*

Proof. Let $\mathcal{B} = (\Sigma, E)$ be any loose-data behavioral specification for which the equational reasoning is sound, and e be any Σ -equation. By Proposition 2.8, $\mathcal{B} \models^{ld} e$ iff $E^* \models e^*$, and by equational completeness theorem, iff $E^* \vdash e^*$, where now \vdash is the equational derivation relation. Therefore, $BSAT_{\mathcal{B}}^{ld}$ is in Π_2^0 . \square

6 Others are not in the Arithmetic Hierarchy

We now consider the complexity of behavioral satisfaction for hidden algebra in general, where the data can be any infinite algebra. We saw that for finite data algebras, there is a single first-order sentence that fully characterizes the data algebra and so the satisfaction problem is still in Π_2^0 . For infinite flat data, with no operations on it, we also saw that the Corradini’s completeness theorem is applicable and so the satisfaction problem is also in Π_2^0 .

Now, for infinite data algebras, we assume the fixed data algebra is arithmetically definable, and we shall see that in this case the satisfaction problem is in Π_1^1 . Recall Π_1^1 is the first level of the analytic hierarchy [20], and thus properly contains the arithmetic hierarchy. An elementary canonical representation of the class Π_1^1 is obtained as follows. Let $f : \mathbb{N} \rightarrow \mathbb{N}$. We define $\overline{f}(n)$ to be the sequence $\langle f(0), f(1), \dots, f(n \Leftrightarrow 1) \rangle$. Then a predicate $A(x)$ is in Π_1^1 if and only if it can be expressed in the form

$$(1) \quad A(x) := (\forall f : \mathbb{N} \rightarrow \mathbb{N})(\exists y)T(x, y, \overline{f}(y)),$$

where T is some primitive recursive predicate.

Proposition 6.1 *Let us consider the general fixed-data hidden algebra logic where Σ is enumerable, E is arithmetic, and D is a fixed enumerable infinite*

data algebra with an arithmetic representation. Then the behavioral satisfaction problem is in Π_1^1 .

Proof. Any model of \mathcal{B} can be encoded by a function $f: \mathbb{N} \rightarrow \mathbb{N}$. Thus, $\mathcal{B} \models e$ holds if and only if every $f: \mathbb{N} \rightarrow \mathbb{N}$ which codes a valid model of \mathcal{B} satisfies the equations e^* . Since equational validity for the model defined by f is arithmetic, the condition on f is arithmetic. Therefore $\mathcal{B} \models e$ is Π_1^1 . \square

Now we show that there are natural fixed data algebras for which the behavioral satisfaction problem is Π_1^1 -hard. Let the data algebra D be $(\mathbb{N}, 0, S)$, the set of integers with the successor function. Almost any infinite data algebra effectively contains D of course; further, D is well-known to have a decidable first-order theory.

Theorem 6.2 *There is a finite set of equations E over a finite signature and a family of equations e_n , $n = 0, 1, 2, \dots$, such that the behavioral satisfaction problem $\mathcal{B} \models e_n$ is Π_1^1 -complete.*

Proof. Fix some Π_1^1 -complete predicate $A(x)$ expressed in the form (1). We construct a hidden signature and hidden equations and give a many-one reduction from the predicate $A(x)$ to the behavioral satisfiability problem.

In addition, to the visible sort of \mathbb{N} , we let our hidden signature have a single hidden sort. This hidden sort will act only vacuously as a source of dummy variables — we use h to denote a variable of hidden sort.

The language includes a finite list of function symbols which have one hidden input and several visible inputs. These function symbols represent primitive recursive functions augmented with an extra, ignored, hidden input. For example, the functions corresponding to addition and multiplication are $Add(h, m, n)$ and $Mult(h, m, n)$ and the set of equations contains their defining equations, namely,

$$\begin{aligned} Add(h, x, 0) &= x \\ Add(h, x, S(y)) &= S(Add(h, x, y)) \\ Mult(h, x, 0) &= 0 \\ Mult(h, x, S(y)) &= Add(h, Mult(h, x, y), x) \end{aligned}$$

The *Not* function, which implements negation if one thinks of zero as denoting truth and non-zero as denoting falsity, is defined by

$$\begin{aligned} Not(h, 0) &= S(0) \\ Not(h, S(y)) &= 0 \end{aligned}$$

In a similar fashion, we include the primitive recursive functions for coding sequences: $\langle \rangle$ is the code for the empty sequence, and $Concat(h, w, a)$ is the operation for appending an integer a to the end of a sequence w . We also include enough primitive recursive functions in the language so as to include the function p_T defined by

$$p_T(h, x, y, z) = 0 \text{ iff } T(x, y, z).$$

Finally, we add function symbols $f(h, x, y)$ and $g(h, x, y)$ and equations that make $g(h, x, y)$ equal to $\overline{f}(h, x, y)$, namely,

$$\begin{aligned} g(h, x, 0) &= \langle \rangle \\ g(h, x, S(y)) &= \text{Concat}(h, g(h, x, y), f(h, x, y)). \end{aligned}$$

We do not add any equations restricting the function f .

One final function symbol is needed: $a(h, x)$ which is intended to equal 0 if and only if f witnesses the truth of $A(x)$. For this we add an equation

$$\text{Mult}(h, \text{Not}(h, g(h, x, y)), a(h, x)) = 0$$

which enforces the condition

$$g(h, x, y) = 0 \rightarrow a(h, x) = 0.$$

We take as the set E all the above equations, and let \mathcal{B} be (Σ_D, E) . For $n \geq 0$, let \underline{n} denote the term $S^n(0)$ with value n . Let e_n be the equation $a(h, \underline{n}) = 0$. Then it is straightforward to verify that

$$A(n) \text{ is true} \Leftrightarrow \mathcal{B} \models e_n.$$

□

It is an interesting observation that the above proof used an equational system with no “methods”, but only “attributes.” This implies that it is not really a result about behavioral logics, but is instead a result about equational logic over a fixed algebra. Indeed, our proof can be recast as showing that in ordinary equational logic, it can be Π_1^1 -hard to decide equational satisfaction in ω -models. Moreover, this technique can be used to transfer any hardness result concerning ordinary equational logic over a fixed domain D into a hardness result for hidden algebra over the fixed data algebra D .

7 Conclusion and Future Work

We showed that for some behavioral specifications in any behavioral logic currently in use, the satisfaction problem is Π_2^0 -hard. Since the class Π_2^0 properly includes both the classes of r.e. problems and co-r.e. problems, our result has two major implications. The first one is that there is *no* algorithm to prove true statements, in particular the behavioral logics are incomplete. The second is that there is *no* algorithm to reject false statements.

Then we showed the Π_2^0 -completeness for the finite fixed-data and flat fixed-data hidden algebra logics, respectively, and also for those loose-data behavioral logics for which the equational reasoning is sound. The behavioral satisfaction problem was shown to be Π_1^1 -complete in some cases of fixed-data hidden algebra logics over infinite data algebras.

There are still many cases of behavioral logics which can be derived from the two basic logics we presented in Section 2, which we did not analyze in this paper. For example, what is the upper bound on the complexity of behavioral satisfaction in loose-data behavioral logics in which equational reasoning is not

sound, such as coherent hidden algebra and gcd hidden algebra? How about hidden algebra logic over infinite data algebra but with all operations having exactly one hidden argument and no visible arguments? How important is the restriction to not allow operations having more than one hidden argument?

Acknowledgments: We would like to thank Joseph Goguen for his comments on various versions of this work and for his continuous belief that the behavioral logics are incomplete. Special thanks to Ugo Montanari, Andrea Corradini, and Bogdan Warinschi for various technical discussions related to the work in this paper.

References

- [1] Michel Bidoit and Rolf Hennicker. Observer complete definitions are behaviourally coherent. In Kokichi Futatsugi, Joseph Goguen, and José Meseguer, editors, *OBJ/CafeOBJ/Maude at Formal Methods '99*, pages 83–94. Theta, 1999. Proceedings of a workshop held in Toulouse, France, 20th and 22nd September 1999.
- [2] Corina Cîrstea. A coequational approach to specifying behaviours. In Bart Jacobs and Jan Rutten, editors, *Proceedings of the Second Workshop on Coalgebraic Methods in Computer Science (CMCS'99), Amsterdam, The Netherlands, March 1999*, volume 19 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 1999.
- [3] Andrea Corradini. A complete calculus for equational deduction in coalgebraic specification. Technical Report SEN-R9723, ISSN 1386-396X, CWI, 1997.
- [4] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*. World Scientific, 1998. AMAST Series in Computing, volume 6.
- [5] Răzvan Diaconescu and Kokichi Futatsugi. Behavioral coherence in object-oriented algebraic specification. *Journal of Universal Computer Science*, 6(1):74–96, 2000. Also Japan Advanced Institute for Science and Technology Technical Report number IS-RR-98-0017F, 1998.
- [6] Joseph Goguen. Types as theories. In George Michael Reed, Andrew William Roscoe, and Ralph F. Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford, 1991. Proceedings of a Conference held at Oxford, June 1989.
- [7] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Hartmut Ehrig and Fernando Orejas, editors, *Proceedings, Tenth Workshop on Abstract Data Types*, pages 1–29. Springer, 1994. Lecture Notes in Computer Science, Volume 785.
- [8] Joseph Goguen and Grant Malcolm. Hidden coinduction: Behavioral correctness proofs for objects. *Mathematical Structures in Computer Science*, 9(3):287–319, 1999.

- [9] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, to appear. Also UCSD Dept. Computer Science & Eng. Technical Report CS97-538, May 1997.
- [10] Joseph Goguen and Grigore Roșu. Hiding more of hidden algebra. In *FM'99 – Formal Methods*, pages 1704–1719. Springer, 1999. Lecture Notes in Computer Sciences, Volume 1709, Proceedings of World Congress on Formal Methods, Toulouse, France.
- [11] Joseph Goguen and Grigore Roșu. A protocol for distributed cooperative work. In Gheorghe Ștefănescu, editor, *Proceedings of Workshop on Distributed Systems 1999 (WDS'99), Iasi, Romania, 2 September 1999*, volume 28 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 1999.
- [12] H. Peter Gumm and Tobias Schröder. Covarieties and complete covarieties. In Bart Jacobs, Larry Moss, Horst Reichel, and Jan Rutten, editors, *Proceedings of the First Workshop on Coalgebraic Methods in Computer Science (CMCS'98), Lisbon, Portugal, March 1998*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 43–56. Elsevier Science, 1998.
- [13] Rolf Hennicker and Michel Bidoit. Observational logic. In *Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1999.
- [14] Grigore Roșu. A Birkhoff-like axiomatizability result for hidden algebra and coalgebra. In Bart Jacobs, Larry Moss, Horst Reichel, and Jan Rutten, editors, *Proceedings of the First Workshop on Coalgebraic Methods in Computer Science (CMCS'98), Lisbon, Portugal, March 1998*, volume 11 of *Electronic Notes in Theoretical Computer Science*, pages 179–196. Elsevier Science, 1998.
- [15] Grigore Roșu. Behavioral coinductive rewriting. In Kokichi Futatsugi, Joseph Goguen, and José Meseguer, editors, *OBJ/CafeOBJ/Maude at Formal Methods '99*, pages 179–196. Theta, 1999. Proceedings of a workshop held in Toulouse, France, 20th and 22nd September 1999.
- [16] Grigore Roșu. Equational axiomatizability for coalgebra. *Theoretical Computer Science*, to appear, 2000.
- [17] Grigore Roșu and Joseph Goguen. Hidden congruent deduction. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, pages 252–267. Springer, 2000. Lecture Notes in Artificial Intelligence, Volume 1761; papers from a conference held in Vienna, November 1998.
- [18] Grigore Roșu and Joseph Goguen. Hidden congruent deduction. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, pages 252–267. Springer, 2000. Lecture Notes in Artificial Intelligence, Volume 1761; papers from a conference held in Vienna, November 1998.

- [19] Grigore Roşu and Joseph Goguen. Circular coinduction. Technical Report CSE2000–0647, University of California at San Diego, February 2000. Written October 1999.
- [20] Hartley Rogers Jr. *Theory of Recursive Functions and Effective Computability*. MIT press, Cambridge, MA, 1987.
- [21] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI, 1996.