Under consideration for publication in J. Functional Programming

On the Complexity of Stream Equality

Jörg Endrullis, Dimitri Hendriks, Rena Bakhshi
VU University Amsterdam, The Netherlands
and Grigore Roşu
University of Illinois at Urbana-Champaign, USA
University Alexandru Ioan Cuza, Iaşi, Romania

Abstract

We study the complexity of deciding the equality of streams specified by systems of equations. There are several notions of stream models in the literature, each generating a different semantics of stream equality. We pinpoint the complexity of each of these notions in the arithmetical or analytical hierarchy. Their complexity ranges from low levels of the arithmetical hierarchy such as Π^0_2 for the most relaxed stream models, to levels of the analytical hierarchy such as Π^1_1 and up to subsuming the entire analytical hierarchy for more restrictive but natural stream models. Since all these classes properly include both the semi-decidable and co-semi-decidable classes, it follows that regardless of the stream semantics employed, there is no complete proof system or algorithm for determining equality or inequality of streams. We also discuss several related problems, such as the existence and uniqueness of stream solutions for systems of equations, as well as the equality of such solutions.

Contents

1 Introduction

In functional programming, the use of infinite data structures dates back to (Landin, 1965; Henderson & Morris, Jr., 1976; Friedman & Wise, 1976). More recently, also in other branches of computer science, interest has grown towards infinite data, as witnessed by the application of type theory to infinite objects (Coquand, 1993), the emergence of coalgebraic techniques for infinite data types (Aczel, 1988; Rutten, 2003), infinitary term rewriting (Dershowitz *et al.*, 1991; Endrullis *et al.*, 2012a) and infinitary lambda calculus (Böhm, 1975; Kennaway *et al.*, 1997).

One of the simplest examples of an infinite data-structure is the *stream*, specifically the stream of bits $0,1$. Streams can be equivalently regarded as functions on natural numbers in a trivial way, by associating to each natural number n the element on the n-th position in the stream. Since the equality of functions on natural numbers is an arbitrarily complex problem, so is the equality of streams in general. However, there is a relatively broad interest in streams defined in a particular but intuitive and meaningful way, namely equationally. For example, the usual zeros and ones streams containing only 0 and 1 bits, respectively, as well as a blink stream of alternating 0 and 1 bits and the zip binary operation on streams, can be defined equationally as follows:

$$\left.\begin{aligned} \mathsf{zeros} &= 0 : \mathsf{zeros} \\ \mathsf{ones} &= 1 : \mathsf{ones} \\ \mathsf{blink} &= 0 : 1 : \mathsf{blink} \\ \mathsf{zip}(a : x, y) &= a : \mathsf{zip}(y, x) \end{aligned}\right\} \tag{1}$$

The equational specification of streams and other infinite objects is common practice in coalgebra, term rewriting and functional programming.

There is an obvious connection between equational stream specifications and functional programs computing lazy lists. Many programming or specification languages with support for lazy evaluation or rewriting, allow for streams defined equationally as above; for example Haskell (Peyton-Jones, 2003)[1], Miranda (Turner, 1986), Clean (Sleep *et al.*, 1993), or Maude (Clavel *et al.*, 2003). In the programming community there is also a long-standing tradition of employing stream-like abstractions, for example, for stream I/O (Landin, 1965).

The set of streams can be formally defined in many different, but ultimately equivalent ways; e.g., as a coinductive type (Geuvers, 1992), as a final coalgebra (Rutten, 2005), as an

[1] Haskell provides a built-in $\mathsf{zip} :: [a] \to [b] \to [(a,b)]$, but here we prefer to use ‘zip’ for the interlacing of lists, as defined by the equation in (1). The operation is also known as *perfect shuffle*.

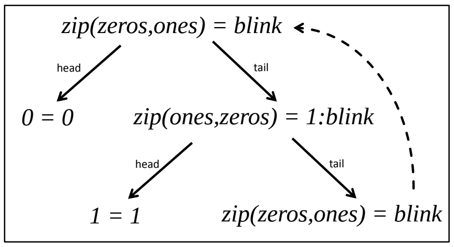

Fig. 1. Intuitive proof by circular coinduction of $\mathsf{zip}(\mathsf{zeros},\mathsf{ones}) = \mathsf{blink}$

observational specification (Bidoit *et al.*, 2003) or as a hidden logic theory (Roşu, 2000). All these approaches build upon the observation that streams cannot be defined using conventional equational specifications; one reason for this is that equational specifications allow too many models, making it impossible to prove many interesting properties of streams. Consider, for example, infinite streams of bits together with their usual constructor : and together with the streams and stream operations defined equationally above. Then note that the expected properties

$$\begin{aligned} \mathsf{zip}(\mathsf{zeros},\mathsf{zeros}) &= \mathsf{zeros}\ , \\ \mathsf{zip}(\mathsf{ones},\mathsf{ones}) &= \mathsf{ones}\ , \\ \mathsf{zip}(\mathsf{zeros},\mathsf{ones}) &= \mathsf{blink} \end{aligned}$$

cannot be proved equationally, not even by making use of induction, because they actually do *not* hold in the initial model of the equations above. Indeed, the initial model (Goguen *et al.*, 1977) has as elements equivalence classes of provably equal ground terms, such as $\{\mathsf{zeros}, 0 : \mathsf{zeros}, 0 : 0 : \mathsf{zeros}, \dots\}$ or $\{\mathsf{zip}(\mathsf{zeros},\mathsf{zeros}), 0 : \mathsf{zip}(\mathsf{zeros},\mathsf{zeros}), 0 : 0 : \mathsf{zip}(\mathsf{zeros},\mathsf{zeros}), \dots\}$, but there is nothing in the ordinary equational setting that can collapse these two equivalence classes into one, which is what would be needed in order for the first equation above to hold in the initial model, for example.

There are several approaches to proving streams equal, such as *coinduction* in a coalgebraic equational specification of streams (Rutten, 2005), *context induction* (Hennicker, 1991) in an observational equational framework, *guarded coinduction* (Coquand, 1993) in a higher-order typed framework, or *circular coinduction* (Roşu, 2000) in a hidden logic framework; the first three typically need human support, while the latter is automatic. By circular coinduction, for example, we can prove $\mathsf{zip}(\mathsf{zeros},\mathsf{ones}) = \mathsf{blink}$ as follows:

(i) check that the two streams have the same head, 0;
(ii) take the tail of the two streams and generate new goal $\mathsf{zip}(\mathsf{ones},\mathsf{zeros}) = 1 : \mathsf{blink}$; this becomes the next task;
(iii) check that the two new streams have the same head, 1;
(iv) take the tail of the two new streams; after simplification one gets the new goal $\mathsf{zip}(\mathsf{zeros},\mathsf{ones}) = \mathsf{blink}$, which is nothing but the original proof task;
(v) conclude that $\mathsf{zip}(\mathsf{zeros},\mathsf{ones}) = \mathsf{blink}$ holds.

The intuition for the above "proof", depicted in Fig. 1, is that the two streams have been exhaustively tried to be distinguished by iteratively checking their heads and taking their tails. Ending up in circles (we obtained the same new proof task as the original one) means that the two streams are indistinguishable, so equal. There are tools that attempt to automate proving stream equality, such as (Lucanu *et al.*, 2009; Zantema & Endrullis, 2011), as well as methods for mechanically proving stream equality in assisted theorem provers like Coq (Niqui, 2009; Danielsson, 2010; Endrullis *et al.*, 2013). Since some techniques can show many challenging equalities of streams and seem not to fail even on large and tricky examples, one may be (wrongly) tempted to prove them complete; by a complete technique in this context we mean one which answers "yes" on precisely the inputs consisting of pairs of equal streams—on the others it may either not terminate, or terminate with an output different from "yes". Also, since equational logic is complete, one may (wrongly) think that streams defined equationally must also admit some complete proof system. Moreover, since two different streams must differ on some position of finite index, one could (also wrongly) think that at least one can detect when two streams are not equal.

In this paper we study the precise complexity of deciding the equality of streams specified equationally like above. In order to address such a problem we have to first make a choice for what it means for two streams to be equal, or in other words to make a choice for a semantics of streams and of their equality. A careful analysis of the vast literature on streams and related topics reveals that there are several meaningful candidate semantics, which essentially differ in their choice of stream *models*, that is, the semantics used to interpret specifications. Although the differences between the various models may appear to be minor and irrelevant at first sight, we show that in fact they have a crucial effect on the complexity of the stream equality problem. As one may expect, the more relaxed the stream models, the lower the complexity of the equality problem. We show that even with the most relaxed models that we are aware of, the problem is still Π^0_2. Recall that Π^0_2 is the class in the arithmetic hierarchy which properly extends both the semi-decidable and the co-semi-decidable classes, and contains predicates (over natural numbers) of the form $P(a) := \forall x \in \mathbb{N}. \exists y \in \mathbb{N}. R(a,x,y)$ where R is a decidable predicate (Rogers, Jr., 1967). In other words, the results in this paper tell us that there is actually *no* algorithm or proof system that is complete for equality of streams in general, so the problem is strictly harder than that of proving equality in equational or first-order logics, as well as *no* algorithm or proof system that is complete for inequality of streams.

Before we discuss in more detail the various semantics that we use in the sequel, we mention that all of the complexity results we obtain, as well as those in (Buss & Roşu, 2000; Roşu, 2006; Endrullis *et al.*, 2012b), originate from *non-productive* specifications. A specification is *productive* if its terms can be evaluated to infinite constructor terms in the limit. For a stream term this means that its infinite normal forms looks like $a_0 : a_1 : a_2 : \ldots$. See (Sijtsma, 1989; Endrullis *et al.*, 2008; Endrullis *et al.*, 2010a) for a precise definition of productivity. Productivity itself is undecidable, namely Π^0_2-complete, as shown in (Endrullis *et al.*, 2009).

The complexity results in this paper are all based on the comparison of non-productive specifications. The proofs inherently encode productivity problems, exploiting the choice of freedom the models have for giving semantics to undefined (parts of) objects.

Indeed, equality of normal forms for productive specifications is a suitable candidate for the semantics of equivalence of specifications. However, even if we restrict to productive specifications (and thus separate the problem of productivity from that of equality), the problem of stream equality is undecidable, namely Π^0_1-complete, as shown in (Grabmayer *et al.*, 2012). This semantics (equality of normal forms for productive specifications) thus forms an exception in comparison to all other semantics that we are about to discuss, in the sense that inequalities can be recursively enumerated.

The (productive) specifications used for Π^0_1-completeness of their equivalence are remarkably simple; they are formed by equations $X = t$ where terms t are built from the signature $\Sigma = \{\mathsf{zip}_k, \mathsf{proj}_{i,k} \mid k \in \mathbb{N}, 0 \leq i < k\} \cup B$ with B a finite non-empty set of data constants, and including the following defining equations for zip_k and $\mathsf{proj}_{i,k}$:

$$\begin{aligned}
\mathsf{zip}_k(a : x_0, x_1, \ldots, x_{k-1}) &= a : \mathsf{zip}_k(x_1, \ldots, x_{k-1}, x_0) \\
\mathsf{proj}_{0,k}(a : x) &= a : \mathsf{proj}_{k-1,k}(x) \\
\mathsf{proj}_{i,k}(a : x) &= \mathsf{proj}_{i-1,k}(x) \qquad (0 < i < k)\,.
\end{aligned}$$

When we restrict the signature even further by dropping the projection functions $\mathsf{proj}_{i,k}$, and allow only symbols zip_k for a fixed $k \geq 2$, then equality becomes decidable, again see (Grabmayer *et al.*, 2012). It is an open problem whether equivalence of 'zip-mix' specifications, where zips of different arities are allowed, is decidable.

1.1 On Semantics of Stream Equality

The semantic candidates for stream equality that we consider in this paper are as follows:

I. Equality in hidden models, i.e., behavioral equivalence.
II. Equality in extensional models.
   (a) Equality in all extensional models.
   (b) Equality in all full extensional models.
III. Equality of sets of solutions.
   (a) Equality of sets of solutions over all extensional models.
   (b) Equality of sets of solutions over all full extensional models.

The 'right' choice of equality depends on the intended application. The classic semantics mentioned in items II and III above, are defined by model-theoretic means. From an algebraic perspective, these are the natural semantics to consider for equational reasoning. Semantics I, which is based on hidden models with behavioral equality, has a wide range of applications for modeling non-determinism, and hidden values.

We briefly describe the three semantics. The basis of each of them is an interpretation of specifications in Σ-algebras, see e.g. (Ehrig & Mahr, 1985). Here Σ is the signature of the specification that we intend to model, that is, a set of ranked symbols occurring in the specification. A Σ-algebra $\mathscr{A}$ consists of a carrier set A, also called the domain of $\mathscr{A}$, and an interpretation function $[\![\cdot]\!]$ of the symbols in Σ (respecting their arities). We consider specifications of streams over the set $\{0,1\}$.[2] We let algebras always contain the following two interpretations:

$$[\![\mathsf{head}]\!] : A_S \to \{0,1\} \qquad \text{and} \qquad [\![\mathsf{tail}]\!] : A_S \to A_S\,,$$

[2] The concrete choice of $\{0,1\}$ is irrelevant for the results in this paper; the results remain unchanged when replacing $\{0,1\}$ by any finite set B of symbols of arity 0 that contains at least 2 elements.

where A_S is the carrier of $\mathscr{A}$ corresponding to its streams, $[\![\mathsf{head}]\!](w)$ represents the first element of the stream w, and $[\![\mathsf{tail}]\!](w)$ the stream without the first element.

In general, there are no restrictions on how Σ-algebras represent streams. For example, in some Σ-algebras the stream elements of A_S can be infinite words of 0 and 1 bits, in others they can be functions from $\mathbb{N}$ to $\{0,1\}$, in others infinite trees (with a specific traversal meaning), etc. Nevertheless, in all cases, any stream Σ-term s with variables in a set $\mathscr{X}$ can be uniquely interpreted into a stream element $[\![s]\!]_\alpha \in A_S$ for each assignment $\alpha : \mathscr{X} \to A$ of its variables to appropriate elements in the carrier of the Σ-algebra.

The interpretations of the head and tail operations determine a notion of *behavioral equivalence* on the streams of the Σ-algebra, which can be used to define the semantics of stream equations. Two streams $u, v \in A_S$ are behaviorally equivalent, written $u \equiv v$, iff they are indistinguishable with respect to all $\langle \mathsf{head}, \mathsf{tail}\rangle$-experiments, that is,

$$u \equiv v \iff \forall n \in \mathbb{N}.\ [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(u)) = [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(v)),$$

where $[\![\mathsf{tail}]\!]^n(u) = \overbrace{[\![\mathsf{tail}]\!](\cdots[\![\mathsf{tail}]\!]}^{n \text{ times}}(u)\cdots)$. Given two stream Σ-terms s and t, possibly with variables in set $\mathscr{X}$, we say $\mathscr{A}$ *behaviorally satisfies* equation $s = t$, written $\mathscr{A} \models\!\!\equiv s = t$, iff $[\![s]\!]_\alpha \equiv [\![t]\!]_\alpha$ for all $\alpha : \mathscr{X} \to A$. As usual, if E is a set of equations then $\mathscr{A} \models\!\!\equiv E$ iff $\mathscr{A} \models\!\!\equiv e$ for each $e \in E$. If $\mathscr{A} \models\!\!\equiv E$ then we also say that $\mathscr{A}$ is a *model* of E.

We define $E \models\!\!\equiv e$ iff $\mathscr{A} \models\!\!\equiv E$ implies $\mathscr{A} \models\!\!\equiv e$ for all Σ-algebras $\mathscr{A}$. Depending upon which Σ-algebras are allowed to serve as stream models, the problem whether $E \models\!\!\equiv e$ can have different degrees of complexity. Intuitively, the more Σ-algebras are allowed as stream models, the fewer pairs in the relationship $E \models\!\!\equiv e$, so the lower the complexity of deciding it. Our results in this paper confirm this intuition.

Equality in Hidden Models (I). Hidden models (Goguen, 1991; Malcolm, 1997; Goguen & Malcolm, 2000; Roşu, 2000) are the least restrictive stream models that we are aware of. In fact, they add no additional restrictions to the above. In particular, they do not even require the existence of a constructor operation ':' for streams (prepending a 0 or 1 to an existing stream). For example, the streams zeros, ones, blink, and the operation zip discussed at the beginning of this section can be defined as follows:

$$\begin{array}{rrr}
\mathsf{head}(\mathsf{zeros}) = 0 & \mathsf{tail}(\mathsf{zeros}) = \mathsf{zeros} & \\
\mathsf{head}(\mathsf{ones}) = 1 & \mathsf{tail}(\mathsf{ones}) = \mathsf{ones} & \\
\mathsf{head}(\mathsf{blink}) = 0 & \mathsf{head}(\mathsf{tail}(\mathsf{blink})) = 1 & \mathsf{tail}(\mathsf{tail}(\mathsf{blink})) = \mathsf{blink} \\
\mathsf{head}(\mathsf{zip}(x,y)) = \mathsf{head}(x) & \mathsf{tail}(\mathsf{zip}(x,y)) = \mathsf{zip}(y, \mathsf{tail}(x)). &
\end{array}$$

If E is the equational specification above then we can show that $E \models\!\!\equiv e$, where e is any of

$$\mathsf{zip}(\mathsf{zeros}, \mathsf{ones}) = \mathsf{zeros}$$
$$\mathsf{zip}(\mathsf{ones}, \mathsf{ones}) = \mathsf{ones}$$
$$\mathsf{zip}(\mathsf{zeros}, \mathsf{ones}) = \mathsf{blink}.$$

While hidden algebras turned out to be quite appropriate for specifying and reasoning about non-deterministic and/or concurrent systems (Goguen & Malcolm, 2000; Roşu, 2000), one can admittedly claim that, at least in the context of streams, for certain purposes there could be too many models allowed for equational specifications. In general there is no requirement in the hidden algebra semantic approach that all streams are constructed with ':'. That is, there may be elements $\sigma \in A_S$ in the stream domain for which there exist no $a \in \{0,1\}$ and $\tau \in A_B$ such that $\sigma = [\![:]\!](a,\tau)$. For example, if we replace the equational specification above with the specification (1) at the beginning of this section where ':' is an operation whose semantics is given by the equations

$$\mathsf{head}(a:x) = a \qquad\qquad \mathsf{tail}(a:x) = x$$

then there is no way to prove the equations e above, because, essentially, there is no way to even prove $\mathsf{head}(\mathsf{zip}(x,y)) = \mathsf{head}(x)$. We can only prove it when x is a stream of the form $a:x'$. In fact, the real problem is the fact that the behavioral equivalence relation is not required to be a congruence (i.e., to be preserved by all operations in Σ) in a hidden algebra; if it were, we would be able to transform the hidden algebra into an equivalent one which has the property that each stream element x has the form $a:x'$ for some bit element a and some other stream x' (see Proposition 3.15).

This brings us to a first meaningful restriction on hidden algebras, which yields to what (Bidoit *et al.*, 2003) call *behavioral algebras*: a behavioral Σ-algebra is a hidden Σ-algebra in which the behavioral equivalence is a congruence. It turns out (see Proposition 3.15) that $E \models\!\!\equiv e$ with behavioral models if and only if $E \models\!\!\equiv e$ with extensional models, so we do not need to discuss behavioral models in much depth.

Equality in Extensional Models (II). Extensional models further restrict the hidden models to ones whose behavioral equivalence is the identity. In other words, each stream in the model is uniquely characterized by its behavior: $u = v \iff u \equiv v$. This suggests a further simplification without affecting the satisfaction problem $E \models\!\!\equiv u = v$ in any way: we introduce so-called extensional algebras where the carrier A_S contains only infinite sequences over $\{0,1\}$, i.e., $A_S \subseteq \{0,1\}^\omega$. A particularly interesting special case is when $A_S = \{0,1\}^\omega$, that is, when their carrier contains all the streams; we call such extensional models *full*.

Equality of Sets of Solutions (III). Semantics I and II are useful when the objects under consideration are specified in the same specification. These semantics interpret the objects simultaneously in each model satisfying the specification. But they fail to be effective when streams or stream operations are underspecified. Consider, for example, an underspecified constant M together with a renamed copy N of it,

$$\mathsf{M} = 0 : \mathsf{tail}(\mathsf{M}) \qquad\qquad \mathsf{N} = 0 : \mathsf{tail}(\mathsf{N})\,. \tag{2}$$

Here M and N are not equal in every model, because we can take M to be $0\,0^\omega$ and N to be $0\,1^\omega$. Nevertheless, M and N are equal in the sense that they exhibit the same behaviors. That is, in every extensional model they have the same set of solutions: every stream starting with a zero is a solution for M as well as for N. Thus, M and N are equal with respect to semantics III which defines equality via the set of solutions. Like in the case of extensional models, we consider two kinds of sets of solutions: in all extensional models, versus in all full extensional models.

In contrast to I and II, semantics III is more suitable for comparing objects specified by different specifications, as we explain below. The objects are compared via the set of their solutions (in their respective specifications). This semantics is well-known from equations over real (or complex) numbers, where two equations, like

$$(x-1)^2 - 1 = 0 \qquad\qquad \text{and} \qquad\qquad x^2 - 2x = 0\,,$$

are equivalent if they have the same solutions for x, here $\{0,2\}$.

Semantics III enables us to compare streams M and N given by separate specifications E_{M} and E_{N}, respectively. An alternative approach would be the application of semantics II to the union $E_{\mathsf{M}} \cup E_{\mathsf{N}}$. But even if the specifications have disjoint signatures (using renaming), taking the union of the specifications may result in an additional restriction on the admissible domains, and thus objects may be wrongly identified, see further Remark 6.18. For a trivial example, assume that one of the specifications has no model, then also the union has no models, despite the fact that the other specification may have a unique solution.

Two stream constants M and N are equal with respect to semantics III if the set of solutions of M in E_{M} coincides with the set of solutions of N in E_{N}:

$$\{\, [\![\mathsf{M}]\!]^{\mathscr{A}} \mid \mathscr{A} \models\!\equiv E_{\mathsf{M}} \,\} \;=\; \{\, [\![\mathsf{N}]\!]^{\mathscr{A}} \mid \mathscr{A} \models\!\equiv E_{\mathsf{N}} \,\}$$

Here the set of solutions of a constant X in a specification E_{X} is the set of interpretations of X in all (full) extensional models of E_{X}.

1.2 Contributions

We characterize the complexity of the problem of deciding equivalence of equational specifications of streams for each of the semantics I, IIa, IIb, IIIa and IIIb listed above. As all of these problems are undecidable, we classify them by means of the arithmetical and analytical hierarchies; see Figure 2. In the arithmetical hierarchy the complexity of a problem P is classified by the minimal number of quantifier alternations in first-order formulas that characterize P. The analytical hierarchy extends this classification to second-order arithmetic, then counting the alternations of set quantifiers.

A We show that the problem of deciding behavioral equivalence in all hidden models (semantics I) is Π^0_2-complete, and thus resides at a low level of the arithmetical hierarchy. This is Theorem 5.1.

B We then strengthen the notion of models by requiring behavioral equivalence $\equiv$ to be a congruence, leading to behavioral models. Surprisingly, it turns out that this simple requirement catapults the complexity of deciding equality out of the arithmetical hierarchy to the level Π^1_1 of the analytical hierarchy. See Theorem 6.6. This also holds for extensional models, which further restrict the behavioral equivalence to be the identity and their carriers to be subsets of $\{0,1\}^{\omega}$, because we show that satisfaction with extensional models is equivalent to satisfaction with behavioral models.

C We further strengthen the extensional models to be full, that is, require their domain to contain all bitstreams $\{0,1\}^{\omega}$. This again yields a huge jump in the complexity of the problem: the complexity of deciding equality in all full extensional models subsumes the entire arithmetical and analytical hierarchy (see Definition 6.11). See Theorem 6.14.

$$\begin{tikzcd}
 & & \textcircled{C} & & \\
 & & \vdots & & \\
\forall X.\,\exists Y.\,\Phi(X,Y) & \Pi^1_2 \arrow[ur] & & \Sigma^1_2 \arrow[ul] & \exists X.\,\forall Y.\,\Phi(X,Y) \\
 & & \Delta^1_2 \arrow[ul] \arrow[ur] & & \\
\textit{well-foundedness}\ \textcircled{B}\ \ \forall X.\,\Phi(X) & \Pi^1_1 \arrow[ur] & & \Sigma^1_1 \arrow[ul] & \exists X.\,\Phi(X) \\
 & & \Pi^1_0 = \Delta^1_0 = \Sigma^1_0 \arrow[ul] \arrow[ur] & & \textit{arithmetic predicates } \Phi \\
 & & \vdots & & \\
\textit{totality}\ \textcircled{A}\ \ \forall x.\,\exists y.\,D(x,y) & \Pi^0_2 \arrow[ur] & & \Sigma^0_2 \arrow[ul] & \exists x.\,\forall y.\,D(x,y) \\
 & & \Delta^0_2 \arrow[ul] \arrow[ur] & & \\
\forall x.\,D(x) & \Pi^0_1 \arrow[ur] & & \Sigma^0_1 \arrow[ul] & \textit{recursively enumerable}\ \ \exists x.\,D(x) \\
 & & \Pi^0_0 = \Delta^0_0 = \Sigma^0_0 \arrow[ul] \arrow[ur] & & \textit{decidable predicates } D
\end{tikzcd}$$

Fig. 2. Arithmetical (bottom) and analytical hierarchy (top).

Since all these results hold for specifications of bitstreams, they serve as a lower bound on the hardness of the equivalence problem for coinductive objects with richer structure.

Further results. Besides the complexity of deciding equivalence in the mentioned semantics, we also consider the following related problems:

(i) *Complexity of deciding equality of sets of solutions.*

We study the complexity of deciding whether two terms (possibly given by distinct specifications) have the same set of solutions. Again, we have the choice whether to consider all extensional models, or only full extensional models.

When we consider the set of solutions over all extensional models, the problem turns out to be Π^1_2-complete. See Theorem 6.20. This is a higher complexity class than Π^1_1 where equality in all extensional models resides. Intuitively, the reason is that for every solution of one specification, we need to check the existence of an equivalent solution of the other specification (and this translates to second-order quantifiers $\forall\exists$ quantifying over models).

When we consider the set of solutions over all full extensional models, the complexity of the problem is again raised to subsume the analytical hierarchy. See Theorem 6.19.

(ii) *Deciding the existence of (i) at least one, (ii) at most one, (iii) precisely one solution.*

When giving semantics to specifications of (finite or infinite) objects, it is a natural question to ask whether the specification has at least one solution, that is, whether there exists a model at all. A specification may admit several models, and it is interesting to know whether a term under consideration has the same interpretation in all these models, or can have multiple solutions.

The obvious questions are as follows; does a given term have

(i) at least one solution,
(ii) at most one solution,
(iii) precisely one solution?

Again, when considering these questions over all full models, the complexity of each problem subsumes the entire analytical hierarchy. See Theorems 6.15, 6.16, and 6.17.

The situation becomes more interesting when we consider all extensional models, or equivalently, behavioral models. The problems (i) and (ii) turn out to be Σ^1_1-complete and Π^1_1-complete, respectively. See Theorems 6.9 and 6.8. Problem (iii) is both Π^1_1-hard and Σ^1_1-hard, but is strictly contained in Δ^1_2. See Theorem 6.10.

(iii) *Hidden models for streams over natural numbers.*

Finally, we consider the complexity of deciding the equality of streams of natural numbers, instead of bits, in hidden models. This raises the complexity from the class Π^0_2 of the arithmetical hierarchy (for specifications of bitstreams) to the class Π^1_1 of the analytical hierarchy. See Theorem 7.2.

Note: This article merges, extends and modifies results published in two ICFP conferences which were six years apart, namely (Roşu, 2006) in ICFP 2006 and (Endrullis *et al.*, 2012b) in ICFP 2012. The former was the first to claim the results for hidden models, while the latter was the first to claim the results for (full) extensional models. This article presents the two groups of strongly related results in a uniform notational setting, discusses additional relationships and connections between them (such as the relationship between behavioral and extensional models), and finally includes all the detailed proofs of the claimed results.

1.3 Related Results in Computer Science

Let us mention a few related complexity results in the area of computer science. Dynamic logic has been shown to be Π^1_1-complete in (Meyer *et al.*, 1981), see also (Harel *et al.*, 2000). The paper (Castro & Cucker, 1989) presents several complexity results on ω-computations. For the proof of Theorem 6.20, we have made use of their Π^1_2-completeness result for the problem of deciding whether a non-deterministic Turing machine accepts all ω-words. For further results on ω-languages, see (Finkel & Lecomte, 2009). The paper (Harel, 1985) gives a nice exposition of various complexity classes up to Π^1_1 and Σ^1_1, and illustrates these classes by means of various tiling problems. In (Endrullis *et al.*, 2011b) the complexity of various properties of rewriting systems is determined, and Π^1_1-completeness of dependency pair problems is shown. The paper (Endrullis *et al.*, 2009) is concerned with the complexity of productivity; it is shown that *strong* and *weak* productivity are Π^1_1- and Σ^1_1-complete, respectively.

2 Turing Machines, Levels of Undecidability, and Term Rewriting

In order to set notation and to make the exposition self-contained, we recall the basic notions of Turing machines, of levels of undecidability including the arithmetic and the analytic hierarchies, and of term rewriting.

2.1 Turing Machines

We consider Turing machines with left- and right-infinite tapes.

Definition 2.1. A *(deterministic) Turing machine* T is a tuple $\langle Q, \Gamma, q_0, \delta \rangle$ consisting of

– a finite set of states Q,

– an initial state $q_0 \in Q$,

– a finite alphabet Γ containing a designated *blank* symbol $\Box$, and

– a partial *transition function* $\delta : Q \times \Gamma \rightharpoonup Q \times \Gamma \times \{L, R\}$.

A *configuration* of a Turing machine T is a pair $\langle q, \tau \rangle$ consisting of a state $q \in Q$ and a *tape* $\tau : \mathbb{Z} \to \Gamma$. We define the *transition relation* $\to_T$ on the set of configurations by

$$\begin{aligned}
(q, \tau) &\to (q', \tau') \quad \text{if } \delta(q, \tau(0)) = \langle q', \tau'(1), L \rangle \text{ and } \tau'(x) = \tau(x-1) \text{ for } x \in \mathbb{Z} \setminus \{1\}, \\
(q, \tau) &\to (q', \tau') \quad \text{if } \delta(q, \tau(0)) = \langle q', \tau'(-1), R \rangle \text{ and } \tau'(x) = \tau(x+1) \text{ for } x \in \mathbb{Z} \setminus \{-1\}.
\end{aligned}$$

The Turing machine T is said to *halt on tape* τ_0 *with output* $b \in \{0, 1\}$ if the maximal sequence

$$\langle q_0, \tau_0 \rangle \to_T \langle q_1, \tau_1 \rangle \to_T \cdots \to_T \langle q_n, \tau_n \rangle$$

of $\to_T$-steps starting from the initial configuration $\langle q_0, \tau_0 \rangle$ is finite (that is, the configuration $\langle q_n, \tau_n \rangle$ does not admit any further steps), and $\tau_n(0) = b$. We say that T *halts on input* $w \in \{0, 1\}^*$ *with output* b if it halts with output b on the tape τ defined by $\tau(z) = w(z)$ for all $0 \le z < |w|$, and $\tau(z) = 0$ for every other $z \in \mathbb{Z}$.

Convention 2.2. For convenience we restrict Γ to the alphabet $\Gamma = \{0, 1\}$ where 0 is the blank symbol $\Box$, and we denote Turing machines by triples $\langle Q, q_0, \delta \rangle$.

Note that we use a non-standard variant of Turing machines: we restrict the alphabet to $\{0, 1\}$ *without* a third symbol for $\Box$, and we have no designated halting states (the machine halts if there is no transition possible). It is straightforward to show that by this restriction we do not lose any expressive power.

Proposition 2.3. *Turing machines as restricted in Convention 2.2 can define all computable functions.*

As input for the Turing machines we use a unary number representation, that is, we use $11 \cdots 1$ (n times 1) to encode the natural number n. Of course, other encodings are possible, as long as the encoding is computable, and the Turing machine is able to detect the end of the input (here one has to take care of the double role of 0). For encoding multiple inputs $n_1, \ldots, n_k$ we will interleave the unary representations of $k, n_1, \ldots, n_k$; see further

Definitions 4.2 and 4.7. For example, for $k = 2$, the inputs $n_1 = 3$ and $n_2 = 2$ will be interleaved as

$$\mathsf{zip}_3(11000\cdots, \hat{1}\hat{1}\hat{1}\hat{0}\hat{0}\cdots, \overline{11000}\cdots) = 1\hat{1}1\bar{1}0\hat{1}0\bar{1}0\hat{1}0\bar{0}0\hat{0}0\bar{0}0\hat{0}\cdots.$$

The markings above the symbols indicate the origin of the interleaved symbols. Note that the number k can be extracted from the result by taking the elements at indices 0, 2, 4, 6, ..., the number n_1 from the elements at indices 1, 5, 9, 13, ..., and the number n_2 from the elements at indices 3, 7, 11, 15, ...

2.2 Levels of Undecidability

We briefly introduce the complexity related notions relevant for this paper: promise problems, reducibility, hardness and completeness, as well as the arithmetical and the analytical hierarchy. All results in this section are classical. For more details, we refer to the standard textbooks (Rogers, Jr., 1967), (Shoenfield, 1971), (Hinman, 1978), (Odifreddi, 1992), and (Odifreddi, 1999).

Definition 2.4. Let $A \subseteq P \subseteq \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. The *promise problem for A with promise P*, written $A|_P$, is the question of deciding on the input of $n \in P$ whether $n \in A$. For the case $P = \mathbb{N}$, we speak of the *membership problem for A*.

We identify the membership problem for A with the set A itself, and the promise problem for A with promise P with the pair $\langle A, P\rangle \in \wp(\mathbb{N}) \times \wp(\mathbb{N})$, also written $A|_P$.

Definition 2.5. Let $A, B, P, Q \subseteq \mathbb{N}$. Then $A|_P$ *can be (many-one) reduced to* $B|_Q$, denoted by $A \leq B$, if there exists a partial recursive function $f : \mathbb{N} \rightharpoonup \mathbb{N}$ such that $P \subseteq \mathit{domain}(f)$, $f(P) \subseteq Q$, and $\forall n \in P. n \in A \Leftrightarrow f(n) \in B$.

Definition 2.6. Let $B, Q \subseteq \mathbb{N}$ and $\mathscr{P} \subseteq \wp(\mathbb{N}) \times \wp(\mathbb{N})$. Then $B|_Q$ is called *$\mathscr{P}$-hard* if every $A|_P \in \mathscr{P}$ can be reduced to $B|_Q$. Moreover, $B|_Q$ is *$\mathscr{P}$-complete* if additionally $B|_Q$ can be reduced to some $A|_P \in \mathscr{P}$.

We stress that Definition 2.6 does not require that a $\mathscr{P}$-complete promise problem $B|_Q$ is a member of $\mathscr{P}$ itself. This allows for classifying promise problems using the usual arithmetic and analytical hierarchy (for membership problems).

Lemma 2.7. *If $A|_P$ can be reduced to $B|_Q$ and $A|_P$ is* $\mathscr{P}$-hard*, then B is* $\mathscr{P}$-hard.

We use $\langle\langle\cdot\rangle\rangle$ to denote the well-known Gödel encoding of finite *lists* of numbers as elements of $\mathbb{N}$,

$$\langle\langle n_1, n_2, \ldots, n_k\rangle\rangle := p_1^{n_1+1} \cdot p_2^{n_2+1} \cdots p_k^{n_k+1},$$

where $p_1 < p_2 < \cdots < p_k$ are the first k prime numbers.

Next, we recall the definitions of the arithmetical and analytical hierarchy. See also Figure 2.

Definition 2.8. Let $\Sigma^0_0 := \Pi^0_0 := \Delta^0_0$ be the collection of recursive sets of natural numbers (the decidable problems). For $n \geq 1$, we define:

$$\begin{aligned}
\Sigma^0_n &:= \left\{ \{ m \in \mathbb{N} \mid \exists x \in \mathbb{N}.\, \langle\!\langle x, m \rangle\!\rangle \in B \} \mid B \in \Pi^0_{n-1} \right\},\\
\Pi^0_n &:= \left\{ \{ m \in \mathbb{N} \mid \forall x \in \mathbb{N}.\, \langle\!\langle x, m \rangle\!\rangle \in B \} \mid B \in \Sigma^0_{n-1} \right\},\\
\Delta^0_n &:= \Sigma^0_n \cap \Pi^0_n\,.
\end{aligned}$$

The *arithmetical hierarchy* consists of the classes Π^0_n, Σ^0_n and Δ^0_n for $n \in \mathbb{N}$.

For example, the membership $a \in A$ for every set $A \in \Pi^0_2$ can be defined by a formula $\forall x_1.\, \exists x_2.\, \forall x_3.\, P(a, x_1, x_2, x_3)$ where P is a decidable predicate. The classes Σ^0_1 and Π^0_1 are the collections of semi-decidable and co-semi-decidable problems, respectively. It is well-known that $\Delta^0_n \subsetneq \Sigma^0_n$, $\Delta^0_n \subsetneq \Pi^0_n$, and $\Sigma^0_n \cup \Pi^0_n \subsetneq \Delta^0_{n+1}$ for all $n \geq 1$, see (Hinman, 1978). Hence $\bigcup_{n\in\mathbb{N}} \Sigma^0_n = \bigcup_{n\in\mathbb{N}} \Pi^0_n = \bigcup_{n\in\mathbb{N}} \Delta^0_n$.

The analytical hierarchy extends this classification of sets to formulas of the language of second-order arithmetic, that is, with set (or equivalently function) quantifiers. The following definition makes use of a result from recursion theory, see (Rogers, Jr., 1967), stating that if there is at least one quantifier over sets, then two *quantifiers over numbers* suffice (for functions quantifiers, one quantifier over numbers suffices).

Definition 2.9. Let $\Sigma^1_0 := \Pi^1_0 := \Delta^1_0 = \bigcup_{n\in\mathbb{N}} \Pi^0_n$ be the set of all arithmetic predicates. A set $A \subseteq \mathbb{N}$ is in Π^1_n for $n > 0$ if there is a decidable predicate $P \subseteq \wp(\mathbb{N})^n \times \mathbb{N}^3$ such that for all $a \in \mathbb{N}$:

$$\begin{aligned}
a \in A &\iff \forall \xi_1.\, \exists \xi_2. \ldots \forall \xi_{n-1}.\, \exists \xi_n.\, \forall x_1.\, \exists x_2.\, P(\xi_1, \ldots, \xi_n, a, x_1, x_2)\\
a \in A &\iff \forall \xi_1.\, \exists \xi_2. \ldots \exists \xi_{n-1}.\, \forall \xi_n.\, \exists x_1.\, \forall x_2.\, P(\xi_1, \ldots, \xi_n, a, x_1, x_2)
\end{aligned}$$

for n even, and n odd, respectively. Here, $\xi_1, \ldots, \xi_n \subseteq \mathbb{N}$ range over sets of natural numbers, (the corresponding quantifiers are set quantifiers), and $x_1, x_2 \in \mathbb{N}$ range over natural numbers (the corresponding quantifiers are quantifiers over numbers). Dually, A is in Σ^1_n, if the condition holds with all $\forall$ and $\exists$ quantifiers swapped. Finally, $\Delta^1_n := \Pi^1_n \cap \Sigma^1_n$.

We introduce some representative problems for different levels of the arithmetical and analytical hierarchy. We begin with the well-known empty tape halting problem for Turing machines.

Definition 2.10. The *empty tape halting problem* is the following problem:

INPUT: A Turing machine T.

QUESTION: Does T halt on the empty tape?

Proposition 2.11. *The empty tape halting problem is Σ^0_1-complete.*

The class Σ^0_1 corresponds to one existential number quantifier over a decidable predicate. Here the decidable predicate $P(n)$ is whether the Turing machine halts in n steps, and we quantify over the number of steps until termination. The class Σ^0_1 consists of the recursively enumerable, or semi-decidable problems. Indeed, we can enumerate the Turing machines that halt on the empty tape (but not the Turing machines which do not halt).

The problem becomes more difficult if we ask for termination on an infinite set of inputs. For example termination on all natural numbers is known as the totality problem. This problem resides at the level Π^0_2 of the arithmetical hierarchy.

Definition 2.12. The *totality problem* is the following problem:

INPUT: A Turing machine T.

QUESTION: Does T halt on every input $n \in \mathbb{N}$?

Proposition 2.13. *The totality problem is* Π^0_2*-complete.*

To see that the totality problem is in Π^0_2, observe that it has a positive answer if and only if *for any* input $n \in \mathbb{N}$, *there exists* a computation which halts T; checking whether a given computation on a given tape is correct and ends in a halting state is decidable.

A significantly more difficult problem is deciding the well-foundedness of (computable) binary relations $P \subseteq \mathbb{N} \times \mathbb{N}$ (given in the form of Turing machines).

The well-foundedness problem resides in the class Π^1_1 of the analytical hierarchy. The check whether there are no infinite chains $n_1 \mathrel{P} n_2 \mathrel{P} n_3 \mathrel{P} \dots$ requires a universal quantification over all streams $n_1 : n_2 : n_3 : \dots$ in front of an arithmetical formula that verifies the existence of an index $i \in \mathbb{N}$ such that $n_i \mathrel{P} n_{i+1}$ does not hold.

Definition 2.14. The *well-foundedness problem* for decidable binary relations is the following problem:

INPUT: Decidable binary predicate $P \subseteq \mathbb{N} \times \mathbb{N}$ in form of a Turing machine.

QUESTION: Is P well-founded?

Proposition 2.15. *The well-foundedness problem is* Π^1_1*-complete.*

The following is a result of (Castro & Cucker, 1989) stating that the problem of deciding whether the ω-language of a non-deterministic Turing machine contains all words $\{0,1\}^\omega$ is Π^1_2-complete.

Definition 2.16. Let T be a *non-deterministic* Turing machine, and $w \in \{0,1\}^\omega$ a stream. We *start* T *on the stream* w by letting T run on the configuration τ defined by $\tau(n) = w(n)$ for all $n \in \mathbb{N}$ and $\tau(z) = 0$ for all $z < 0$. A run of T is *complete* if every tape position $p \geq 0$ is visited (that is, positions right of the starting position), and it is *oscillating* if some tape position is visited infinitely often. A run is *accepting* if it is complete and not oscillating, that is, it visits every position $p \geq 0$ at least once, but only finitely often. The ω*-language* $\mathscr{L}^\omega(T)$ *of* T is the set of all streams $w \in \{0,1\}^\omega$ such that T has an accepting run w.

Proposition 2.17. *The following problem is* Π^1_2*-complete:*

INPUT: *A Turing machine T (encoded as a natural number).*

QUESTION: *Does T accept all streams $w \in \{0,1\}^\omega$, that is, $\mathscr{L}^\omega(T) = \{0,1\}^\omega$?*

2.3 Terms and Term Rewriting

Although the specifications we consider in this paper always consist of equations over finite terms we also make use of infinite terms, which we define below. Moreover, as specifications of bitstreams are inherently sorted, with a sort for bits, and a sort for bitstreams, we introduce sorted terms.

Definition 2.18. Let $\mathscr{S}$ be a set of sorts. An *$\mathscr{S}$-sorted set C* is a family of sets $\{C_s\}_{s\in\mathscr{S}}$. Let C and D be $\mathscr{S}$-sorted sets. Then an $\mathscr{S}$-sorted *function* (or *map*) from C to D is a function $f : C \to D$ such that $f(C_s) \subseteq D_s$ for all $s \in \mathscr{S}$, that is, a function that respects the sorts.

An $\mathscr{S}$-sorted *signature* Σ is a set of symbols $f \in \Sigma$, each having a *type* $(s_1,\ldots,s_n,s) \in \mathscr{S}^{n+1}$, which we denote by $f :: s_1 \times \cdots \times s_n \to s$, where n is the arity of f. If the arity of a symbol f is 0 we just write $f :: s$ and call it a *constant*. Let $\mathscr{X}$ be an $\mathscr{S}$-sorted set of *variables*. The $\mathscr{S}$-sorted set of (finite and) infinite *terms $Ter^\infty(\Sigma,\mathscr{X})$* is coinductively defined as follows: for every $s \in \mathscr{S}$, and every term $t \in Ter^\infty(\Sigma,\mathscr{X})_s$ we have either

- t is a variable $t \in \mathscr{X}_s$, or
- t is of the form $t = f(t_1,\ldots,t_n) \in Ter^\infty(\Sigma,\mathscr{X})_s$ with $f \in \Sigma$ of type $s_1 \times \cdots \times s_n \to s$, and terms $t_1 \in Ter^\infty(\Sigma,\mathscr{X})_{s_1},\ldots,t_n \in Ter^\infty(\Sigma,\mathscr{X})_{s_n}$.

Let $t \in Ter^\infty(\Sigma,\mathscr{X})$. The set of *positions $\mathscr{P}os(t)$* $\subseteq \mathbb{N}^*$ of t is defined by

$$\mathscr{P}os(x) = \{\varepsilon\}\,, \qquad \mathscr{P}os(f(t_1,\ldots,t_n)) = \{\varepsilon\} \cup \{ip \mid 1 \le i \le n,\ p \in \mathscr{P}os(t_i)\}\,.$$

A term $t \in Ter^\infty(\Sigma,\mathscr{X})$ is called *finite* if the set $\mathscr{P}os(t)$ is finite, and we use $Ter(\Sigma,\mathscr{X})$ to denote the set of finite $\mathscr{S}$-sorted terms.

For $p \in \mathscr{P}os(t)$, the *subterm $t|_p$ of t at position p* is defined by

$$t|_\varepsilon = t\,, \qquad\qquad f(t_1,\ldots,t_n)|_{ip} = t_i|_p\,.$$

Let $s \in \mathscr{S}$. A *substitution σ* is a map $\sigma : \mathscr{X}_s \to Ter^\infty(\Sigma,\mathscr{X})_s$. The domain of a substitution σ is extended to terms $Ter^\infty(\Sigma,\mathscr{X})$ by

$$\sigma(f(t_1,\ldots,t_n)) = f(\sigma(t_1),\ldots,\sigma(t_n))\,.$$

Let $\Box$ be a fresh variable, $\Box \notin \mathscr{X}$. A *context C* is a term $Ter^\infty(\Sigma,\mathscr{X} \cup \{\Box\})$ containing precisely one occurrence of the variable $\Box$. By $C[t]$ we denote the term $\sigma(C)$ where $\sigma(\Box) = t$ and $\sigma(x) = x$ for all $x \in \mathscr{X}$.

A *rewrite rule* $\ell \to r$ is a pair $(\ell,r) \in Ter(\Sigma,\mathscr{X}) \times Ter(\Sigma,\mathscr{X})$ of finite terms of the same sort such that $\ell \notin \mathscr{X}$ and $Var(r) \subseteq Var(\ell)$. A *term rewrite system (TRS) R* is a finite set of rewrite rules. A TRS induces a rewrite relation on the set of terms as follows. For $p \in \mathbb{N}^*$ we define $\to_{R,p} \subseteq Ter^\infty(\Sigma,\mathscr{X}) \times Ter^\infty(\Sigma,\mathscr{X})$, a *rewrite step at position p*, by

$$C[\ell\sigma] \to_{R,p} C[r\sigma] \quad \text{if } C \text{ is a context with } C|_p = \Box,\ \ell \to r \in R,\ \sigma : \mathscr{X} \to Ter^\infty(\Sigma,\mathscr{X}).$$

We write $s \to_R t$ if $s \to_{R,p} t$ for some $p \in \mathbb{N}^*$. A *normal form* is a term without a redex occurrence, that is, a term that is not of the form $C[\ell\sigma]$ for some context C, rule $\ell \to r \in R$ and substitution σ.

We write $\to^*$ for the reflexive-transitive closure of a relation $\to$.

Remark 2.19. Intuitively, the above coinductive definition of the set $Ter^\infty(\Sigma,\mathscr{X})$ means that the grammar rules may be applied an infinite number of times. One can thus think of infinite terms as infinite labeled trees. Alternatively (now ignoring sorts) one may define a term $t \in Ter^\infty(\Sigma,\mathscr{X})$ as a partial map $t : \mathbb{N}^* \rightharpoonup \Sigma \cup \mathscr{X}$ such that $t(\varepsilon) \in \Sigma \cup \mathscr{X}$, and for all $p \in \mathbb{N}^*$ and $i \in \mathbb{N}$ we have $t(pi) \in \Sigma \cup \mathscr{X}$ if and only if $t(p) \in \Sigma$ of arity n and $1 \le i \le n$. The set of positions $\mathscr{P}os(t)$ of a term $t \in Ter^\infty(\Sigma)$ then is the domain of t, i.e., the set of values $p \in \mathbb{N}^*$ such that $t(p)$ is defined: $\mathscr{P}os(t) = \{p \in \mathbb{N}^* \mid t(p) \in \Sigma \cup \mathscr{X}\}$. Note that the set $\mathscr{P}os(t)$ is prefix-closed.

An $\mathscr{S}$-sorted *equation* $s = t$ consists of terms $s, t \in Ter(\Sigma, \mathscr{X})_s \times Ter(\Sigma, \mathscr{X})_s$ for some $s \in \mathscr{S}$. An *equational specification* is a finite set of equations. Equational specifications are very similar to term rewriting systems. The only difference is that in equational reasoning the equations (rules in term rewriting) may be applied in both directions (from left to right, and from right to left).

3 Stream Specifications and Semantics

Streams are one-sided infinite sequences of symbols. There are various ways of introducing streams: as functions $\mathbb{N} \to A$ mapping an index n to the n-th element of the stream, as final coalgebras over the functor $X \mapsto A \times X$ (Rutten, 2000), using coinductive types (Geuvers, 1992), or observational models (Malcolm, 1997; Goguen & Malcolm, 2000; Roşu, 2000; Bidoit *et al.*, 2003). All these definitions are equivalent in the sense that the resulting coalgebras are isomorphic.

For the study of the model-theoretic semantics of equality, we focus on equational specifications of *bit*streams, i.e., systems of equations (partially) defining streams over the alphabet $\{0, 1\}$. Due to their simplicity, bitstreams can be embedded in almost every non-trivial coinductive structure. Specifications of bitstreams are inherently sorted, with a sort B for bits, and a sort S for bitstreams.

Convention 3.1. For notational and technical simplicity, from now on we let $\mathscr{S} = \{B, S\}$, noting that the subsequent results do not depend on this restriction.

3.1 Hidden Algebra Semantics

The philosophy of hidden algebra (Goguen, 1991; Malcolm, 1997; Goguen & Malcolm, 2000; Roşu, 2000) is that the objects of study are blackboxes whose contents are hidden, the only way to interact with them being by means of *experiments* using a subset of operations specially chosen for this purpose. Some of the experiments end with a visible result (an observable value), others with a hidden result (another blackbox). By systematically and iteratively applying all the available experiments to an object, we obtain the (typically infinite) *behavior* of that object. Two objects are *behaviorally equivalent* if and only if they cannot be distinguished by any of the available experiments.

In the case of streams, the blackboxes with hidden contents are the streams themselves, the visible values are the bits $\{0, 1\}$, and the experiments are the operations head and tail. Thus, the behaviors of a stream blackbox consist of precisely the bits corresponding to the infinite number of stream positions reachable with head and tail operations, and two streams are behaviorally equivalent if and only if they have the same bits corresponding to the same positions in them. We next particularize the hidden algebra notions to streams.

Definition 3.2. A *bitstream signature* Σ is an $\mathscr{S}$-sorted signature such that $0, 1, \mathsf{head}, \mathsf{tail} \in \Sigma$ where $0, 1 :: B$ are the usual bits and where $\mathsf{head} :: S \to B$, $\mathsf{tail} :: S \to S$ are the *stream destructors* giving the head and respectively the tail of a stream. A *bitstream specification* over Σ is a *finite* set E of equations over Σ.

Example 3.3. Define $\Sigma = \{0, 1, +, \mathsf{head}, \mathsf{tail}, \mathsf{A}, \mathsf{T}\}$ with $+ :: B \times B \to B$, $\mathsf{head} :: S \to B$, $\mathsf{tail}, \mathsf{T} :: S \to S$, and $\mathsf{A} :: S \times S \to S$. Then the following system of equations establishes a bitstream specification over Σ,

$$\begin{aligned} 0+0 &= 0 & 0+1 &= 1 \\ 1+1 &= 0 & 1+0 &= 1 \\ \mathsf{head}(\mathsf{A}(\sigma,\tau)) &= \mathsf{head}(\sigma) + \mathsf{head}(\tau) & \mathsf{tail}(\mathsf{A}(\sigma,\tau)) &= \mathsf{A}(\mathsf{tail}(\sigma), \mathsf{tail}(\tau)) \\ \mathsf{head}(\mathsf{T}(\sigma)) &= \mathsf{head}(\sigma) & \mathsf{tail}(\mathsf{T}(\sigma)) &= \mathsf{T}(\mathsf{A}(\sigma, \mathsf{tail}(\sigma))) \,. \end{aligned}$$

This is a modified version of a specification in (Endrullis *et al.*, 2013), where it is used to illustrate the method of circular coinduction by proving that the unary stream function T is an involution, that is, $\mathsf{T}(\mathsf{T}(\sigma)) = \sigma$.

We think of the stream sort S as *hidden*, in the sense that the exact representation of streams in models is irrelevant. The only way we can observe them is by means of observing their elements using the head and tail operations.

Definition 3.4. A *hidden algebra* $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ consists of

(i) an $\mathscr{S}$-sorted domain A where $A_B = \{0, 1\}$,
(ii) for every $f :: s_1 \times \cdots \times s_n \to s \in \Sigma$ an *interpretation* $[\![f]\!] : A_{s_1} \times \cdots \times A_{s_n} \to A_s$,
(iii) $0, 1 \in \Sigma$ with $[\![0]\!] = 0$ and $[\![1]\!] = 1$.

Definition 3.5. Let $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ be a hidden algebra, and $\alpha : \mathscr{X} \to A$ a variable assignment. The *interpretation of terms* $[\![\cdot]\!]_\alpha^{\mathscr{A}} : \mathit{Ter}(\Sigma, \mathscr{X}) \to A$ is defined inductively by:

$$[\![x]\!]_\alpha^{\mathscr{A}} = \alpha(x) \qquad\qquad [\![f(t_1, \ldots, t_n)]\!]_\alpha^{\mathscr{A}} = [\![f]\!]([\![t_1]\!]_\alpha^{\mathscr{A}}, \ldots, [\![t_n]\!]_\alpha^{\mathscr{A}})$$

We write $[\![\cdot]\!]_\alpha$ for $[\![\cdot]\!]_\alpha^{\mathscr{A}}$ whenever the algebra $\mathscr{A}$ is clear from the context. For ground terms $t \in \mathit{Ter}(\Sigma, \varnothing)$, we have $[\![t]\!]_\alpha = [\![t]\!]_\beta$ for all assignments α, β; we then write $[\![t]\!]$ for short.

Definition 3.6. Let $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ be a hidden algebra. We define *behavioral equivalence* $\equiv$ by

$$\sigma \equiv \tau \iff \forall n \in \mathbb{N}.\, [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(\sigma)) = [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(\tau)) \,,$$

for all $\sigma, \tau \in A_S$. On the domains corresponding to the other sorts in $\mathscr{S}$ (i.e., those different from S) we fix $\equiv$ to be the identity relation.

We say that $[\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(\sigma)$ is a $\langle \mathsf{head}, \mathsf{tail} \rangle$*-experiment on* σ. So $\sigma \equiv \tau$ iff σ and τ cannot be distinguished by any $\langle \mathsf{head}, \mathsf{tail} \rangle$-experiment.

Remark 3.7. We stress that A_S may contain elements $x, y \in A_S$ that are not equal ($x \neq y$) but have the same behavior ($x \equiv y$), see Example 3.9. Note that $\equiv$ is preserved by $[\![\mathsf{head}]\!]$ and $[\![\mathsf{tail}]\!]$, but that in general it does not need to be a congruence on A.

Definition 3.8. Let E be a bitstream specification over Σ. A hidden algebra $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ *behaviorally satisfies* E, or $\mathscr{A}$ is a *hidden model* of E, written $\mathscr{A} \models\!\equiv E$, if for every equation of E, the left- and right-hand sides are behaviorally equivalent:

$$[\![u]\!]_\alpha \equiv [\![v]\!]_\alpha \qquad\qquad \text{for every } u = v \text{ in } E \text{ and } \alpha : \mathscr{X} \to A$$

We say that an equation $s = t$ is *behaviorally satisfied in all hidden models of* E, written $E \models\!\equiv s = t$, if $\mathscr{A} \models\!\equiv E$ implies $\mathscr{A} \models\!\equiv s = t$ for every hidden algebra $\mathscr{A}$.

As we mentioned in Remark 3.7, behavioral equivalence in hidden models is not required to be a congruence. This property is useful if one wants to model certain aspects of non-deterministic behavior, as in the following example.

Example 3.9. The equation

$$\mathsf{tail}(\mathsf{push}(x)) = x \tag{3}$$

can be used to model an operation push that non-deterministically prepends a 0 or 1 to a stream. It admits a hidden model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ where $A_S = \{0,1\}^\omega \times \{0,1\}^\omega$, and

$$\begin{aligned}
[\![\mathsf{head}]\!](\langle a\sigma, \tau\rangle) &= a\\
[\![\mathsf{tail}]\!](\langle a\sigma, \tau\rangle) &= \langle \sigma, \tau\rangle\\
[\![\mathsf{push}]\!](\langle \sigma, b\tau\rangle) &= \langle b\sigma, \tau\rangle
\end{aligned}$$

for all $a,b \in \{0,1\}$ and $\sigma,\tau \in \{0,1\}^\omega$. Note that we have $\langle\sigma,\tau\rangle \equiv \langle\sigma',\tau'\rangle$ if and only if $\sigma = \sigma'$, and so there is a large number of behaviorally equivalent elements. The model employs the second component of the pair for generating random bits for the operation $[\![\mathsf{push}]\!]$. If $\equiv$ was required to be a congruence, then such non-deterministic behavior would not be possible. In particular, every model of (3) in which $\equiv$ is a congruence, satisfies

$$\mathsf{push}(\mathsf{tail}(\mathsf{push}(x))) = \mathsf{push}(x) \tag{4}$$

which does not allow for the intended non-determinism. Observe that (4) is not satisfied in the hidden model $\mathscr{A}$, for example

$$[\![\mathsf{push}]\!]([\![\mathsf{tail}]\!]([\![\mathsf{push}]\!](\langle\sigma,01\tau\rangle))) = \langle 1\sigma,\tau\rangle \;\;\not\equiv\;\; \langle 0\sigma, 1\tau\rangle = [\![\mathsf{push}]\!](\langle\sigma,01\tau\rangle)\,.$$

3.2 Behavioral Algebra Semantics

We now adapt the setup by requiring $\equiv$ to be a *congruence relation*, that is,

$$\sigma \equiv \tau \quad\Rightarrow\quad [\![f]\!](\ldots,\sigma,\ldots) \equiv [\![f]\!](\ldots,\tau,\ldots)$$

for any $f \in \Sigma$. (Here only the displayed argument of the function $[\![f]\!]$ is altered from σ to τ.) The resulting models are called *behavioral* in (Bidoit *et al.*, 2003).

Definition 3.10. A *behavioral algebra* is a hidden algebra where $\equiv$ is a congruence relation. For a bitstream specification E, we say that $s = t$ *is behaviorally satisfied in all behavioral models of* E, written $E \models\!\!\equiv_{\mathsf{bhv}} s = t$, if $\mathscr{A} \models\!\!\equiv E \Rightarrow \mathscr{A} \models\!\!\equiv s = t$ for every behavioral algebra $\mathscr{A}$.

3.3 Extensional Algebra Semantics

In behavioral algebras, elements that are behaviorally equivalent also behave the same when put in a context. As a consequence, we can identify elements with the same behavior. For convenience, we identify each element with the infinite binary sequence from $\{0,1\}^\omega$ that represents its observable behavior. This leads to the following definition of extensional algebras:

Definition 3.11. An *extensional algebra* $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ is a hidden algebra such that we have $A_S \subseteq \{0,1\}^\omega$ and for all $a \in A_B$ and $x \in A_S$: $[\![\mathsf{head}]\!](ax) = a$ and $[\![\mathsf{tail}]\!](ax) = x$.

For a bitstream specification E, we say that $s = t$ *is satisfied in all extensional models of* E, written $E \models\!\!\equiv_{\text{ext}} s = t$, if $\mathscr{A} \models\!\!\equiv E \Rightarrow \mathscr{A} \models\!\!\equiv s = t$ for every extensional algebra $\mathscr{A}$.

We stress that for extensional algebras, the domain is not required to contain all streams. The absence of such a requirement is natural, especially for hidden or behavioral algebras. For hidden and behavioral algebras, the domain is an arbitrary set A, and it is typically not required that for every possible observable behavior, there exists an element in the domain A that exhibits this behavior. As we will see, behavioral algebras are equivalent to extensional algebras. For this equivalence, it is important that extensional algebras are not required to contain all streams.

Remark 3.12. Why do we not require to restrict to ground models? Ground models are models where every element of the domain is an interpretation of a ground term, i.e., a term without variables. The reason is that $E \models s = t$ — equality in all models — is meaningful even if there are no ground terms at all. For example, we have

$$E \models \mathsf{zip}(x,x) = \mathsf{dup}(x)\,,$$

where the specification E consist of

$$\mathsf{zip}(a : x, y) = a : \mathsf{zip}(y, x) \qquad\qquad \mathsf{dup}(a : x) = a : a : \mathsf{dup}(x)\,.$$

The hardness proofs in the paper show that the problem of deciding $E \models s = t$ has the same complexity if s and t are restricted to be ground terms. Then it can be shown that the complexity does not change when restricting to ground models. The reason is that a countermodel (satisfying E but not $s = t$) remains a countermodel even if we restrict to interpretations of ground terms.

Lemma 3.13. *In every extensional algebra, behavioral equivalence $\equiv$ coincides with equality $=$, that is, $x = y$ if and only if $x \equiv y$.*

Proof. Let $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ be an extensional algebra. Fix $w, u \in A_S \subseteq \{0,1\}^\omega$. It suffices to show that $w \equiv u$ implies $w = u$. For a contradiction, assume $w \equiv u$ and $w \neq u$. Since $w, u \in \{0,1\}^\omega$ it follows that there exists $n \in \mathbb{N}$ such that $w(n) \neq u(n)$. By definition of extensional algebras, we have $[\![\mathsf{head}]\!](ax) = a$ and $[\![\mathsf{tail}]\!](ax) = x$, for all $a \in A_B$ and $x \in A_S$. By induction it follows that for all $x \in A_S$, we have $[\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(x)) = x(n)$. Hence $[\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(w)) = w(n) \neq u(n) = [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n(u))$, thus contradicting $w \equiv u$. ☐

As a consequence, behavioral satisfaction coincides with the usual notion of satisfaction where left- and right-hand sides of equations are required to have equal interpretations.

Lemma 3.14. An extensional algebra $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ satisfies a bitstream specification E if and only if $[\![u]\!]_\alpha = [\![v]\!]_\alpha$ for every $u = v$ in E and $\alpha : \mathscr{X} \to A$.

Proof. This lemma is a direct consequence of the definition of behavioral satisfaction for hidden algebras and Lemma 3.13. ☐

In the sequel, we will use Lemmas 3.13 and 3.14 tacitly. These two lemmas allow us to reason about extensional models of equational specifications without having to worry about distinct, but behaviorally equivalent elements in the model.

The following proposition states that extensional algebras are, for our purposes, equivalent to behavioral algebras, that is, hidden algebras for which behavioral equivalence $\equiv$ is a congruence.

Proposition 3.15. *We have $E \models_{\mathrm{bhv}} s = t$ if and only if $E \models_{\mathrm{ext}} s = t$. That is, an equation $s = t$ is behaviorally satisfied in all behavioral models of a specification E if and only if $s = t$ is satisfied in all extensional models of E.*

Proof. It suffices to show that the following properties are equivalent:

(i) There exists a behavioral algebra $\mathscr{A}$ such that $\mathscr{A} \models_{\equiv} E$ and $\mathscr{A} \not\models_{\equiv} s = t$.
(ii) There exists an extensional algebra $\mathscr{A}$ such that $\mathscr{A} \models_{\equiv} E$ and $\mathscr{A} \not\models_{\equiv} s = t$.

The implication (ii) $\Rightarrow$ (i) follows immediately, since every extensional algebra is a behavioral algebra.

For the implication (i) $\Rightarrow$ (ii), let $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ be a behavioral algebra, that is, behavioral equivalence $\equiv$ is a congruence. Let $\mathscr{A}/_{\equiv} = \langle A/_{\equiv}, [\![\cdot]\!]/_{\equiv}\rangle$ be the quotient algebra, i.e., $A/_{\equiv}$ is the set of congruence classes of A with respect to $\equiv$. For symbols $f \in \Sigma$ of arity n and $B_1, \dots, B_n \in A/_{\equiv}$, we define $[\![f]\!]/_{\equiv}(B_1, \dots, B_n) = B$ if $[\![f]\!](b_1, \dots, b_n) = b$ for $b_1 \in B_1, \dots, b_n \in B_n$, and B is the congruence class of b with respect to $\equiv$. The quotient algebra $\mathscr{A}/_{\equiv}$ is a behavioral algebra that, due to $\equiv$ being a congruence, behaviorally satisfies the same equations as $\mathscr{A}$. Let $\mathscr{A}'$ be the algebra obtained from $\mathscr{A}/_{\equiv}$ by renaming the domain elements into the streams they represent, that is, $a \in (\mathscr{A}/_{\equiv})_S$ becomes $[\![\mathsf{head}]\!](a) : [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!](a)) : \dots$. Then $[\![:]\!](x, \sigma) = x\sigma$, since in $\mathscr{A}/_{\equiv}$ every stream has a unique representative in the model. Hence, $\mathscr{A}'$ is an extensional algebra. Moreover, for elements a, b of the domain of $\mathscr{A}/_{\equiv}$, we have $a \equiv b$ iff $a = b$. Hence, $\mathscr{A}'$ is a model of an equation $s = t$ if and only if $s = t$ is behaviorally satisfied in $\mathscr{A}$. □

3.4 Full Extensional Algebra Semantics

For extensional algebras, the domain is a *subset* of $\{0,1\}^\omega$. While strongly motivated by the desired relationships with the hidden and behavioral models, once streams are identified with their behaviors in $\{0,1\}^\omega$ it is quite natural to drop the *subset* restriction and thus require all extensional models to have a fixed carrier, the set of all possible streams (or stream behaviors). Then each model only provides particular interpretations of the operations in Σ over all possible streams.

Definition 3.16. We say that an extensional algebra $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ is *full* if its domain contains all bitstreams, $A_S = \{0,1\}^\omega$. For a bitstream specification E, we say that $s = t$ *is satisfied in all full extensional models of* E, written $E \models_{\mathrm{full}} s = t$, if $\mathscr{A} \models_{\equiv} E \Rightarrow \mathscr{A} \models_{\equiv} s = t$ for every full extensional algebra $\mathscr{A}$.

The difference between extensional and full extensional models may seem insignificant, but as shown in Section 6.3, requiring the domains to contain all streams yields a huge jump in the complexity of deciding stream equality.

Remark 3.17. We briefly discuss the differences between hidden models, extensional (or equivalently behavioral) models and full extensional models. In Example 3.9 we have seen that hidden models can be used for modeling non-determinism. As explained, this is not possible with behavioral or extensional models.

Let us compare extensional and full extensional models. In extensional models, the domain is not required to contain all streams. This allows us to write equational specifications that restrict the possible domains. For example, consider the following specification:

$$\begin{aligned}
\mathsf{uc}(x, \mathsf{count}(x)) &= \mathsf{ones} && (5)\\
\mathsf{uc}(x, \mathsf{ones}) &= \mathsf{zeros} && (6)\\
\mathsf{uc}(a : y, 1 : y) &= \mathsf{uc}(x, y) && (7)\\
\mathsf{uc}(0 : 0 : x, 0 : y) &= \mathsf{uc}(0 : x, 0 : y) && (8)\\
\mathsf{uc}(1 : 1 : x, 0 : y) &= \mathsf{uc}(1 : x, 0 : y) && (9)\\
\mathsf{uc}(0 : 1 : x, 0 : y) &= \mathsf{zeros} && (10)\\
\mathsf{uc}(1 : 0 : x, 0 : y) &= \mathsf{zeros} && (11)
\end{aligned}$$

In every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ of this specification, the domain contains only streams that are ultimately constant, that is, for every $\sigma \in A_S$ there exists $n \in \mathbb{N}$ such that $[\![\mathsf{tail}]\!]^n(\sigma) \in \{0^\omega, 1^\omega\}$. We sketch the proof of this fact. The idea is that for every stream $\sigma \in A_S$, the index of the first 0 in $[\![\mathsf{count}]\!](\sigma)$ determines the maximum length of the non-constant prefix of σ. Let $\sigma \in A_S$ be a stream in the domain. If $[\![\mathsf{count}]\!](\sigma)$ contains no 0, then (5) and (6) yield an immediate contradiction. Let $z \in \mathbb{N}$ be the index of the first 0 in $[\![\mathsf{count}]\!](\sigma)$. Then from (7) we can derive that $[\![\mathsf{uc}]\!](\sigma, [\![\mathsf{count}]\!](\sigma)) = [\![\mathsf{uc}]\!]([\![\mathsf{tail}]\!]^z(\sigma), [\![\mathsf{tail}]\!]^z([\![\mathsf{count}]\!](\sigma)))$. Moreover, by the choice of z, the first element of $[\![\mathsf{tail}]\!]^z([\![\mathsf{count}]\!](\sigma))$ is a 0. Then (8)–(11) ensure that the stream $[\![\mathsf{tail}]\!]^z(\sigma)$ is constant. In particular, equations (10) and (11) yield a contradiction with (5) if the stream contains any occurrence of 01 or 10.

Clearly, the above specification has no full extensional model. However, it is always possible to 'simulate' non-full extensional models using full extensional models. That is, for every equational specification E over Σ we can construct an equational specification E' over the extended signature $\Sigma \cup \{\mathsf{dom}\}$ with $\mathsf{dom} :: S \to S$, satisfying

(i) for every extensional model $\mathscr{A} \models E$ there exists a full extensional model $\mathscr{A}' \models E'$ such that $[\![\mathsf{f}]\!]^{\mathscr{A}} = [\![\mathsf{f}]\!]^{\mathscr{A}'}$ for all $\mathsf{f} \in \Sigma$, and
(ii) for every full extensional model $\mathscr{A}' \models E'$ there exists an extensional model $\mathscr{A} \models E$ such that $[\![\mathsf{f}]\!]^{\mathscr{A}} = [\![\mathsf{f}]\!]^{\mathscr{A}'}$ for all $\mathsf{f} \in \Sigma$.

Let us sketch the construction of E'. We use the range of $[\![\mathsf{dom}]\!]$ to model the domain of an arbitrary extensional model. We then need to modify the equations in E by replacing each variable x by the term $\mathsf{dom}(x)$:

$$x \mapsto \mathsf{dom}(x)\,.$$

This forces that the equations hold precisely for the elements in the range of $[\![\mathsf{dom}]\!]$. Moreover, we need to extend E with equations

$$\mathsf{f}(\mathsf{dom}(x_1), \dots, \mathsf{dom}(x_n)) = \mathsf{dom}(\mathsf{f}(\mathsf{dom}(x_1), \dots, \mathsf{dom}(x_n)))$$

for every $\mathsf{f} \in \Sigma$ where n is the arity of f. These equations guarantee that the range $[\![\mathsf{dom}]\!]$ is closed under all functions $[\![\mathsf{f}]\!]^{\mathscr{A}}$ with $\mathsf{f} \in \Sigma$. Actually, for these equations to work, we assume without loss of generality that $[\![\mathsf{dom}]\!](\sigma) = \sigma$ for every σ in the range of $[\![\mathsf{dom}]\!]$. The resulting specification E' fulfills the properties (i) and (ii).

Let us further mention that for certain purposes, full models are the natural choice. For example, the equation

$$\mathsf{g}(\mathsf{f}(x)) = x$$

can be used to ensure that $[\![g]\!]$ is the inverse function of $[\![f]\!]$. However, if we interpret this equation in arbitrary (non-full) extensional models, then $[\![g]\!]$ and $[\![f]\!]$ are inverses of each other only for the stream in the domain of the model which could be as small as $\{0^\omega\}$.

(Full) Extensional models are not necessarily unique. The reason is that there can underspecified elements or functions, as illustrated by the following example.

Example 3.18. The specification

$$\mathsf{even}(\mathsf{A}) = \mathsf{zeros} \qquad \mathsf{zeros} = 0 : \mathsf{zeros} \qquad \mathsf{even}(x : y : \sigma) = x : \mathsf{even}(\sigma)$$

admits many non-isomorphic full extensional models. The interpretation of the constant $\mathsf{A} :: S$ can be any stream of the form $0 : x_1 : 0 : x_2 : 0 : x_3 : \cdots$.

Definition 3.19. Let E be a bitstream specification over Σ, and $s,t \in Ter(\Sigma, \mathscr{X})$ with $s,t :: S$. Then s and t are said to be

(i) *equal in all hidden models of E* if $E \models\!\equiv s = t$,
(ii) *equal in all behavioral models of E* if $E \models\!\equiv_{\mathrm{bhv}} s = t$,
(iii) *equal in all extensional models of E* if $E \models\!\equiv_{\mathrm{ext}} s = t$, and
(iv) *equal in all full extensional models of E* if $E \models\!\equiv_{\mathrm{full}} s = t$.

3.5 Equality of Sets of Solutions

We define the set of solutions of a term s in a specification E, that is, the set of all possible interpretations $[\![s]\!]$ of s in all models of E.

Definition 3.20. Let E be a bitstream specification over Σ, and $s \in Ter(\Sigma, \varnothing)$ with $s :: S$. Then the set of

(i) *solutions of s in E over all extensional models* is
$$[\![s]\!]_{E,\mathrm{ext}} = \{\, [\![s]\!]^{\mathscr{A}} \mid \mathscr{A} \models\!\equiv E \,\}\,.$$
(ii) *solutions of s in E over all full extensional models* is
$$[\![s]\!]_{E,\mathrm{full}} = \{\, [\![s]\!]^{\mathscr{A}} \mid \mathscr{A} \text{ full}, \mathscr{A} \models\!\equiv E \,\}\,.$$

Remark 3.21. Note that in Definition 3.20 we define the set of solutions only for ground terms $s \in Ter(\Sigma, \varnothing)$. For terms with variables there are different choices as to what the set of solutions should be. An obvious choice is to take the set of solutions as the interpretations in all models, under all possible interpretations of the variables, that is,

$$[\![s]\!]_{E,\mathrm{ext}} = \{\, [\![s,\alpha]\!]^{\mathscr{A}} \mid \mathscr{A} \models\!\equiv E,\ \alpha : \mathscr{X} \to A \,\}\,,$$

where $s \in Ter(\Sigma, \mathscr{X})$ is a term with variables. This definition can easily be reduced to sets of solutions of ground terms as follows. For every variable $x \in Var(s)$ we introduce a fresh constant x'. Let the substitution σ be defined by $\sigma(x) = x'$ if $x \in \mathit{Var}(s)$, and $\sigma(x) = x$ otherwise. Then the set of solutions of s is equal to the set of solutions of the ground term $s\sigma$, that is:

$$[\![s]\!]_{E,\mathrm{ext}} = [\![s\sigma]\!]_{E,\mathrm{ext}}$$

Let $\mathscr{A}$ be such that $\mathscr{A} \models E$ and let $\alpha : \mathscr{X} \to A$. Then $[\![s,\alpha]\!]^{\mathscr{A}} = [\![s\sigma]\!]^{\mathscr{A}'}$ where $\mathscr{A}'$ is obtained from $\mathscr{A}$ by defining $[\![x']\!] = \alpha(x)$ for all $x \in \mathit{Var}(s)$. Reversely, $[\![s\sigma]\!]^{\mathscr{A}} = [\![s,\alpha]\!]^{\mathscr{A}}$ where α is defined by $\alpha x = [\![x']\!]$ for all $x \in \mathit{Var}(s)$.

Another possible choice for the set of solutions of terms with variables is to consider the terms as functions depending on the assignment of the variables. Then the domain of these functions depends on the domain of the model. Thus for a sensible comparison of these functions it seems useful to restrict to full models.

Note that we have given the definition of solutions only for extensional models. Clearly, we want the set of solutions of a stream term to be a set of streams. In hidden and behavioral models, the domain A is an arbitrary set, and thus the interpretation is an arbitrary element. In order to generalize the definition to hidden algebras one needs to consider the set of streams that represent all possible behaviors of s in hidden models of the specification E.

Definition 3.22. Let E_s and E_t be bitstream specifications over Σ_s and Σ_t, respectively. Let $s \in \mathit{Ter}(\Sigma_s, \varnothing)$ and $t \in \mathit{Ter}(\Sigma_t, \varnothing)$. Then s and t are said to have

(i) *equal solutions over all extensional models*, when $[\![s]\!]_{E_s,\mathrm{ext}} = [\![t]\!]_{E_t,\mathrm{ext}}$, and
(ii) *equal solutions over all full extensional models*, when $[\![s]\!]_{E_s,\mathrm{full}} = [\![t]\!]_{E_t,\mathrm{full}}$.

Definition 3.23. Let E be a bitstream specification over Σ, and $s \in \mathit{Ter}(\Sigma, \varnothing)$. Then s is said to have

(i) *exactly one solution over all extensional models*, if $|[\![s]\!]_{E,\mathrm{ext}}| = 1$,
(ii) *exactly one solution over all full extensional models*, if $|[\![s]\!]_{E,\mathrm{full}}| = 1$,
(iii) *at least one solution over all extensional models*, if $|[\![s]\!]_{E,\mathrm{ext}}| \geq 1$,
(iv) *at least one solution over all full extensional models*, if $|[\![s]\!]_{E,\mathrm{full}}| \geq 1$,
(v) *at most one solution over all extensional models*, if $|[\![s]\!]_{E,\mathrm{ext}}| \leq 1$,
(vi) *at most one solution over all full extensional models*, if $|[\![s]\!]_{E,\mathrm{full}}| \leq 1$.

4 Turing Machines as Equational Specifications

For defining streams and operations on streams we will frequently employ the *stream constructor* ':', a distinguished symbol of the signature Σ which we write *infix*. We tacitly assume that specifications include the following equations

$$\mathsf{head}(a : x) = a \qquad\qquad \mathsf{tail}(a : x) = x \tag{12}$$

These equations fully define the interpretation $[\![:]\!]$ of the stream constructor, that is, in every *extensional* model $\langle A, [\![\cdot]\!]\rangle$ of (12) we have, for all $a \in A_B$ and $x \in A_S$,

$$[\![:]\!](a,x) = ax\,.$$

Remark 4.1. For hidden models, the equations (12) do not define the interpretation $[\![:]\!]$ uniquely, but up to behavioral equivalence $\equiv$ (even if the interpretations $[\![\mathsf{head}]\!]$ and $[\![\mathsf{tail}]\!]$ are fixed). Actually, Section 5 is independent of the equations (12).

We define a set of standard equations (for bitstream specifications) that will be used throughout the paper.

Definition 4.2. Fix an integer $m \geq 0$. Let $\Sigma_m = \{0, 1, :, \mathsf{zeros}, \mathsf{ones}\} \cup \{\mathsf{zip}_k \mid 1 \leq k \leq m\}$, with $\mathsf{zip}_k :: S^k \to S$. We define the specification Z_m over Σ_m to consist of the following equations, for all k such that $1 \leq k \leq m$,

$$\begin{aligned}
\mathsf{zeros} &= 0 : \mathsf{zeros} & \mathsf{zip}_1(x_1) &= x_1 \\
\mathsf{ones} &= 1 : \mathsf{ones} & \mathsf{zip}_2(a : x_1, x_2) &= a : \mathsf{zip}_2(x_2, x_1) \\
& & \mathsf{zip}_k(x_1, \ldots, x_k) &= \mathsf{zip}_2(x_1, \mathsf{zip}_{k-1}(x_2, \ldots, x_k)) \qquad (k > 2)\,.
\end{aligned}$$

If for some $m \geq 2$ the symbol zip_m occurs in a specification S then S (implicitly) contains the equations $\mathsf{zip}_k(\ldots) = \ldots$ for all $k \leq m$, that is, $Z_m \subseteq S$.

To give an example,

$$[\![\mathsf{zip}_3]\!](x, y, z) = x(0)\,y(0)\,x(1)\,z(0)\,x(2)\,y(1)\,x(3)\,z(1)\,x(4)\,y(2)\,\cdots\,.$$

The function $[\![\mathsf{zip}_k]\!]$ was also used in the work (Endrullis *et al.*, 2011a), which introduces a new approach to comparing the complexity of streams on the basis of reducibility via finite state transducers.

Lemma 4.3. *Fix $m \in \mathbb{N}$, and let S be a specification with $Z_m \subseteq S$ (see Definition 4.2). Then for every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ of S we have*

$$\begin{aligned}
[\![\mathsf{zeros}]\!](n) &= 0 & [\![\mathsf{zip}_1]\!](x_1)(n) &= x_1(n) && (13)\\
[\![\mathsf{ones}]\!](n) &= 1 & [\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2n) &= x_1(n) && (14)\\
& & [\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2n+1) &= [\![\mathsf{zip}_{k-1}]\!](x_2, \ldots, x_k)(n) && (15)
\end{aligned}$$

for all k such that $2 \leq k \leq m$, $x_1, \ldots, x_k \in A_S$, and $n \in \mathbb{N}$. In closed form this is

$$\begin{aligned}
[\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2^{i-1}(2n+1) - 1) &= x_i(n), \qquad (1 \leq i < k) && (16)\\
[\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2^{k-1}(n+1) - 1) &= x_k(n)\,. && (17)
\end{aligned}$$

Proof. For streams $x \in A_S$ we write $x = x(0)x(1)\cdots$ and $x' = x(1)x(2)\cdots$. First note that $[\![\mathsf{zip}_2]\!](x,y)(2n) = x(n)$ and $[\![\mathsf{zip}_2]\!](x,y)(2n+1) = y(n)$ follow by induction on n; distinguish three cases: $[\![\mathsf{zip}_2]\!](x,y)(0) = x(0)$, $[\![\mathsf{zip}_2]\!](x,y)(2n+1) = [\![\mathsf{zip}_2]\!](y,x')(2n) = y(n)$, and $[\![\mathsf{zip}_2]\!](x,y)(2n+2) = [\![\mathsf{zip}_2]\!](y,x')(2n+1) = x'(n) = x(n+1)$. From this and the definition of zip_k, equations (14) and (15) follow directly.

Next, we prove (16) and (17) by induction on $k \geq 2$. The base case $k = 2$ boils down to (14) and (15). So let $k > 2$. The case $i = 1$ is just (14) again. For $1 < i < k$ we reason $[\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2^{i-1}(2n+1) - 1) = [\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2(2^{i-2}(2n+1) - 1) + 1) = [\![\mathsf{zip}_{k-1}]\!](x_2, \ldots, x_k)(2^{i-2}(2n+1) - 1)$ by (15), which is equal to $x_i(n)$ by the induction hypothesis. Similarly, $[\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2^{k-1}(n+1) - 1) = [\![\mathsf{zip}_k]\!](x_1, \ldots, x_k)(2(2^{k-2}(n+1) - 1) + 1) = [\![\mathsf{zip}_{k-1}]\!](x_2, \ldots, x_k)(2^{k-2}(n+1) - 1) = x_k(n)$. $\square$

We define a translation of Turing machines to equational specifications of bitstream functions, based on the standard translation to term rewriting systems from (Terese, 2003). However, we represent the tape to the left and to the right of the head using streams instead of finite lists, and have one instead of four rules for 'extending' the tape. In particular, the equation for extending the tape is the equation for zeros from Definition 4.2. The terms of the shape $q(x,y)$ represent configurations of the Turing machine, where the stream y contains the tape content below and right of the head, and x the tape content left of the head *backwards*. So the head of the machine stands on the first symbol of y.

Definition 4.4. Let $T = \langle Q, q_0, \delta \rangle$ be a Turing machine. We define the signature $\Sigma = \{0, 1, :, \mathsf{zeros}\} \cup Q$ where the symbols $q \in Q$ have type $q :: S \times S \to B$. We define the specification E_T over Σ to consist of the following equations:

$$\begin{aligned} \mathsf{zeros} &= 0 : \mathsf{zeros} \\ q(a : x, b : y) &= q'(x, a : b' : y) && \text{for every } \delta(q,b) = \langle q', b', L \rangle, \\ q(x, b : y) &= q'(b' : x, y) && \text{for every } \delta(q,b) = \langle q', b', R \rangle, \\ q(x, b : y) &= b && \text{whenever } \delta(q,b) \text{ is undefined.} \end{aligned}$$

We write $E_T \models s = t$ to denote that $s = t$ can be derived from E_T by standard equational reasoning. Moreover, we use R_T to denote the term rewriting system obtained from E_T by orienting all equations from left to right.

Proposition 4.5. *A Turing machine $T = \langle Q, q_0, \delta \rangle$ halts on input w with output b if and only if $E_T \models q_0(\mathsf{zeros}, w_1 : w_2 : \ldots : w_{|w|} : \mathsf{zeros}) = b$.*

Proof. Let $U = \mathit{Ter}(\{0, 1, :, \mathsf{zeros}\}, \varnothing)_S$, the set of terms of the form $a_0 : a_1 : \ldots : a_{k-1} :$ zeros, with $k \in \mathbb{N}$ and $a_i \in \{0,1\}$, $0 \le i < k$. Define $\sim \,=\, \leftrightarrow^*_{\mathsf{zeros}}$, the equivalence relation induced by the rule zeros $\to 0$: zeros (which is in R_T). We define a map $F : U \times U \to \{0,1\}^{\mathbb{Z}}$ as follows. For $s = a_0 : \cdots : a_{n_s - 1}$: zeros and $t = b_0 : \cdots : b_{n_t - 1}$: zeros, set $F(s,t)(x) = a_{|x|-1}$ if $-n_s \le x < 0$, $F(s,t)(x) = b_x$ if $0 \le x < n_t$, and $F(s,t)(x) = 0$ for all other x. Clearly F is invariant under $\sim$, i.e., $F(s,t) = F(s',t')$ iff $s \sim s'$ and $t \sim t'$. The initial tape τ_0 with input w is (not uniquely) represented by $\tau_0 = F(s_0, t_0)$ with $s_0 =$ zeros and $t_0 = w_1 : w_2 : \cdots : w_{|w|}$: zeros. If T halts on input w, then all tapes τ_i in the maximal sequence $\langle q_0, \tau_0 \rangle \to_T \langle q_1, \tau_1 \rangle \to_T \cdots \to_T \langle q_n, \tau_n \rangle$ have a finite non-zero part, and can thus be represented by $F(s_i, t_i)$ for some $s_i, t_i \in U$, $0 \le i \le n$. Equally clear is that in the rewrite sequence $q_0(s_0, t_0) \to_{R_T/\sim} q_1(s_1, t_1) \to_{R_T/\sim} \cdots \to_{R_T/\sim} b$ all stream terms s_i, t_i are in the set U. Here, for relations R, E, we write R/E for the composed relation $E^* R E^*$. The reason for reasoning modulo $\sim$ is that it is sometimes required to 'extend the tape', i.e., unfold the constant zeros, in order for a rule $q(\ldots) \to q'(\ldots)$ to be applicable.

Now to prove the claim, it suffices to show that for all $q, q' \in Q$ and $s, t, s', t' \in U$,

$$\begin{aligned} \langle q, F(s,t) \rangle \to_T \langle q', F(s',t') \rangle \quad &\iff \quad q(s,t) \to_{R_T/\sim} q'(s',t')\text{, and} \\ \langle q, F(s,t) \rangle \not\to_T \quad &\iff \quad q(s,t) \to_{R_T/\sim} F(s,t)(0)\,. \end{aligned}$$

Let $\tau = F(s,t)$, and $\tau' = F(s',t')$, and assume $\langle q, \tau \rangle \to_T \langle q', \tau' \rangle$. We distinguish two cases. If $\delta(q, \tau(0)) = \langle q', d, L \rangle$, then $s \sim a : u$ and $t \sim \tau(0) : v$ for some $a \in \{0,1\}$ and $u, v \in U$, and it can be verified that $\tau' = F(u, a : d : v)$. Hence $q(s,t) \to_{R_T/\sim} q'(s',t')$, since $s' \sim u$, and $t' \sim a : d' : v$. For the case $\delta(q, \tau(0)) = \langle q', d, R \rangle$, we pick v such that $t \sim \tau(0) : v$, and check $\tau' = F(d : s, v)$, whence $q(s,t) \to_{R_T/\sim} q'(s',t')$ follows from $s' \sim d : s$ and $t' \sim v$. If $\langle q, \tau \rangle$ is a halting configuration, the output is $\tau(0)$. For a suitably chosen v with $v \sim t$ (i.e., unfold zeros if $t =$ zeros), we get $q(s,t) \sim q(s,v) \to_{R_T} \tau(0)$.

On the other hand, given $q(s,t) \to_{R_T/\sim} q'(s',t')$, the step $\langle q, F(s,t)\rangle \to_T \langle q', F(s',t')\rangle$ follows directly from the definitions and invariance of F under $\sim$. Likewise so for the halting case $q(s,t) \to_{R_T/\sim} F(s,t)(0)$. □

Notation 4.6. For an integer $n \geq 0$ we use $\underline{n}$ to denote the stream term

$$\underline{n} = \underbrace{1 : \cdots : 1}_{n \text{ times}} : \mathsf{zeros}\,.$$

For a set $\xi \subseteq \mathbb{N}$, we write $\underline{\xi}$ to denote the infinite stream term

$$\underline{\xi} = \chi_\xi(0) : \chi_\xi(1) : \chi_\xi(2) : \ldots\,,$$

where $\chi_\xi : \mathbb{N} \to \{0,1\}$ is the characteristic function of ξ, i.e., $\chi_\xi(n) = 1$ iff $n \in \xi$.

As said before, in contrast to the encoding of Turing machines as in (Terese, 2003), we do not have equations for extending the tape. We use $q(s,t)$ only for infinite stream terms s,t or finite terms s,t that evaluate to fully defined streams in the limit.

Apart from the additional rule for termination and the lack of rules for extending the tape, the translation R_T is standard, and the rewrite rules model the transition relation of Turing machines in a one-to-one fashion. We consider the rewrite system R_T as the operational semantics of the Turing machine T, and we define concepts like input and oracles directly on the term representations. We pass k-tuples $\langle n_1, \ldots, n_k\rangle \in \mathbb{N}^k$ of natural numbers as input to a Turing machine by choosing the following start configuration:

$$q_0(\mathsf{zeros}, \mathsf{zip}_{k+1}(\underline{k}, \underline{n_1}, \ldots, \underline{n_k}))\,,$$

The particular encoding of tuples is not important. For equational specifications the encoding $\mathsf{zip}_{k+1}(\underline{k}, \underline{n_1}, \ldots, \underline{n_k})$ is more convenient than the Gödel encoding. Yet another possibility would be the encoding $(1 :)^k 0 : (1 :)^{n_1} 0 : \ldots : (1 :)^{n_k} 0 : \mathsf{zeros}$. We have opted for the zip operation simply because we use this encoding also for writing multiple oracles (which are represented as infinite stream terms) on the tape.

We obtain Turing machines with oracles $\xi_1, \ldots, \xi_m \subseteq \mathbb{N}$ by writing the oracles elementwise interlaced on the tape left of the head, as follows:

Definition 4.7. Let $T = \langle Q, q_0, \delta\rangle$ be a Turing machine. For stream terms $\xi_1, \ldots, \xi_m :: S$ and $n_1, \ldots, n_k :: S$, we define

$$T(\xi_1, \ldots, \xi_m; n_1, \ldots, n_k) := q_0(\mathsf{zip}_m(\xi_1, \ldots, \xi_m), \mathsf{zip}_{k+1}(\underline{k}, n_1, \ldots, n_k))$$

Here the intuition is as follows: The stream terms $\xi_1, \ldots, \xi_m$ represent oracles, and the stream terms $n_1, \ldots, n_k$ represent natural numbers. The term $T(\xi_1, \ldots, \xi_m; n_1, \ldots, n_k)$ corresponds to the Turing machine T started with m oracles $\xi_1, \ldots, \xi_m$ and k input numbers $n_1, \ldots, n_k$. While the terms $\xi_1, \ldots, \xi_m$ may be infinite, we only consider finite sequences of rewrite steps (corresponding to a finite number of transition steps of the Turing machine).

Note that for passing the tuple of natural numbers $\langle n_1, \ldots, n_k\rangle$ to the Turing machine, we include the length k of the tuple within the interleaving zip_{k+1}. In this way, the Turing machine can determine how many arguments have been passed. For the oracles $\xi_1, \ldots, \xi_m$ we have chosen for not passing m since for the Turing machines in this paper we do not need a variable number of oracles.

Definition 4.8. A Turing machine $T = \langle Q, q_0, \delta \rangle$ *halts (with output* $b \in \{0,1\}$*)* on inputs $n_1, \ldots, n_k \in \mathbb{N}$ with oracles $\xi_1, \ldots, \xi_m \subseteq \mathbb{N}$ if there is a rewrite sequence

$$T(\underline{\xi_1}, \ldots, \underline{\xi_m}; \underline{n_1}, \ldots, \underline{n_k}) \to_{R_T}^{*} b \,.$$

We note that due to the rules for zip_n and zeros, there are infinite rewrite sequences even if the Turing machine halts. However, R_T is orthogonal and therefore outermost-fair rewriting is a normalizing strategy, see (Ketema & Simonsen, 2010). That is, outermost-fair rewriting using R_T computes the normal form $b \in \{0,1\}$ if it exists. This normal form is unique by (Klop & de Vrijer, 2005; Endrullis *et al.*, 2010b).

Definition 4.9. A predicate $P \subseteq \wp(\mathbb{N})^m \times \mathbb{N}^k$ is *decidable* if there is a Turing machine T such that for all $\vec{\xi} \in \wp(\mathbb{N})^m$ and $\vec{n} \in \mathbb{N}^k$ we have that T halts on input $\vec{n}$ with oracles $\vec{\xi}$, and the output is 1 if and only if $\langle \vec{\xi}, \vec{n} \rangle \in P$.

We note that the above definition is independent of our particular variant of Turing machines, and of the encoding of inputs and oracles. It can easily be seen to correspond with the standard definitions in (Rogers, Jr., 1967; Shoenfield, 1971).

In correspondence with Definition 4.7 we define for a Turing machine $T = \langle Q, q_0, \delta \rangle$, and streams $\xi_1, \ldots, \xi_m, n_1, \ldots, n_k \in \{0,1\}^\omega$,

$$[\![T]\!](\xi_1, \ldots, \xi_m; n_1, \ldots, n_k) := [\![q_0]\!]([\![\mathsf{zip}_m]\!](\xi_1, \ldots, \xi_m), [\![\mathsf{zip}_{k+1}]\!]([\![\underline{k}]\!], n_1, \ldots, n_k)) \,.$$

Then for the models of Turing machine specifications we have:

Lemma 4.10. *Let* $P \subseteq \wp(\mathbb{N})^m \times \mathbb{N}^k$ *be decidable, and let* $T = \langle Q, q_0, \delta \rangle$ *be the corresponding Turing machine. Then in every extensional model* $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ *of a specification including the equations* $Z_{\max(m,k+1)}$ *from Definition 4.2 and* E_T*, we have for every* $\vec{\xi} \in \wp(\mathbb{N})^m$ *and* $\vec{n} \in \mathbb{N}^k$*:* $(\vec{\xi}, \vec{n}) \in P$ *if and only if* $[\![T]\!](\xi_1, \ldots, \xi_m; n_1, \ldots, n_k) = 1$.

Proof. By assumption P is decidable, hence $T(\underline{\xi_1}, \ldots, \underline{\xi_m}; \underline{n_1}, \ldots, \underline{n_k})$ has a normal form in $\{0,1\}$, and the normal form is 1 if and only if $(\vec{\xi}, \vec{n}) \in P$. □

5 Equality in Hidden Models

As seen in Section 3, the hidden Σ-algebra stream models are the most relaxed models that we consider in this paper, in that any of the other models are obtained by adding restrictions to hidden Σ-algebras. In this section we study the complexity of deciding whether $E \models\!\equiv e$ with hidden model semantics, where E is a finite bitstream specification and e is a stream equation $s = t$. Specifically, we show that this problem has the lowest complexity among all the similar problems discussed in the sequel: Π_2^0-complete.

The result below was first claimed in (Roşu, 2006), but in a slightly more involved stream hidden algebraic setting, where the bitstream signatures Σ were required to also include the stream constructor ':' and the bitstream specifications to include the equations $\mathsf{head}(b : x) = b$ and $\mathsf{tail}(b : x) = x$. However, as noted in (Endrullis *et al.*, 2012b), the lack of congruence of the behavioral equivalence relation interacted unexpectedly with the ':' construct in some examples, so we have reworked the result below to eliminate the dependence of ':' and its special properties in the proof. The Π_2^0-hardness part still uses a ':' operation symbol, but it defines it like any other symbol, without any special properties.

The membership in Π^0_2 follows by framing the problem as follows: *for any* $n \in \mathbb{N}$, *there exists* a first-order logic (FOL) proof of $E \vdash \mathsf{head}(\mathsf{tail}^n(s)) = \mathsf{head}(\mathsf{tail}^n(t))$, where the proof checking part is decidable. This is possible thanks to the completeness of FOL. The Π^0_2-hardness is shown by reduction from the totality problem (see Proposition 2.13).

Theorem 5.1. *The following problem is* Π^0_2*-complete:*

INPUT: *Bitstream specification* E*, terms* $s,t :: S$*.*

QUESTION: *Is* $s = t$ *behaviorally satisfied in all hidden models of* E*?*

Proof. Let us first show the membership in Π^0_2. To do that, we need to prove some properties relating behavioral satisfaction to first-order logic satisfaction. Let E be a bitstream specification and let E_B be the potentially infinite $\text{FOL}_=$ specification (i.e., it can have infinitely many finite formulae) defined as follows:

– add to E_B all the equations in E of sorts different from S;

– add for each equation $u = v$ of sort S in E a decidable (or recursive) set of equations of sort B, namely $\{\mathsf{head}(\mathsf{tail}^n(u)) = \mathsf{head}(\mathsf{tail}^n(v)) \mid n \in \mathbb{N}\}$;

– add the formulae $\neg(0 = 1)$ and $b = 0 \vee b = 1$, where b is a variable of sort B; these are the only non-equational formulae in E_B.

Then the following hold:

(1) For any algebra $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$, $\mathscr{A} \models E_B$ iff $\mathscr{A}^{\{0,1\}}$ is a hidden model of streams and $\mathscr{A}^{\{0,1\}} \models\!\!\equiv E$, where $\mathscr{A}^{\{0,1\}}$ is $\mathscr{A}$ whose elements A_0 and A_1 in A_B are renamed by 0 and 1, respectively, and $\models$ is the standard satisfaction in $\text{FOL}_=$;

(2) If e is an equation of sort B, then $E \models\!\!\equiv e$ iff $E_B \models e$ (the latter is $\text{FOL}_=$ entailment).

To show (1), first note that for any algebra $\mathscr{A}$, $\mathscr{A}$ satisfies the two non-equational formulae $\neg(0 = 1)$ and $b = 0 \vee b = 1$ in E_B if and only if the carrier A_B of $\mathscr{A}$ has precisely two elements and those correspond to the constant operations 0 and 1, if and only if $\mathscr{A}^{\{0,1\}}$ is a hidden model of streams. Also, note that in this case, if e is some equation of sort different from S in E, then $\mathscr{A} \models e$ if and only if $\mathscr{A}^{\{0,1\}} \models\!\!\equiv e$; that is because the behavioral equivalence relation is identity on all sorts different from S. All that is left to prove now is that for any equation $u = v$ of sort S in E, it is the case that $\mathscr{A} \models \{\mathsf{head}(\mathsf{tail}^n(u)) = \mathsf{head}(\mathsf{tail}^n(v)) \mid n \in \mathbb{N}\}$ if and only if $\mathscr{A}^{\{0,1\}} \models\!\!\equiv u = v$. Note that $[\![\mathsf{head}(\mathsf{tail}^n(u))]\!]_\alpha = [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n([\![u]\!]_\alpha))$ for any mapping $\alpha : \mathscr{X} \to A$ and for any $n \in \mathbb{N}$, and similarly for v. Therefore, $\mathscr{A} \models \{\mathsf{head}(\mathsf{tail}^n(u)) = \mathsf{head}(\mathsf{tail}^n(v)) \mid n \in \mathbb{N}\}$ if and only if $[\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n([\![u]\!]_\alpha)) = [\![\mathsf{head}]\!]([\![\mathsf{tail}]\!]^n([\![v]\!]_\alpha))$ for any $n \in \mathbb{N}$, if and only if (by Definition 3.6) $\mathscr{A}^{\{0,1\}} \models\!\!\equiv u = v$.

For (2), let e be an equation of sort B, and let us first assume that $E \models\!\!\equiv e$. To show that $E_B \models e$, let us pick some algebra $\mathscr{A}$ such that $\mathscr{A} \models E_B$. By (1), it follows that $\mathscr{A}^{\{0,1\}}$ is a hidden model and $\mathscr{A}^{\{0,1\}} \models\!\!\equiv E$, which implies that $\mathscr{A}^{\{0,1\}} \models\!\!\equiv e$. Since e has a sort different from S, as explained in the proof of (1) it follows that $\mathscr{A} \models e$. Therefore, $E_B \models e$. Conversely, let $E_B \models e$ and let $\mathscr{A}$ be any hidden model such that $\mathscr{A} \models\!\!\equiv E$. By the definition of hidden models, $\mathscr{A}$ is also a algebra; moreover, $\mathscr{A}^{\{0,1\}} = \mathscr{A}$. By (1) it follows that $\mathscr{A} \models E_B$, which implies $\mathscr{A} \models e$. Since e has sort B, it follows that $\mathscr{A} \models\!\!\equiv e$. Hence $E \models\!\!\equiv e$.

Note that Definitions 3.6 and 3.8 imply that for any equation $s = t$ of sort S, $E \models\!\equiv s = t$ if and only if $E \models\!\equiv \mathsf{head}(\mathsf{tail}^n(s)) = \mathsf{head}(\mathsf{tail}^n(t))$ for all $n \in \mathbb{N}$. Then an immediate consequence of (2) above is that $E \models\!\equiv s = t$ if and only if $E_B \models \mathsf{head}(\mathsf{tail}^n(s)) = \mathsf{head}(\mathsf{tail}^n(t))$ for all $n \in \mathbb{N}$, where the latter entailment is that of $\textsc{fol}_=$.

We are now ready to show the membership in Π^0_2. We use the observation above and the fact that $\models$ in $\textsc{fol}_=$ admits complete deduction. Specifically, $E \models\!\equiv s = t$ if and only if for any $n \in \mathbb{N}$ *there is some proof* π_n in $\textsc{fol}_=$ of the entailment $E_B \models \mathsf{head}(\mathsf{tail}^n(s)) = \mathsf{head}(\mathsf{tail}^n(t))$. Rigorously speaking, we can encode all proofs in $\textsc{fol}_=$ as natural numbers m. Then we can more easily see that our stream equality problem is in Π^0_2 because it is a problem of the form $P(x) := \forall n.\, \exists m.\, R(x,n,m)$, where x ranges over pairs of equational bitstream specifications and equations of streams $(E,\, s = t)$, P is the predicate that $E \models\!\equiv s = t$, n ranges over the number of tail operations followed by a head applied to the streams s and t and m over $\textsc{fol}_=$ proofs, and R is a procedure checking that m is a correct proof for $E_B \models \mathsf{head}(\mathsf{tail}^n(s)) = \mathsf{head}(\mathsf{tail}^n(t))$ for all $n \in \mathbb{N}$. Since checking a given $\textsc{fol}_=$ proof is a decidable problem, the stream equality problem is in class Π^0_2.

We next show the Π^0_2-hardness by reduction from the totality problem (see Definition 2.12 and Proposition 2.13). Let us first give the totality problem a more convenient formulation. We claim that there are some Turing machines U, which according to our input encodings described in Section 2.1 take two natural numbers n and k as input, such that the following problem:

INPUT:	An integer $k \geq 0$
QUESTION:	Does U halt on all inputs 1^n01^k for all $n \in \mathbb{N}$?

which we refer to as TOTALITY, is Π^0_2-complete. It is obvious that TOTALITY is in Π^0_2 for any Turing machine U. To show that it is Π^0_2-hard, we may choose U to be a universal Turing machine such that on input 1^j01^k, U halts if and only if Turing machine with Gödel number k halts on input j, under some canonical assignment of Gödel numbers to Turing machines. Therefore, the conventional totality problem (Definition 2.12) has a positive solution for Turing machine M if and only if TOTALITY(k) has a positive solution for the Gödel index k of M. It follows then by Proposition 2.13 that TOTALITY is Π^0_2-complete. We henceforth fix some choice of U that makes the TOTALITY problem above Π^0_2-complete. Also, since in this case the particular output value (0 or 1) with which U terminates is irrelevant, we assume that it always terminates with output 1.

We next construct a hidden equational bitstream specification E corresponding to the Turing machine U chosen above and an equation e_k corresponding to any $k \in \mathbb{N}$ with the property that TOTALITY(k) has a positive solution if and only if $E \models\!\equiv e_k$. This will guarantee the desired Π^0_2-hardness result for behavioral satisfaction problem for hidden models of streams.

Let Σ be the bitstream signature containing, in addition to $0, 1, \mathsf{head}, \mathsf{tail} \in \Sigma$, the symbols $\mathsf{zeros}, \mathsf{ones} :: S$, ':' of type $B \times S \to S$, as well all the symbols $q :: S \times S \to B$ corresponding to the states q of U like in Definition 4.4. Here we do not need to add the various zip symbols discussed in Section 4. We now construct the bitstream specification E over Σ by adding the equations corresponding to the transition function of U (see Definition 4.4)

$$q(x, b : y) = q'(b' : x, y) \qquad \text{for every } \delta(q,b) = \langle q', b', R \rangle,$$
$$q(a : x, b : y) = q'(x, a : b' : y) \qquad \text{for every } \delta(q,b) = \langle q', b', L \rangle,$$

and, additionally, for halting configurations:

$$q(x, b : y) = b \qquad \text{whenever } \delta(q,b) \text{ is undefined.}$$

Since we assumed that U always terminates with output 1, the b in the last equation above is always 1. Since in hidden algebras for streams there is no predefined relationship between the interpretations of head, tail and ':', as well as no requirement stating that ':' is a constructor for streams or even that operations on streams preserve the behavioral equivalence generated by head/tail-observations, we need to add more equations to E.

We encode the fact that U's tape is infinite with the following equations

$$q(x, \mathsf{zeros}) = q(x, 0 : \mathsf{zeros})$$
$$q(\mathsf{zeros}, y) = q(0 : \mathsf{zeros}, y)$$

for all states q of U. Note that adding the equation $\mathsf{zeros} = 0 : \mathsf{zeros}$ instead of the above does not work, because the operations q need not preserve the behavioral equality. Computations in Turing machine U can now be mimicked with equational deduction with the equations above the same way like described in Section 4 (replacing applications of the equation $\mathsf{zeros} = 0 : \mathsf{zeros}$ by applications of one of the equations $q(x, \mathsf{zeros}) = q(x, 0 : \mathsf{zeros})$ or $q(\mathsf{zeros}, y) = q(0 : \mathsf{zeros}, y)$).

We also need the stream ones to yield as many 1 bits as requested with head/tail-observations, which can be easily achieved by adding the following:

$$\mathsf{head}(\mathsf{ones}) = 1$$
$$\mathsf{tail}(\mathsf{ones}) = \mathsf{ones}$$

Note that we do not need equations of the form $\mathsf{head}(b : x) = b$ and $\mathsf{tail}(b : x) = x$, because we do not assume that streams in our hidden models are necessarily constructed with :. Finally, to express the question in the TOTALITY problem, we add one more symbol to Σ, $\mathsf{is}_{\mathsf{total}} :: S \to S$, and two more equations to E:

$$\mathsf{head}(\mathsf{is}_{\mathsf{total}}(x)) = q_s(\mathsf{zeros}, x)$$
$$\mathsf{tail}(\mathsf{is}_{\mathsf{total}}(x)) = \mathsf{is}_{\mathsf{total}}(1 : x)$$

where q_s is the operation symbol corresponding to the starting state of the Turing machine U. Note that the head and tail equations above do not interfere with the other equations of E, so a result similar to Proposition 4.5 also holds in our context here, that is, U halts on input w if and only if $E \models q_0(\mathsf{zeros}, w_1 : w_2 : \ldots : w_{|w|} : \mathsf{zeros}) = 1$. Moreover, by the property (2) proved at the beginning of the proof of this theorem, the above holds if and only if $E_B \models q_0(\mathsf{zeros}, w_1 : w_2 : \ldots : w_{|w|} : \mathsf{zeros}) = 1$ (because all the equations in E relevant for such derivations are also in E_B).

Given $k \in \mathbb{N}$, let e_k be the stream equation

$$\mathsf{is}_{\mathsf{total}}(0 : (1 :)^k : \mathsf{zeros}) = \mathsf{ones}$$

By the property (2) at the beginning of the proof of this theorem, $E \models\equiv e_k$ if and only if

$$E_B \models \mathsf{head}(\mathsf{tail}^n(\mathsf{is}_{\mathsf{total}}(0:(1:)^k:\mathsf{zeros}))) = \mathsf{head}(\mathsf{tail}^n(\mathsf{ones}))$$

for all $n \in \mathbb{N}$, if and only if

$$E_B \models q_0((1:)^n:0:(1:)^k:\mathsf{zeros}) = 1$$

which holds if and only if U halts on the input 1^n01^k. Therefore, TOTALITY(k) has a positive solution if and only if $E \models\equiv e_k$, which completes the proof. □

6 Equality in Behavioral Models and Extensional Models

In this section we study two extensions of the hidden models semantics:

(A) We consider behavioral algebras that extend hidden algebras with the requirement that behavioral equivalence $\equiv$ being a congruence. As we have shown in Section 3, behavioral algebras are equivalent to extensional algebras.

(B) We consider full models which require that the domain of the algebras contains all streams over $\{0,1\}$.

We consider these semantics with respect to the following problems:

(i) equality in all models,
(ii) the existence of at most one solution over all models,
(iii) the existence of at least one solution over all models,
(iv) the existence of precisely one solution over all models.

We show that the extension (A) lifts the complexity of deciding equality in all models to the level Π^1_1 of the analytical hierarchy, and extension (B) yields a complexity that subsumes the entire analytical hierarchy.

6.1 Auxiliary Bitstream Specifications

We give some (systems of) equations that are used repeatedly throughout this section.

$$\left.\begin{aligned} \mathsf{is}_{\mathsf{zeros}}(\mathsf{zeros}) &= \mathsf{ones} \\ \mathsf{is}_{\mathsf{zeros}}(0:x) &= \mathsf{is}_{\mathsf{zeros}}(x) \\ \mathsf{is}_{\mathsf{zeros}}(1:x) &= \mathsf{zeros} \end{aligned}\right\} \tag{18}$$

This function does what its name suggests; it checks whether the argument is the stream of zeros. We use the stream of zeros for *false*, and the stream of ones for *true*.

Note that the equations are on terms, but when we speak of the *functions* they define, we refer to the set of possible interpretations in models of specifications that contain these equations. So for this example, in every model, the interpretation $[\![\mathsf{is}_{\mathsf{zeros}}]\!]$ is forced to satisfy $[\![\mathsf{is}_{\mathsf{zeros}}]\!](0^\omega) = 1^\omega$ (interpretation of the first equation), and $[\![\mathsf{is}_{\mathsf{zeros}}]\!](w) = 0^\omega$ for every $w \in A_S \setminus \{0^\omega\}$ (due to the second and third equation).

Streams of natural numbers are encoded as bitstreams, by a unary representation of numbers separated by zeros, as follows:

$$\mathrm{nat2bit}(nx) = 1^n\,0\,\mathrm{nat2bit}(x)$$

for all $n \in \mathbb{N}$ and streams $x \in \mathbb{N}^{\mathbb{N}}$, e.g., the stream $3\,1\,0\,2\ldots$ is encoded as $1110100110\ldots$.

Next we define functions $[\![\mathsf{uhd}]\!]$ and $[\![\mathsf{utl}]\!]$ that are the unary counterpart for $[\![\mathsf{head}]\!]$ and $[\![\mathsf{tail}]\!]$ on streams of natural numbers, as follows:

$$\left.\begin{aligned} \mathsf{uhd}(0:x) &= \mathsf{zeros} \\ \mathsf{uhd}(1:x) &= 1:\mathsf{uhd}(x) \\ \mathsf{utl}(0:x) &= x \\ \mathsf{utl}(1:x) &= \mathsf{utl}(x) \end{aligned}\right\} \tag{19}$$

For instance, we have

$$\begin{aligned} \mathsf{uhd}(1:1:1:0:1:0:0:1:1:\ldots) &= 1:1:1:\mathsf{zeros}\,, \\ \mathsf{utl}(1:1:1:0:1:0:0:1:1:\ldots) &= 1:0:0:1:1:\ldots \end{aligned}$$

Lemma 6.1. *In every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ of a specification including the equations from* (18) *and* (19) *we have:*

(i) $[\![\mathsf{is}_{\mathsf{zeros}}]\!](0^\omega) = 1^\omega$,
$[\![\mathsf{is}_{\mathsf{zeros}}]\!](w) = 0^\omega$ *for every* $w \in A_S \setminus \{0^\omega\}$,
(ii) $[\![\mathsf{uhd}]\!](1^n\,0\,w) = 1^n\,0^\omega$ *for every* $w \in A_S$,
$[\![\mathsf{uhd}]\!](1^\omega) = 1^\omega$,
(iii) $[\![\mathsf{utl}]\!](1^n\,0\,w) = w$ *for every* $w \in A_S$.

Remark 6.2. The restriction to extensional models is crucial in Lemma 6.1. For hidden models in general, it seems impossible to devise equations for $\mathsf{is}_{\mathsf{zeros}}$ that guarantee that in the model

(i) $[\![\mathsf{is}_{\mathsf{zeros}}]\!](\sigma) \equiv [\![\mathsf{ones}]\!]$ whenever $\sigma \equiv [\![\mathsf{zeros}]\!]$, and
(ii) $[\![\mathsf{is}_{\mathsf{zeros}}]\!](\sigma) \equiv [\![\mathsf{zeros}]\!]$ whenever $\sigma \not\equiv [\![\mathsf{zeros}]\!]$.

For example, note that the Equations (18) do not ensure this property in hidden models. The equation $\mathsf{is}_{\mathsf{zeros}}(\mathsf{zeros}) = \mathsf{ones}$ only guarantees that $[\![\mathsf{is}_{\mathsf{zeros}}]\!]([\![\mathsf{zeros}]\!]) \equiv [\![\mathsf{ones}]\!]$, but it does not ensure that $[\![\mathsf{is}_{\mathsf{zeros}}]\!](\sigma) \equiv [\![\mathsf{ones}]\!]$ for elements $\sigma \in A_S$ that are behaviorally equivalent with but not identical to $[\![\mathsf{zeros}]\!]$, $\sigma \equiv [\![\mathsf{zeros}]\!]$ and $\sigma \neq [\![\mathsf{zeros}]\!]$.

However, we emphasize that $\mathsf{is}_{\mathsf{zeros}}$ is not the reason that the main results of this section do not generalize to hidden models. In particular, $\mathsf{is}_{\mathsf{zeros}}$ is not essential for the proof of Theorem 6.6; the result can be obtained without $\mathsf{is}_{\mathsf{zeros}}$ as in the proof of Theorem 6.9. We crucially employ $\mathsf{is}_{\mathsf{zeros}}$ only for results on the uniqueness of solutions.

The crucial point why the results do not generalize to hidden models is the definition of natstr (below). In hidden models, there seems to be no possibility (for similar reasons as above) to enforce that the interpretation $[\![\mathsf{X}]\!]$ of a stream constant X contains always eventually a 0, and thus can be interpreted as a stream of natural numbers.

Taking a closer look at Lemma 6.1, we note that all interpretations are uniquely defined, apart from $[\![\mathsf{utl}]\!](1^\omega)$ which can be any stream depending on the model. One way to avoid this problem is to ensure that a certain bitstream is a valid encoding of a stream of natural numbers, where we call a bitstream a valid encoding when it contains infinitely many zeros:

$$\left.\begin{aligned}\mathsf{natstr}(\mathsf{ones}) &= \mathsf{zeros}\\ \mathsf{natstr}(0:x) &= 1:\mathsf{natstr}(x)\\ \mathsf{natstr}(1:x) &= \mathsf{natstr}(x)\end{aligned}\right\}\tag{20}$$

Then an equation $\mathsf{natstr}(\mathsf{X}) = \mathsf{ones}$ guarantees that in every model, the interpretation $[\![\mathsf{X}]\!]$ contains an infinite number of zeros, and thus represents a stream of natural numbers:

Lemma 6.3. *In every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ of a specification including the equations from* (20) *we have: $[\![\mathsf{natstr}]\!](w) = 1^z$ where $z \in \mathbb{N} \cup \{\omega\}$ is the number of zeros in w.*

Proof. The second and third equation 'walk' over the stream, deleting 1's and converting 0's to 1's. If the stream contains infinitely many 0's, then an infinite stream of 1's will be produced. However, if some tail of the stream contains only 1's then the top equation ensures that the interpretation is unequal to 1^ω. ☐

Definition 6.4. Let $T = \langle Q, q_0, \delta\rangle$ be a Turing machine, and fix $m \in \mathbb{N}$. The *canonical model* $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ for the union of the specifications E_T, Z_m (Definition 4.2), and (18), (19) and (20) introduced above, is defined to consist of the domain $A_S = \{0,1\}^\omega$ with interpretations $[\![\cdot]\!]$ as given in Lemmas 4.3, 6.1, and 6.3, extended by

(i) $[\![\mathsf{utl}]\!](1^\omega) = 1^\omega$,
(ii) $[\![q]\!](\vec{x},\vec{y}) = 1$ whenever $q(\ulcorner\vec{x}\urcorner,\ulcorner\vec{y}\urcorner) \to^* 1$, and $[\![q]\!](\vec{x},\vec{y}) = 0$ otherwise,
for all streams $x_1,\dots,x_m$ and $y_1,\dots,y_k$, where for a stream x we denote by $\ulcorner x\urcorner$ the infinite term such that $[\![\ulcorner x\urcorner]\!] = x$.

In item (ii) of the above definition the intention is that the arguments $\vec{x}$ represent oracles (sets of integers), and that the arguments $\vec{y}$ represent input integers. The reason for doing it like this is that the rewrite relation $\to$ is defined on terms, while at the same time $[\![q]\!]$ has to be defined on all streams.

Lemma 6.5. *The canonical model is a model of the union of the equational specifications E_T, Z_m,* (18)*,* (19) *and* (20)*.*

Proof. The rewrite system R_T is orthogonal, consequently we have infinitary unique normal forms, see (Terese, 2003). Hence, we can employ a normal forms semantics for $[\![q]\!]$ (where we map terms without normal forms to 0); this justifies Definition 6.4 (ii). For the remaining equations for zeros, ones, zip_n, $\mathsf{is}_{\mathsf{zeros}}$, uhd, utl and natstr it is straightforward to check that the interpretation forms a model of the equations, that is, that the interpretation of the left-hand sides coincides with the interpretation of the right-hand sides for every valuation of the variables. ☐

6.2 Equality in All Extensional Models

In this section, we consider the complexity of deciding the following problems for specifications of bitstreams:

(i) equality in all extensional models,

(ii) the existence of at most one solution over all extensional models,
(iii) the existence of at least one solution over all extensional models,
(iv) the existence of precisely one solution over all extensional models.

Deciding equality in all models (i) turns out to be Π^1_1-complete. Let us sketch the ideas for both membership and hardness.

– The membership in the class Π^1_1 follows from the Downward Löwenheim–Skolem theorem. The idea is that if there exists a model that fulfills the specification, but invalidates the goal equality, then there exists a countable model that does so. This countable model can basically be constructed by taking the values of the variables that invalidate the goal equation, and then close this set under all function interpretations in the model (thus in particular the interpretations of stream constants). The quantification over countable models can be done by using a Π^1_1-formula.
– To show Π^1_1-hardness we reduce the well-foundedness problem (Proposition 2.15) to an equality problem. The idea is to employ an underspecified stream constant X to 'guess' a non-well-founded path. Here the term 'guessing' refers to the fact that the interpretation of X is determined by the model. Then the question of the existence of a non-well-founded path translates to the question whether there exists a model that invalidates the goal equation. The only requirement on X is $\mathsf{natstr}(\mathsf{X}) = \mathsf{ones}$ to ensure that $[\![\mathsf{X}]\!]$ contains an infinite number of zeros, and thus encodes a stream of natural numbers $n_1, n_2, n_3 \ldots$ in the form $1^{n_1}\,0\,1^{n_2}\,0\,1^{n_3}\,0\ldots$. Then by recursion on X we can define a stream term $\mathsf{run} = T(n_1,n_2) : T(n_2,n_3) : T(n_3,n_4) : \ldots$ where $T(n,m)$ is 0 if and only if the pair $\langle n,m\rangle$ is in the relation computed by T. Then the stream $\mathsf{is}_{\mathsf{zeros}}(\mathsf{run})$ is the stream 1^ω if the path X is not well-founded, and 0^ω otherwise. Thus $\mathsf{is}_{\mathsf{zeros}}(\mathsf{run})$ is equal to the stream 0^ω in all models if and only if every path is well-founded.

As an immediate consequence we obtain that the problem (ii) of deciding whether there is at most one solution is a Π^1_1-complete problem as well. The term $\mathsf{is}_{\mathsf{zeros}}(\mathsf{run})$ has the single solution 0^ω if and only if every path X is well-founded. If there is a non-well-founded path X, then the term $\mathsf{is}_{\mathsf{zeros}}(\mathsf{run})$ allows for the solution 1^ω as well.

For Σ^1_1-completeness of problem (iii), the existence of at least one solution, we extend the equational system with $\mathsf{is}_{\mathsf{zeros}}(\mathsf{run}) = \mathsf{ones}$. Then this system has a model if and only if there exists a non-well-founded path. This is the complement of the well-foundedness problem, and the complement of the class Π^1_1 is Σ^1_1.

For problem (iv), having precisely one solution, note that the specifications for problems (ii) and (iii) have precisely one solution if they have at most one solution, and at least one solution, respectively. Hence, the problem (iv) is both Π^1_1-hard and Σ^1_1-hard. On the other hand, having precisely one solution is the conjunction of having at least one, and at most one solution. Thus the problem can be described by the conjunction of a Π^1_1-formula and a Σ^1_1-formula, and hence it is strictly contained in the class Δ^1_2. Beyond these observations, the precise complexity of problem (iv) remains open.

For the complexity of equality in all extensional models we obtain:

Theorem 6.6. *The following problem is Π^1_1-complete:*

INPUT: *Bitstream specification E, terms $s,t :: S$.*

QUESTION: *Are s and t equal in all extensional models of E?*

Proof. We reduce the well-foundedness problem (see Proposition 2.15) to an equality problem. Let $M \subseteq \mathbb{N} \times \mathbb{N}$ be a decidable predicate, and $T = \langle Q, q_0, \delta \rangle$ be the corresponding Turing machine. We define the following specification E:

$$\begin{aligned}
\mathsf{S} &= \mathsf{is}_{\mathsf{zeros}}(\mathsf{run}(1,\mathsf{X})) \\
\mathsf{natstr}(\mathsf{X}) &= \mathsf{ones} \\
\mathsf{run}(0,x) &= \mathsf{ones} \\
\mathsf{run}(1,x) &= 0 : \mathsf{run}(\underbrace{T(\mathsf{zeros};\mathsf{uhd}(x),\mathsf{uhd}(\mathsf{utl}(x)))}_{\Phi(x)},\mathsf{utl}(x))
\end{aligned}$$

together with the equations from E_T and Definition 4.2, (18), (19) and (20). We prove that: $E \models_{\mathrm{ext}} \mathsf{S} = \mathsf{zeros}$ if and only if M is well-founded.

For the direction '$\Rightarrow$' let M be non-well-founded, and let $n_0\ M\ n_1\ M\ n_2\ M \cdots$ be an infinite chain. We construct an algebra $\mathscr{A} = \langle A, [\![\cdot]\!] \rangle$ such that $\mathscr{A} \models E$ but not $\mathscr{A} \models \mathsf{S} = \mathsf{zeros}$. We define $\mathscr{A}$ as an extension of the canonical model (Definition 6.4). The values of $[\![\Phi(x)]\!]$ and $[\![\mathsf{utl}(x)]\!]$ are determined by the canonical model, and together with the above equations for the symbol run we obtain, for every stream $\xi \in \{0,1\}^\omega$, that $[\![\mathsf{run}]\!](0,\xi) = 1^\omega$, and $[\![\mathsf{run}]\!](1,\xi) = 0 : [\![\mathsf{run}]\!]([\![\Phi(\underline{\xi})]\!], [\![\mathsf{utl}]\!](\xi))$. Hence, there is a unique interpretation $[\![\mathsf{run}]\!]$ that results in a model for the equations of run. We define $\kappa_i = 1^{n_i}\,0\,1^{n_{i+1}}\,0\,1^{n_{i+2}}\ldots$. and we let $\underline{n} = 1^n\,0^\omega$. Then for $i \in \mathbb{N}$ we have

$$\begin{aligned}
[\![\mathsf{run}]\!](1,\kappa_i) &= 0 : [\![\mathsf{run}]\!]([\![\Phi]\!](\kappa_i), \kappa_{i+1}) \\
&= 0 : [\![\mathsf{run}]\!]([\![T]\!](\underline{n_i}, \underline{n_{i+1}}), \kappa_{i+1}) \\
&= 0 : [\![\mathsf{run}]\!](1, \kappa_{i+1})
\end{aligned}$$

since we have that $[\![\mathsf{uhd}]\!](\kappa_j) = \underline{n_j}$ and $[\![\mathsf{utl}]\!](\kappa_j) = \kappa_{j+1}$ for all $j \in \mathbb{N}$ by Lemma 6.1. Thus, $[\![\mathsf{run}]\!](1,\kappa_0) = 0^\omega$. Let $[\![\mathsf{X}]\!] = \kappa_0$ and $[\![\mathsf{S}]\!] = 1^\omega$. Then $[\![\mathsf{natstr}]\!]([\![\mathsf{X}]\!]) = [\![\mathsf{ones}]\!]$ by Lemma 6.3, and $[\![\mathsf{S}]\!] = [\![\mathsf{is}_{\mathsf{zeros}}]\!]([\![\mathsf{run}]\!](1,[\![\mathsf{X}]\!]))$ by Lemma 6.1. We have constructed a model, where $[\![\mathsf{S}]\!] = 1^\omega$, and, hence, $E \not\models_{\mathrm{ext}} \mathsf{S} = \mathsf{zeros}$.

For the direction '$\Leftarrow$' let M be well-founded. Let $\mathscr{A}$ be an algebra such that $\mathscr{A} \models E$. We show that $[\![\mathsf{S}]\!] = 0^\omega$. Since $[\![\mathsf{natstr}]\!]([\![\mathsf{X}]\!]) = [\![\mathsf{ones}]\!]$, $[\![\mathsf{X}]\!]$ contains infinitely many zeros by Lemma 6.3. Thus, $[\![\mathsf{X}]\!] = 1^{n_0}\,0\,1^{n_1}\,0\,1^{n_2}\ldots$ for some $n_0, n_1, n_2, \ldots \in \mathbb{N}$. Let $\kappa_i = 1^{n_i}\,0\,1^{n_{i+1}}\,0\,1^{n_{i+2}}\ldots$ for $i \in \mathbb{N}$. Then

$$\begin{aligned}
[\![\mathsf{run}]\!](1,\kappa_i) &= 0 : [\![\mathsf{run}]\!]([\![T]\!](\underline{n_i}, \underline{n_{i+1}}), \kappa_{i+1}) \\
&= \begin{cases} [\![\mathsf{run}]\!](1,\kappa_{i+1}) & \text{if } [\![T]\!](\underline{n_i}, \underline{n_{i+1}}) = 1\,, \\ [\![\mathsf{run}]\!](0,\kappa_{i+1}) = 1^\omega & \text{if } [\![T]\!](\underline{n_i}, \underline{n_{i+1}}) = 0\,. \end{cases}
\end{aligned}$$

Hence, $[\![\mathsf{run}]\!](1,[\![\mathsf{X}]\!]) = 0^\omega$ if and only if $[\![T]\!](\underline{n_i}, \underline{n_{i+1}}) = 1$ for all $i \in \mathbb{N}$. But this would contradict well-foundedness of M. As a consequence, we obtain that $[\![\mathsf{run}]\!](1,[\![\mathsf{X}]\!]) \neq 0^\omega$ and $[\![\mathsf{S}]\!] = \mathsf{is}_{\mathsf{zeros}}([\![\mathsf{run}]\!](1,[\![\mathsf{X}]\!])) = 0^\omega$ by Lemma 6.1. This concludes the Π^1_1-hardness proof.

For the Π^1_1-membership we show that in "$\forall \mathscr{A}.\, \mathscr{A} \models\equiv E \Rightarrow \mathscr{A} \models\equiv s = t$" we only need to quantify over countable models, which corresponds to a single $\forall \mathscr{A} \subseteq \mathbb{N}$ set quantifier. For this purpose it suffices to prove that if there exists an uncountable extensional model $\mathscr{A}$ such that $(*)$ $\mathscr{A} \models\equiv E$ but $\mathscr{A} \not\models\equiv s = t$, then there exists a countable model $\mathscr{A}'$ with the same property. Let $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ be an uncountable model with the property $(*)$. Let α be an interpretation of the variables such that $[\![s,\alpha]\!] \neq [\![t,\alpha]\!]$. Let $\mathscr{V}$ be the finite set of variables occurring in s and t. Let A' be the smallest set that contains $\{0,1\}$ and $\{[\![v]\!] \mid v \in \mathscr{V}\}$ and is closed under $[\![\mathsf{f}]\!]$ for all $\mathsf{f} \in \Sigma$. Then obviously A' is countable, and we define $\mathscr{A}'$ to be the restriction of $\mathscr{A}$ to the domain A'. By construction we have $\mathscr{A}' \models\equiv E$ and $\mathscr{A}' \not\models\equiv s = t$. □

Due to the equivalence of extensional and behavioral models we obtain:

Theorem 6.7. *The following problem is Π^1_1-complete:*

INPUT: *Bitstream specification E, terms $s,t :: S$.*

QUESTION: *Are s and t equal in all behavioral models of E?*

Proof. Follows directly from Theorem 6.6 and Proposition 3.15. □

The following three results are obtained by slight adaptations of the proof of Theorem 6.6. In the proof of Theorem 6.6, we have $E \models\equiv_{\text{ext}} \mathsf{S} = \mathsf{zeros}$ if and only if S has a unique solution over all extensional models of E. As a consequence, we obtain the following results concerning (unique) solvability:

Theorem 6.8. *The following problem is Π^1_1-complete:*

INPUT: *Bitstream specification E, term s.*

QUESTION: *Does s have <u>at most one</u> solution over all extensional models of E?*

Proof. The Π^1_1-hardness follows from the proof of Theorem 6.6, as S has ≤ 1 solutions if and only if M is well-founded.

The proof of membership in Π^1_1 uses that it suffices to consider countable models as in the proof of Theorem 6.6. Then the formula characterizes the property having at most one solution: $\forall \mathscr{A}_1.\, \forall \mathscr{A}_2.\, (\mathscr{A}_1 \models\equiv E) \wedge (\mathscr{A}_2 \models\equiv E) \Rightarrow [\![s]\!]^{\mathscr{A}_1} = [\![s]\!]^{\mathscr{A}_2}$. The two $\forall$ set quantifiers can be merged into one, and the properties $\mathscr{A} \models\equiv E$, and $[\![s]\!]^{\mathscr{A}_1} = [\![s]\!]^{\mathscr{A}_2}$ are arithmetic. Hence, the property is in Π^1_1. □

Theorem 6.9. *The following problem is Σ^1_1-complete:*

INPUT: *A bitstream specification E, a term s.*

QUESTION: *Does s have <u>at least one</u> solution over all extensional models of E?*

Proof. The Σ^1_1-hardness follows from a tiny adaptation of the proof of Theorem 6.6. We replace the equation $\mathsf{S} = \mathsf{is}_{\mathsf{zeros}}(\mathsf{run}(1,\mathsf{X}))$ by the equations $\mathsf{S} = \mathsf{run}(1,\mathsf{X})$ and $\mathsf{S} = \mathsf{zeros}$. Then every model where $[\![\mathsf{run}(1,\mathsf{X})]\!] \neq 0^\omega$ is ruled out, and hence, the specification has a model, and S a solution, if and only if M is not well-founded.

The membership in Σ^1_1 can be described by the following formula (we again use that we only need to quantify over countable algebras): $\exists \mathscr{A}.\, \mathscr{A} \models\equiv E$. Hence, the property is in Σ^1_1. □

Theorem 6.10. *The following problem is Π^1_1-hard, Σ^1_1-hard and strictly contained in Δ^1_2:*

INPUT: *A bitstream specification E, a term s.*

QUESTION: *Does s have $\underline{\text{exactly one}}$ solution over all extensional models of E?*

Proof. The Π^1_1-hardness follows from the fact that the specification used in the proof of Theorem 6.8 always has a solution; then unique solvability coincides with at most one solution.

The Σ^1_1-hardness is a consequence of the fact that the specification used in the proof of Theorem 6.9 always has at most one solution (due to the equation $\mathsf{S} = \mathsf{zeros}$); then unique solvability coincides with at least one solution.

For the Δ^1_2-membership we observe that a term s has a unique solution if and only if s has at least and s has at most one solution. Therefore unique solvability can be described by the conjunction of a Π^1_1- and a Σ^1_1-formula. □

6.3 Equality in Full Extensional Models

In Section 6.2 we have considered models whose domain was allowed to be any non-empty set of bitstreams ($A_S \subseteq \{0,1\}^\omega$). However, when writing equations such as

$$\mathsf{even}(a : b : x) = a : \mathsf{even}(x) ,$$

the intended semantics is often that these equations should hold for all streams, that is, in full models with domain $A_S = \{0,1\}^\omega$. We find that the restriction to full models results in a huge jump of the complexity, which then subsumes the entire analytical hierarchy.

In this section, we consider the complexity of deciding the following problems for specifications of bitstreams:

(i) equality in all $\underline{\text{full}}$ extensional models,
(ii) the existence of at most one solution over all $\underline{\text{full}}$ extensional models,
(iii) the existence of at least one solution over all $\underline{\text{full}}$ extensional models,
(iv) the existence of precisely one solution over all $\underline{\text{full}}$ extensional models.

It turns out that the complexity of each of these problems subsumes the arithmetical and analytical hierarchies. This means that their precise complexity resides above the analytical hierarchy, that is, in a hierarchy of third-order arithmetic or higher. We leave the determination of their precise complexity to future work.

Definition 6.11. A problem P is said to *subsume the arithmetical and analytical hierarchy* if every problem that is a member of these hierarchies can be reduced to P.

The idea of the proof is as follows. Every formula of the analytical hierarchy can be written in the form

$$\forall \xi_1 . \exists \xi_2 . \forall \xi_3 . \ldots \exists \xi_{2n} . \ \forall x_1 . \exists x_2 . \ M(\xi_1, \xi_2, \ldots, \xi_{2n}, a, x_1, x_2)$$

where $n \in \mathbb{N}$ and M is a decidable predicate. First we replace the existential set quantifiers $\exists \xi_2, \exists \xi_4, \ldots, \exists \xi_{2n}$ by existential quantification over Skolem functions mapping sets to sets:

$g_2, g_4, \dots, g_{2n} : \wp(\mathbb{N}) \to \wp(\mathbb{N})$. We obtain a formula of the form

$$\begin{array}{l}\exists g_2.\, \exists g_4.\, \dots \exists g_{2n}.\\ \forall \xi_1.\, \forall \xi_3.\, \dots \forall \xi_{2n-1}.\ \forall x_1.\, \exists x_2.\ M(\xi_1, g_2(\xi_1), \xi_3, \dots, g_{2n}(\xi_1, \xi_3, \dots, \xi_{2n-1}), a, x_1, x_2)\end{array}$$

When translating this formula into an equality problem, we use stream functions to model $g_2, g_4, \dots, g_n$. We leave these functions unspecified so that the model of the specification can freely determine their interpretations $[\![g_2]\!], \dots, [\![g_{2n}]\!]$. Then the question of the existence of a model (invalidating the equality) translates into the existence of functions $g_2, g_4, \dots, g_{2n}$ satisfying the remainder of the formula.

Now the universal quantifiers $\forall \xi_1, \dots, \forall \xi_{2n-1}$ can be translated into an equation

$$\mathsf{S}(\xi_1, \dots, \xi_{2n-1}) = \dots$$

where $\xi_1, \dots, \xi_{2n-1}$ are stream variables. By definition of models, the equations have to be valid for every interpretation of the variables by the streams in the domain of the model. At this point it is essential that the models are *full*, and thus the equations have to hold for all assignments of the variables.

To prepare for the proof, we introduce some auxiliary specifications. We define nat such that an equation $\mathsf{nat}(\mathsf{X}) = \mathsf{ones}$ guarantees that the interpretation $[\![\mathsf{X}]\!]$ represents a natural number in unary encoding, that is, $[\![\mathsf{X}]\!] = 1^n\, 0^\omega$ for some $n \in \mathbb{N}$, as follows:

$$\left.\begin{array}{rll}\mathsf{nat}(0:1:x) &= \mathsf{zeros} & \text{(21a)}\\ \mathsf{nat}(1:x) &= \mathsf{nat}(x) & \text{(21b)}\\ \mathsf{nat}(0:0:x) &= \mathsf{nat}(0:x) & \text{(21c)}\\ \mathsf{nat}(\mathsf{ones}) &= \mathsf{zeros} & \text{(21d)}\end{array}\right\} \tag{21}$$

Note that $[\![\mathsf{nat}]\!](\mathsf{X}) = 1^\omega$ implies that $[\![\mathsf{X}]\!]$ encodes a natural number. The reverse implication does not hold. In particular, the above equations (21) allow for the interpretation $[\![\mathsf{nat}]\!](w) = 0^\omega$ for every $w \in A_S$. For our purposes it is sufficient that an equation $\mathsf{nat}(\mathsf{X}) = \mathsf{ones}$ in the specification guarantees $[\![\mathsf{nat}]\!](\mathsf{X}) = 1^\omega$, and so $[\![\mathsf{X}]\!]$ encodes a natural number.

Lemma 6.12. *In every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ of a specification including the equations from* (21) *the following implication holds: if* $[\![\mathsf{nat}]\!](w) = 1^\omega$*, then* $w = 1^n\, 0^\omega$ *for some* $n \in \mathbb{N}$.

Proof. If a stream is not of the form $1^n\, 0^\omega$ for some $n \in \mathbb{N}$ then it is 1^ω or contains $\dots 01 \dots$. The equation (21d) rules out the case 1^ω (ensures that the interpretation is not 1^ω).

The equations (21a), (21b) and (21c) are exhaustive in the sense that every stream can be matched by one of them. The equation (21a) rules out streams that contain a 1 after a 0, and the equations (21b) and (21c) 'walk' step by step over the stream (proceed with the tail). □

We moreover define a function leq such that $\mathsf{leq}(x,y) = \mathsf{ones}$ guarantees that the inequality $[\![x]\!] \le [\![y]\!]$ holds pointwise:

$$\left.\begin{array}{rl}\mathsf{leq}(0:x, a:y) &= \mathsf{leq}(x,y)\\ \mathsf{leq}(1:x, 1:y) &= \mathsf{leq}(x,y)\\ \mathsf{leq}(1:x, 0:y) &= \mathsf{zeros}\end{array}\right\} \tag{22}$$

We note that the interpretation $[\![\mathsf{leq}]\!](u,v) = 0^\omega$ for every $u,v \in A_S$ gives rise to a model of this specification. For our purposes, it suffices that an equation $\mathsf{leq}(\mathsf{X},\mathsf{Y}) = \mathsf{ones}$ in a specification implies $[\![\mathsf{leq}]\!]([\![\mathsf{X}]\!],[\![\mathsf{Y}]\!]) = 1^\omega$ and hence $[\![\mathsf{X}]\!] \leq [\![\mathsf{Y}]\!]$ in the model.

Lemma 6.13. *In every extensional model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ of a specification including the equations from* (22) *we have that if $[\![\mathsf{leq}]\!](x,y) = 1^\omega$, then x is pointwise $\leq$ than y, for all $x,y \in A_S$.*

Lemmas 6.12 and 6.13 are valid for non-full models as well. As explained before, the assumption of full models is crucial to guarantee that equations with variables have to hold for all streams (assigned to the variables) and not only the streams in the model.

Theorem 6.14. *The following problem subsumes the analytical hierarchy:*

INPUT: *Bitstream specification E, terms $s,t :: S$.*

QUESTION: *Are s and t equal in all* full *extensional models of E?*

Proof. For every analytical set A, we reduce the membership problem in A to an equality problem. Every set A of the analytical hierarchy can be defined by

$$a \notin A \iff \forall \xi_1.\, \exists \xi_2.\, \forall \xi_3.\, \ldots \exists \xi_n.\ \forall x_1.\, \exists x_2.\ M(\xi_1,\ldots,\xi_n,a,x_1,x_2) \tag{23}$$

where $n \in \mathbb{N}$ is even (without loss of generality since $\Pi^1_n \subset \Pi^1_{n+1}$) and M a decidable predicate. Let $T = \langle Q, q_0, \delta\rangle$ be the Turing machine corresponding to M. Let $a \in \mathbb{N}$ be given. We define E to be the following system of equations:

$$\begin{aligned}
\mathsf{S}(\tau_1,\tau_3,\ldots,\tau_{n-1}) &= \mathsf{run}(1,\, \mathsf{zip}_n(\tau_1,\mathsf{g}_2(\tau_1),\tau_3,\mathsf{g}_4(\tau_1,\tau_3),\\
&\qquad \ldots,\tau_{n-1},\mathsf{g_n}(\tau_1,\tau_3,\ldots,\tau_{n-1})),\, \mathsf{zeros})\\
\mathsf{S}(\tau_1,\tau_3,\ldots,\tau_{n-1}) &= \mathsf{zeros}\\
\mathsf{run}(0,\tau,\gamma_1) &= \mathsf{ones}\\
\mathsf{run}(1,\tau,\gamma_1) &= 0 : \mathsf{run}(T(\tau;A,\gamma_1,\mathsf{h}_2(\tau,\gamma_1)),\ \tau,\ 1:\gamma_1)\\
A &= (1\,:)^a\ \mathsf{zeros}\\
\mathsf{nat}(\mathsf{h}_2(\tau,\gamma_1)) &= \mathsf{ones}
\end{aligned}$$

together with the equations from E_T, Definition 4.2, and (21). The symbols g_{2i} are typed $S^i \to S$. We claim: $E \models_{\text{full}} \mathsf{zeros} = \mathsf{ones}$ if and only if $a \in A$. For this purpose it suffices to show that the specification has a model ($\exists \mathscr{A}.\, \mathscr{A} \models E$) if and only if the formula in the right-hand side of (23) is valid.

The specification models a Skolem normal form of the analytical formula in (23), namely

$$\begin{aligned}
&\exists g_2 : \wp(\mathbb{N}) \to \wp(\mathbb{N}),\ g_4 : \wp(\mathbb{N})^2 \to \wp(\mathbb{N}),\ \ldots,\ g_n : \wp(\mathbb{N})^{\frac{n}{2}} \to \wp(\mathbb{N}).\\
&\quad \exists h_2 : \wp(\mathbb{N})^{\frac{n}{2}} \times \mathbb{N} \to \mathbb{N}.\\
&\qquad \forall \xi_1.\, \forall \xi_3.\, \ldots \forall \xi_{n-1}.\ \forall x_1.\\
&\qquad\quad M(\xi_1, g_2(\xi_1), \xi_3, g_4(\xi_1,\xi_3),\ldots, g_4(\xi_1,\xi_3,\ldots,\xi_{n-1}),\ a,\ x_1, h_2(\xi_1,\ldots,\xi_{n-1},x_1))
\end{aligned}$$

The $\exists$ set quantifiers are modeled by Skolem functions $g_2, g_4, \ldots, g_n$ which in the specification are stream functions that get the value of the preceding $\forall$ quantifiers as arguments, and the number quantifier $\exists x_2$ is modeled by the Skolem function h_2.

In the equational specification, the $\forall$ set quantifiers are modeled by an equation with stream variables; recall that equations have to hold for all assignments of the variables. In particular, the variables $\tau_1, \tau_3, \ldots, \tau_{n-1}$ in the first equation $\mathsf{S}(\tau_1, \tau_3, \ldots, \tau_{n-1}) = \ldots$ model the set quantifiers $\forall\xi_1, \ldots, \forall\xi_{n-1}$, respectively. The specification contains stream functions g_{2i} which are unspecified and therefore can be 'freely chosen' by the model $\mathscr{A}$. Thus, the existential quantification over the Skolem functions corresponds to the existential quantification over all models in $\exists\mathscr{A}.\,\mathscr{A} \models E$.

The streams $\tau_1, g_2(\tau_i), \ldots, \tau_{n-1}, g_n(\tau_1, \tau_3, \ldots, \tau_{n-1})$ that represent the values of the set quantifiers are then interleaved by zip_n, and passed as the second argument, named τ, to run; this argument serves as the left side of the tape for every invocation of the Turing machine T.

The $\forall x_1$ number quantifier is modeled by the third argument γ_1 of run. The initial value of γ_1 is zeros, and '$1 : \Box$' is prepended (corresponding to counting up) each time the Turing machine halts with output 1. The number quantifier $\exists x_2$ is modeled by the Skolem function h_2 for which the equation $\mathsf{nat}(\mathsf{h}_2(\tau, \gamma_1)) = \mathsf{ones}$ ensures by Lemma 6.1 that the interpretation $[\![\mathsf{h}_2(\tau, \gamma_1)]\!]$ is a unary encoding of a natural number. Then the term $T(\tau; A, \gamma_1, \mathsf{h}_2(\tau, \gamma_1))$ with $\tau = \mathsf{zip}_n(\tau_1, \mathsf{g}_2(\tau_1), \tau_3, \mathsf{g}_4(\tau_1, \tau_3), \ldots, \tau_{n-1}, \mathsf{g}_n(\tau_1, \tau_3, \ldots, \tau_{n-1}))$ corresponds precisely to $M(\xi_1, \ldots, \xi_n, a, x_1, x_2)$ in (23).

For '$\Leftarrow$', assume that the formula in (23) is valid. We construct a model $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ as an extension of the canonical model (Definition 6.4). For $[\![g_2]\!], [\![g_4]\!], \ldots, [\![g_n]\!], [\![h_2]\!]$ we pick the Skolem functions for the quantifiers $\exists\xi_2, \exists\xi_4, \ldots, \exists\xi_n, \exists x_2$, respectively (where $[\![h_2]\!]$ is a stream function that works on the unary encoding of natural numbers). For $\sigma \in \{0,1\}^\omega$, we define $[\![\mathsf{nat}]\!](\sigma) = 1^\omega$ if σ is of the form $1^n 0^\omega$, and 0^ω, otherwise. The definition of $[\![\mathsf{run}]\!]$ is analogous to the proof of Theorem 6.6. Finally, we define $[\![\mathsf{S}]\!](\tau_1, \tau_2, \ldots, \tau_{n-1}) = 0^\omega$ for all $\tau_1, \tau_2, \ldots, \tau_{n-1} \in \{0,1\}^\omega$, and $[\![A]\!] = 1^a 0^\omega$. Then it is straightforward to verify that $\mathscr{A}$ is a model of the specification.

For '$\Rightarrow$', let $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ be a model of the specification. Then we let the existential quantifiers $\exists\xi_2, \exists\xi_4, \ldots, \exists\xi_n$ and $\exists x_2$ in (23) behave according to the interpretations $[\![g_2]\!], [\![g_4]\!], \ldots, [\![g_n]\!], [\![h_2]\!]$, respectively (here the translation from sets $\xi \subseteq \mathbb{N}$ to streams $\underline{\xi}$ is as usual). Assume that there exists an assignment of the $\forall$ quantifiers $\forall\xi_1, \forall\xi_2, \ldots, \forall\xi_{n-1}$ and $\forall x_2$ for which the formula in (23) is not valid, that is, $M(\xi_1, \ldots, \xi_n, a, x_1, x_2)$ does not hold where the existential choices are governed by the model as described above. We translate this 'counterexample' back to the model by considering $[\![\mathsf{S}]\!](\underline{\xi_1}, \underline{\xi_3} \ldots, \underline{\xi_{n-1}})$. As in the proof of Theorem 6.6, it is then straightforward to show that $[\![\mathsf{S}]\!](\underline{\xi_1}, \underline{\xi_3} \ldots, \underline{\xi_{n-1}}) \neq 0^\omega$. However, this contradicts the assumption of $\mathscr{A}$ being a model due to the equation $\mathsf{S}(\tau_1, \tau_3, \ldots, \tau_{n-1}) = \mathsf{zeros}$. $\Box$

For the proof of the following theorem, we slightly adapt the specification in the proof of Theorem 6.14 such that it always has a solution, and has more than one solution if and only if the analytical formula in (23) holds.

Theorem 6.15. *The following problem subsumes the analytical hierarchy:*

INPUT: *Bitstream specification E, term s.*

QUESTION: *Does s have <u>at most one</u> solution over all* full *extensional models of E?*

Proof. We take the specification from the proof of Theorem 6.14, but replace the two equations $\mathsf{S}(\dots) = \dots$ by the following equation:

$$\mathsf{ones} = \mathsf{leq}(\mathsf{S}, \mathsf{is}_{\mathsf{zeros}}(\mathsf{run}(1, Z, \mathsf{zeros})))$$

where Z abbreviates the term

$$\mathsf{zip}_n(\tau_1, \mathsf{g}_2(\tau_1), \tau_3, \mathsf{g}_4(\tau_1, \tau_3), \dots, \tau_{n-1}, \mathsf{g}_\mathsf{n}(\tau_1, \tau_3, \dots, \tau_{n-1}))$$

In this way an interpretation $[\![\mathsf{S}]\!] = 0^\omega$ always yields a solution. In addition, by Lemma 6.13 we have that $[\![\mathsf{S}]\!] \neq 0^\omega$ only if $[\![\mathsf{is}_{\mathsf{zeros}}(\mathsf{run}(1, Z, \mathsf{zeros}))]\!] \neq 0^\omega$ for every assignment of the variables $\tau_1, \tau_2, \dots, \tau_{n-1}$. But then $[\![\mathsf{is}_{\mathsf{zeros}}(\mathsf{run}(1, Z, \mathsf{zeros}))]\!] = 1^\omega$ by Lemma 6.1, and as a consequence $[\![\mathsf{run}(1, Z, \mathsf{zeros})]\!] = 0^\omega$. Then it follows as in the proof of Theorem 6.14, $[\![\mathsf{run}(1, Z, \mathsf{zeros})]\!] = 0^\omega$ for all $\tau_1, \tau_2, \dots, \tau_{n-1}$ if and only if the formula in (23) holds. ☐

The proof of Theorem 6.14 immediately yields the following:

Theorem 6.16. *The following problem subsumes the analytical hierarchy:*

INPUT: *Bitstream specification E, term s.*

QUESTION: *Does s have <u>at least one</u> solution over all* full *extensional models of E?*

Proof. It suffices to observe that in the proof of Theorem 6.14, the term zeros has a solution over all models of E if and only if E has a model. ☐

Theorem 6.17. *The following problem subsumes the analytical hierarchy:*

INPUT: *Bitstream specification E, term s.*

QUESTION: *Does s have <u>precisely one</u> solution over all* full *extensional models of E?*

Proof. Again, it suffices to observe that in the proof of Theorem 6.14, the term zeros has precisely one solution if and only if E has a model. ☐

6.4 Comparing Sets of Solutions

In this section, we study the complexity of deciding whether terms have the same set of solutions over all (full) extensional models. It is easy to see that the hardness of these problems is at least that of deciding equality in all (full) extensional models. When considering all extensional models, the problem turns out Π^1_2-complete, and, thus, higher than the degree Π^1_1 of equality in all extensional models.

Remark 6.18. Let us briefly discuss the applicability of equality in all extensional models for the comparison of terms s, t that are specified in independent specifications E_s and E_t. First, we rename the symbols of one of the specifications such that $\Sigma_s \cap \Sigma_t = \{0, 1, :\}$. Thereafter, we consider the validity of $s = t$ in the union $E_s \cup E_t$.

We show on two examples that this approach does not always yield the intended results. Let E_M consist of the single equation $\mathsf{M} = 1 : \mathsf{M}$, and E_N of

$$\mathsf{N} = \mathsf{flip}(\mathsf{N}) \qquad \mathsf{flip}(0 : \sigma) = 1 : \mathsf{flip}(\sigma) \qquad \mathsf{flip}(1 : \sigma) = 0 : \mathsf{flip}(\sigma)$$

Then M has the stream of ones as its unique solution, but N has no solution. Since E_N does not have model, the union $E_\mathsf{M} \cup E_\mathsf{N}$ also does not admit one. Thus, $E_\mathsf{M} \cup E_\mathsf{N} \models_{\mathrm{ext}} \mathsf{M} = \mathsf{N}$

holds for trivial reasons. Nevertheless, we would not like to consider M and N as equivalent (at least if they are given by independent specifications).

Even if the specifications have unique solutions, a similar effect can occur. Let $\mathsf{M} = \mathsf{zeros}$ and let E_{M} consist of the following equations:

$$\begin{aligned} &\mathsf{is}_{\mathsf{zeros}}(\mathsf{nxor}(\sigma)) = \mathsf{zeros} \\ &\mathsf{nxor}(0:0:\sigma) = 1:\mathsf{nxor}(\sigma) \qquad && \mathsf{nxor}(0:1:\sigma) = 0:\mathsf{nxor}(\sigma) \\ &\mathsf{nxor}(1:0:\sigma) = 0:\mathsf{nxor}(\sigma) && \mathsf{nxor}(1:1:\sigma) = 1:\mathsf{nxor}(\sigma) \end{aligned}$$

together with the equations (18) for $\mathsf{is}_{\mathsf{zeros}}$ on page 31. Further let $\mathsf{N} = \mathsf{blink}$ and let E_{N} consist of the equation $\mathsf{blink} = 0 : 1 : \mathsf{blink}$. Both specifications have models, and zeros and blink have unique solutions. For example, E_{M} admits a model whose domain consists of all eventually constant streams. However, E_{M} rules out models for which there exist elements $x \in A_S$ with $[\![\mathsf{nxor}]\!](x) = 0^\omega$. In particular, the stream $0101\ldots$ is excluded from the domain A_S. Hence the union $E_{\mathsf{M}} \cup E_{\mathsf{N}}$ has no models, and $E_{\mathsf{M}} \cup E_{\mathsf{N}} \models_{\mathrm{ext}} \mathsf{zeros} = \mathsf{blink}$ holds trivially.

For the equality of sets of solutions over full extensional models we obtain the following theorem as an immediate consequence of the proof of Theorem 6.14:

Theorem 6.19. *The following problem subsumes the analytical hierarchy:*

INPUT: *Bitstream specifications E_s, E_t, ground terms $s, t :: S$.*

QUESTION: *Do s and t have equal solutions over all* full *extensional models, that is, $[\![s]\!]_{E_s,\mathrm{full}} = [\![t]\!]_{E_t,\mathrm{full}}$?*

Proof. Let E_s be the specification in the proof of Theorem 6.14, and $s = \mathsf{zeros}$. Then $[\![s]\!]_{E_s,\mathrm{full}} = \{0^\omega\}$ if E_s has a model, and $\varnothing$ otherwise. Let $E_t = \{\mathsf{zeros}' = 0 : \mathsf{zeros}'\}$ and $t' = \mathsf{zeros}'$, then we have $[\![t]\!]_{E_t,\mathrm{full}} = \{0^\omega\}$. Thus, $[\![s]\!]_{E_s,\mathrm{full}} = [\![t]\!]_{E_t,\mathrm{full}}$ is equivalent to $E \models_{\mathrm{ext}} \mathsf{zeros} = \mathsf{ones}$ in the proof of Theorem 6.14. ☐

We continue with the investigation of the complexity of deciding whether two terms have the same set of solutions over all extensional models. The proof of Theorem 6.6 yields only Π^1_1-hardness. In order to show Π^1_2-hardness, we employ Proposition 2.17, a result of (Castro & Cucker, 1989), stating that it is a Π^1_2-complete problem to decide whether the ω-language of a non-deterministic Turing machine contains all words $\{0,1\}^\omega$.

Therefore, we consider non-deterministic Turing machines with one-sided tapes. Without loss of generality, we may restrict the non-determinism $\delta : Q \times \Gamma \to \wp(Q \times \Gamma \times \{L,R\})$ to binary choices in each step, that is, $|\delta(q,b)| \leq 2$ for every $q \in Q$ and $b \in \{0,1\}$. (Broader choices then are simulated by sequences of binary choices.) Moreover, for our purposes, it suffices to consider Turing machines that never halt. For the ω-language, halting always corresponds to rejecting a run, and this rejection can be simulated by alternating moving forth and back eternally.

That is, a non-deterministic Turing machine $T = \langle Q, q_0, \delta_0, \delta_1 \rangle$ has two transition functions $\delta_0, \delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L,R\}$ and we allow a non-deterministic choice between these functions in each step. Note that, for modeling non-determinism in an equational specifications, we cannot take the union of the specifications $E_{\langle Q,q_0,\delta_0\rangle}$ and $E_{\langle Q,q_0,\delta_1\rangle}$, since multiple equations having the same left-hand side do not model choice, but additional restrictions on the models of the specification. To this end, we introduce a third argument for the binary function symbols $q \in Q$ in Definition 4.4. This argument then governs the non-deterministic choice. In order to model one-sided tapes, we introduce a fourth argument that stores the position on the tape, and is increased, when moving right, and decreased, when moving left. That is, we adapt Definition 4.4 to:

$$q(x, b : y, i : z, p) = q'(b' : x, y, z, 1 : p)$$
$$q(a : x, b : y, i : z, 1 : p) = q'(x, a : b' : y, z, p)$$

for $\delta_i(q,b) = \langle q', b', R\rangle$ and $\delta_i(q,b) = \langle q', b', L\rangle$, respectively. We use E_T^n to denote this specification, and R_T^n for the corresponding term rewriting system. In the initial configuration, the third argument should be an underspecified stream, allowing for any non-deterministic choice. We pass zeros as fourth argument, thereby ensuring that the head cannot move to negative tape indices.

A *run* of T on a stream $w \in \{0,1\}^\omega$ is a R_T^n rewrite sequence starting from a term $q_0(\mathsf{zeros}, \underline{w}, \underline{N}, \mathsf{zeros})$ where $N \in \{0,1\}^\omega$ determines the non-deterministic choices; here $\underline{w}$ is the term $w(0) : w(1) : \ldots$ A run of T is *complete* if every tape position $p \geq 0$ is visited (that is, positions right of the starting position), and it is *oscillating* if some tape position is visited infinitely often. A run is *accepting* if it is complete and not oscillating, that it, it visits every position $p \geq 0$ at least once, but only finitely often. The ω-language $\mathscr{L}^\omega(T)$ of T is the set of all ω-words $w \in \{0,1\}^\omega$ such that T has an accepting run w. We recall Proposition 2.17 (Castro & Cucker, 1989): the set $\{T \mid \mathscr{L}^\omega(T) = \{0,1\}^\omega\}$ is Π_2^1-complete.

We are now ready for the proof of Π_2^1-completeness of equality of the set of solutions over all extensional models. In the proof, we introduce a fifth argument for the symbol $q \in Q$ in E_T^n which enforces progress and rules out exactly the oscillating runs.

Theorem 6.20. *The following problem is Π_2^1-complete:*

INPUT: *Bitstream specifications E_s, E_t, ground terms $s, t :: S$.*

QUESTION: *Do s and t have equal solutions over all extensional models, that is, $[\![s]\!]_{E_s,\mathrm{ext}} = [\![t]\!]_{E_t,\mathrm{ext}}$?*

Proof. Let $T = \langle Q, q_0, \delta_0, \delta_1\rangle$ be a non-deterministic Turing machine. We reduce the problem in Proposition 2.17 to a decision problem for the equality of the set of solutions over all full models. We let $s = \mathsf{X}$ and define the specification E_s to consist of:

$$q_0(\mathsf{zeros}, \mathsf{X}, \mathsf{N}, \mathsf{zeros}, P) = \mathsf{zeros} \tag{24}$$
$$\mathsf{natstr}(P) = \mathsf{ones} \tag{25}$$
$$q(x, b : y, i : z, p, 1 : v) = q'(b' : x, y, z, 1 : p, v) \quad \text{for } \delta_i(q,b) = \langle q', b', R\rangle \tag{26}$$
$$q(a : x, b : y, i : z, 1 : p, 1 : v) = q'(x, a : b' : y, z, p, v) \quad \text{for } \delta_i(q,b) = \langle q', b', L\rangle \tag{27}$$
$$q(x, y, z, 1 : p, 0 : v) = 0 : q(x, y, z, p, v) \tag{28}$$
$$q(x, y, z, 0 : p, 0 : v) = \mathsf{ones} \tag{29}$$
$$q(a : x, b : y, i : z, 0 : p, 1 : v) = \mathsf{ones} \tag{30}$$

$$\text{for } \delta_i(q,b) = \langle q', b', L\rangle$$

Equation (24) starts T on the stream X with non-deterministic choices governed by N and P for enforcing progress. The streams X and N are unspecified, thus arbitrary. The equation (25) ensures that $[\![P]\!]$ contains infinitely many zeros. The equations (26) and (27) model the computation of T as discussed before, but now in each step removing the context $1 : \Box$ from the fifth argument. If the fifth argument starts with a 0, then (28) decrements the position counter (the fourth argument). Recall that the position counter determines how many steps the Turing machine T is permitted to move left. Thus, always eventually decrementing the counter rules out the oscillating runs. The equations (29) and (30) rule out models where the head move left of the envisaged progress $[\![P]\!]$.

It is important to note that for any non-oscillating run σ, we can define a function $p : \mathbb{N} \to \mathbb{N}$ such that after $p(n)$ steps, T visits only tape indices $\geq n$. Then an assignment $[\![P]\!] = 1^{p(0)}\,0\,1^{p(1)}\,0\,1^{p(2)}\,0\ldots$ in the model will permit this run to happen, that is, the head will never fall behind the envisaged progress and Equations (29) and (30) do not apply.

As a consequence, we have $[\![s]\!]_{E_s,\mathrm{ext}} = \{0,1\}^\omega$ if and only if for every $[\![\mathsf{X}]\!] \in \{0,1\}^\omega$ there exists a non-oscillating run (that is, an appropriate choice $[\![\mathsf{N}]\!]$) of T on $[\![\mathsf{X}]\!]$. Now we define $t = \mathsf{Y}$ and $E_t = \{\mathsf{Y} = \mathsf{Y}\}$ for which obviously $[\![t]\!]_{E_t,\mathrm{ext}} = \{0,1\}^\omega$. Therefore, $[\![s]\!]_{E_s,\mathrm{ext}} = [\![t]\!]_{E_t,\mathrm{ext}}$ if and only if $\mathscr{L}^\omega(T) = \{0,1\}^\omega$. This concludes the proof of Π^1_2-hardness.

For Π^0_2-membership, the problem can be characterized by the following analytical formula: $\forall\langle \mathscr{A}_s, \mathscr{A}_t\rangle.\, \exists\langle \mathscr{A}_s', \mathscr{A}_t'\rangle.\, (\mathscr{A}_s \models\!\equiv E_s \Rightarrow \mathscr{A}_t' \models\!\equiv E_t \wedge [\![s]\!]^{\mathscr{A}_s} = [\![t]\!]^{\mathscr{A}_t'}) \wedge (\mathscr{A}_t \models\!\equiv E_t \Rightarrow \mathscr{A}_s' \models\!\equiv E_s \wedge [\![t]\!]^{\mathscr{A}_t} = [\![s]\!]^{\mathscr{A}_s'})$. As in the proof of Theorem 6.6, here, it suffices to quantify over countable models. $\Box$

7 Hidden Models for Streams of Natural Numbers

We briefly study hidden models for streams of natural numbers. A $\mathbb{N}$-*stream specification* is now defined like a bitstream specification, except that the sorts are $\mathscr{S} = \{N, S\}$, and the symbols are $0 :: N$, $s :: N \to N$ and ':' of type $N \times S \to S$. We adapt the definition of hidden Σ-algebras accordingly.

Definition 7.1. A *hidden* Σ*-algebra* $\mathscr{A} = \langle A, [\![\cdot]\!]\rangle$ consists of

(i) an $\mathscr{S}$-sorted domain A and $A_N = \mathbb{N}$,
(ii) for every $f :: s_1 \times \ldots \times s_n \to s \in \Sigma$ an *interpretation* $[\![f]\!] : A_{s_1} \times \ldots A_{s_n} \to A_s$,
(iii) $0, s \in \Sigma$ with $[\![0]\!] = 0$ and $[\![s]\!](x) = x + 1$,

The definitions of behavioral equivalence and satisfaction are the same as for bitstream specifications. A slight modification of the proof of Theorem 6.9 results in the following.

Theorem 7.2. *The following problem is* Π^1_1*-complete:*

INPUT: $\mathbb{N}$*-stream specification* E*, terms* $s, t :: S$*.*

QUESTION: *Does* $E \models\!\equiv s = t$ *hold? That is, is* $s = t$ *behaviorally satisfied in all hidden models of* E*?*

Proof. We reduce the well-foundedness problem for decidable binary relations to an equality problem. Let $M \subseteq \mathbb{N} \times \mathbb{N}$ be a decidable predicate, and $T = \langle Q, q_0, \delta \rangle$ be the corresponding Turing machine. We define the following specification E:

$$\begin{aligned}
\mathsf{zeros} &= \mathsf{run}(1, \mathsf{X}) \qquad & \mathsf{unary}(0) &= \mathsf{zeros} \\
\mathsf{run}(0, \sigma) &= \mathsf{ones} & \mathsf{unary}(s(x)) &= 1 : \mathsf{unary}(x) \\
\mathsf{run}(1, \sigma) &= 0 : \mathsf{run}(T(\mathsf{zeros}; \mathsf{unary}(\mathsf{head}(\sigma)), \\
&\qquad \mathsf{unary}(\mathsf{head}(\mathsf{tail}(\sigma)))), \mathsf{tail}(\sigma))
\end{aligned}$$

together with the equations from E_T and Definition 4.2. In contrast with the proof of Theorem 6.9, X is now a stream of natural numbers. Since X is unspecified, its interpretation in the model can be an arbitrary stream of natural numbers. As in the proofs of Theorems 6.6 and 6.9, we employ X to guess an infinite path through M. Instead of $\mathsf{uhd}(\cdot)$ and $\mathsf{utl}(\cdot)$ on bitstreams, we now take $\mathsf{unary}(\mathsf{head}(\cdot))$ and $\mathsf{tail}(\cdot)$, respectively, where the function unary converts natural numbers to unary representations in forms of streams. As in the proof of Theorem 6.9, it follows that there exists a hidden Σ-algebra $\mathscr{A}$ with $\mathscr{A} \models\equiv E$ if and only if M is not well-founded. Thus, $E \models\equiv \mathsf{zeros} = \mathsf{ones}$ if and only if M is well-founded. $\square$

8 Conclusions

We have investigated different model-theoretic and rewriting based semantics of equality of infinite objects, specified by systems of equations. It turns out that the complexities for these notions vary from the low levels of the arithmetical hierarchy Π^0_1 and Π^0_2, up to Π^1_1 and Π^1_2 of the analytical hierarchy, and some even subsume the entire arithmetical and analytical hierarchy.

Apart from Π^0_1, none of these classes are recursively enumerable or co-recursively enumerable. Thus, there exists no complete proof systems for proving or disproving equality. An exception is the equality of normal forms for productive specifications for which inequalities can be recursively enumerated (Grabmayer *et al.*, 2012).

References

Aczel, P. (1988). *Non-well-founded Sets*. Lecture Notes, no. 14. Center for the Study of Language and Information, Stanford University.

Bidoit, M., Hennicker, R., & Kurz, A. (2003). Observational Logic, Constructor-Based Logic, and their Duality. *Theoretical Computer Science*, **298**, 471–510.

Böhm, C. (ed). (1975). *Lambda-Calculus and Computer Science Theory, Proceedings of the Symposium Held in Rome, March 25-27, 1975*. LNCS, vol. 37. Springer.

Buss, S.R., & Roşu, G. (2000). Incompleteness of Behavioral Logics. *Electronic Notes in Theoretical Computer Science*, **33**, 61–79.

Castro, J., & Cucker, F. (1989). Nondeterministic ω-Computations and the Analytical Hierarchy. *Logik und Grundlagen der Mathematik*, **35**, 333–342.

Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., & Talcott, C. (2003). The Maude 2.0 System. *Pages 76–87 of: Proc. Conf. on Rewriting Techniques and Applications (RTA 2003)*. Lecture Notes in Computer Science, no. 2706. Springer.

Coquand, Th. (1993). Infinite Objects in Type Theory. *Pages 62–78 of: Proc. Conf. on Types for Proofs and Programs (TYPES 1993)*. Lecture Notes in Computer Science, vol. 806. Springer.

Danielsson, N.A. (2010). Beating the Productivity Checker Using Embedded Languages. *Pages 29–48 of: Proc. Workshop on Partiality and Recursion in Interactive Theorem Provers (PAR 2010).* EPTCS, vol. 43.

Dershowitz, N., Kaplan, S., & Plaisted, D.A. (1991). Rewrite, rewrite, rewrite, rewrite, rewrite. *Theoretical Computer Science*, **83**, 71–96.

Ehrig, H., & Mahr, B. (1985). *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. EATCS Monographs on Theoretical Computer Science. Springer.

Endrullis, J., Grabmayer, C., & Hendriks, D. (2008). Data-Oblivious Stream Productivity. *Pages 79–96 of: Proc. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2008).* Lecture Notes in Computer Science, no. 5330. Springer.

Endrullis, J., Grabmayer, C., & Hendriks, D. (2009). Complexity of Fractran and Productivity. *Pages 371–387 of: Proc. Conf. on Automated Deduction (CADE 22).* LNCS, vol. 5663.

Endrullis, J., Grabmayer, C., Hendriks, D., Isihara, A., & Klop, J.W. (2010a). Productivity of Stream Definitions. *Theoretical Computer Science*, **411**, 765–782.

Endrullis, J., Grabmayer, C., Hendriks, D., Klop, J.W., & van Oostrom, V. (2010b). Unique Normal Forms in Infinitary Weakly Orthogonal Rewriting. *Pages 85–102 of: Proc. Conf. on Rewriting Techniques and Applications (RTA 2010).* LIPIcs, vol. 6.

Endrullis, J., Hendriks, D., & Klop, J.W. (2011a). Degrees of Streams. *Journal of integers*, **11B**(A6), 1–40. Proceedings of the Leiden Numeration Conference 2010.

Endrullis, J., Geuvers, H., Simonsen, J. G., & Zantema, H. (2011b). Levels of Undecidability in Rewriting. *Information and Computation*, **209**(2), 227–245.

Endrullis, J., Hendriks, D., & Klop, J.W. (2012a). Highlights in Infinitary Rewriting and Lambda Calculus. *Theoretical Computer Science*, **464**, 48–71.

Endrullis, J., Hendriks, D., & Bakhshi, R. (2012b). On the Complexity of Equivalence of Specifications of Infinite Objects. *Pages 153–164 of: Proc. ACM SIGPLAN Int. Conf. on Functional Programming (ICFP 2013).* ACM.

Endrullis, J., Hendriks, D., & Bodin, M. (2013). Circular Coinduction in Coq Using Bisimulation-Up-To Techniques. *Pages 354–369 of: Proc. Conf. on Interactive Theorem Proving (ITP 2013).* Lecture Notes in Computer Science, vol. 7998. Springer.

Finkel, O., & Lecomte, D. (2009). Decision problems for Turing machines. *Information Processing Letters*, **109**(23–24), 1223 –1226.

Friedman, D. P., & Wise, D. S. (1976). CONS Should Not Evaluate its Arguments. *Pages 257–284 of: Proc. Int. Coll. on Automata, Languages and Programming (ICALP 1976).*

Geuvers, H. (1992). Inductive and Coinductive Types with Iteration and Recursion. *Pages 193–217 of: Proc. Workshop on Types for Proofs and Programs (TYPES 1992).*

Goguen, J. (1991). Types as Theories. *Pages 357–390 of: Topology and Category Theory in Computer Science*. Oxford University. Proc. Conf. held at Oxford, June 1989.

Goguen, J., & Malcolm, G. (2000). A Hidden Agenda. *Theoretical Computer Science*, **245**(1), 55–101.

Goguen, J. A., Thatcher, J. W., Wagner, E. G., & Wright, J. B. (1977). Initial Algebra Semantics and Continuous Algebras. *Jacm*, **24**(1), 68–95.

Grabmayer, C., Endrullis, J., Hendriks, D., Klop, J.W., & Moss, L.S. (2012). Automatic Sequences and Zip-Specifications. *Pages 335–344 of: Proc. Symp. on Logic in Computer Science (LICS 2012).* IEEE Computer Society.

Harel, D. (1985). Recurring Dominoes: Making the Highly Undecidable Highly Understandable. *Pages 51–71 of: Topics in the Theory of Computation, Selected Papers of the International Conference on Foundations of Computation Theory, FCT 1983*, vol. 102. North-Holland.

Harel, D., Kozen, D., & Tiuryn, J. (2000). *Dynamic Logic*. Foundations of Computing. MIT Press.

Henderson, P., & Morris, Jr., J. H. (1976). A Lazy Evaluator. *Pages 95–103 of: Proc. ACM SIGACT-SIGPLAN Symp. on Principles on Programming Languages (POPL 1976).* ACM.

Hennicker, R. (1991). Context Induction: A Proof Principle for Behavioural Abstractions and Algebraic Implementations. *Formal Aspects of Computation*, **3**(4), 326–345.

Hinman, P.G. (1978). *Recursion-Theoretic Hierarchies.* Perspectives in Mathematical Logic, vol. 9. Springer.

Kennaway, J.R., Klop, J.W., Sleep, M.R., & de Vries, F.-J. (1997). Infinitary Lambda Calculus. *Theoretical Computer Science*, **175**(1), 93–125.

Ketema, J., & Simonsen, J.G. (2010). Infinitary combinatory reduction systems: Normalising reduction strategies. *Logical methods in computer science*, **6**(1:7), 1–35.

Klop, J.W., & de Vrijer, R. (2005). Infinitary Normalization. *Pages 169–192 of: We Will Show Them: Essays in Honour of Dov Gabbay (2).* College Publications. Available at: `ftp://ftp.cwi.nl/pub/CWIreports/SEN/SEN-R0516.pdf`.

Landin, P.J. (1965). Correspondence between Algol 60 and Church Lambda-Notation: part I. *Communications of the ACM*, **8**(2), 89–101.

Lucanu, D., Goriac, E.-I., Caltais, G., & Rosu, G. (2009). CIRC: A Behavioral Verification Tool Based on Circular Coinduction. *Pages 433–442 of: Proc. Conf. on Algebra and Coalgebra in Computer Science (CALCO 2009).* Lecture Notes in Computer Science, vol. 5728. Springer.

Malcolm, G. (1997). Hidden Algebra and Systems of Abstract Machines. *Proc. Symp. on New Models for Software Architecture (IMSA).*

Meyer, A.R., Streett, R.S., & Mirkowska, G. (1981). The Deducibility Problem in Propositional Dynamic Logic. *Pages 238–248 of:* Even, S., & Kariv, O. (eds), *Proc. 8th Coll. on Automata, Languages and Programming (ICALP 1981).* Lecture Notes in Computer Science, vol. 115.

Niqui, M. (2009). Coalgebraic Reasoning in Coq: Bisimulation and the λ-Coiteration Scheme. *Pages 272–288 of: Proc. Workshop on Types for Proofs and Programs (TYPES 2008).* Lecture Notes in Computer Science, vol. 5497. Springer.

Odifreddi, P.G. (1992). Classical Recursion Theory. *The theory of functions and sets of natural numbers, vol. 1.* Studies in Logic and the Foundations of Mathematics, vol. 125. Elsevier.

Odifreddi, P.G. (1999). Classical Recursion Theory. *The theory of functions and sets of natural numbers, vol. 2.* Studies in Logic and the Foundations of Mathematics, vol. 143. Elsevier.

Peyton-Jones, S. (2003). *Haskell 98 Language and Libraries, The Revised Report.* Cambridge University Press.

Roşu, G. (2000). *Hidden Logic.* Ph.D. thesis, University of California.

Roşu, G. (2006). Equality of Streams is a Π^0_2-complete Problem. *Pages 184–191 of: Proc. ACM SIGPLAN Conf. on Functional Programming (ICFP 2006).* ACM.

Rogers, Jr., H. (1967). *Theory of Recursive Functions and Effective Computability.* New York: McGraw-Hill.

Rutten, J. J. M. M. (2000). Universal Coalgebra: A Theory of Systems. *Theoretical Computer Science*, **249**(1), 3–80.

Rutten, J. J. M. M. (2003). Behavioural Differential Equations: a Coinductive Calculus of Streams, Automata, and Power Series. *Theoretical Computer Science*, **308**(1-3), 1–53.

Rutten, J. J. M. M. (2005). A Coinductive Calculus of Streams. *Mathematical Structures in Computer Science*, **15**, 93–147.

Shoenfield, J. R. (1971). *Degrees of Unsolvability.* North-Holland.

Sijtsma, B. A. (1989). On the Productivity of Recursive List Definitions. *ACM Transactions on Programming Languages and Systems*, **11**(4), 633–649.

Sleep, M. R., Plasmeijer, M. J., & van Eekelen, M. C. J. D. (eds). (1993). *Term graph rewriting: Theory and practice.* John Wiley.

Terese. (2003). *Term Rewriting Systems*. Cambridge University Press.

Turner, D. A. (1986). An Overview of Miranda. *SIGPLAN Notices*, **21**(12), 158–166.

Zantema, H., & Endrullis, J. (2011). Proving Equality of Streams Automatically. *Pages 393–408 of: Proc. Conf. on Rewriting Techniques and Applications (RTA 2011)*.