

## WELL-DEFINEDNESS OF STREAMS BY TRANSFORMATION AND TERMINATION

HANS ZANTEMA

Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands, and

Institute for Computing and Information Sciences, Radboud University, Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands  
*e-mail address:* H.Zantema@tue.nl

**ABSTRACT.** Streams are infinite sequences over a given data type. A stream specification is a set of equations intended to define a stream.

We propose a transformation from such a stream specification to a term rewriting system (TRS) in such a way that termination of the resulting TRS implies that the stream specification is well-defined, that is, admits a unique solution. As a consequence, proving well-definedness of several interesting stream specifications can be done fully automatically using present powerful tools for proving TRS termination.

In order to increase the power of this approach, we investigate transformations that preserve semantics and well-definedness. We give examples for which the above mentioned technique applies for the transformed specification while it fails for the original one.

### 1. INTRODUCTION

Streams are among the simplest data types in which the objects are infinite. We consider streams to be maps from the natural numbers to some data type  $D$ . Streams have been studied extensively, e.g., in [1]. The basic constructor for streams is the operator ‘:’ mapping a data element  $d$  and a stream  $s$  to a new stream  $d : s$  by putting  $d$  in front of  $s$ . Using this operator we can define streams by equations. For instance, the stream **zeros** only consisting of 0’s can be defined by the single equation **zeros** = 0 : **zeros**. More complicated streams are defined using stream functions. For instance, the boolean *Fibonacci stream* **Fib** is defined<sup>1</sup> as the limit of the strings  $\phi_i$  where  $\phi_1 = 1$ ,  $\phi_2 = 0$ ,  $\phi_{i+2} = \phi_{i+1}\phi_i$  for  $i \geq 1$ , showing the relationship with Fibonacci numbers. For  $f$  being the function replacing every 0 by 1 and every 1 by 01, one easily proves by induction on  $n$  that  $f(\phi_n) = \phi_{n+1}$  for all

*1998 ACM Subject Classification:* F.4.2, E.1.

*Key words and phrases:* term rewriting, stream specification.

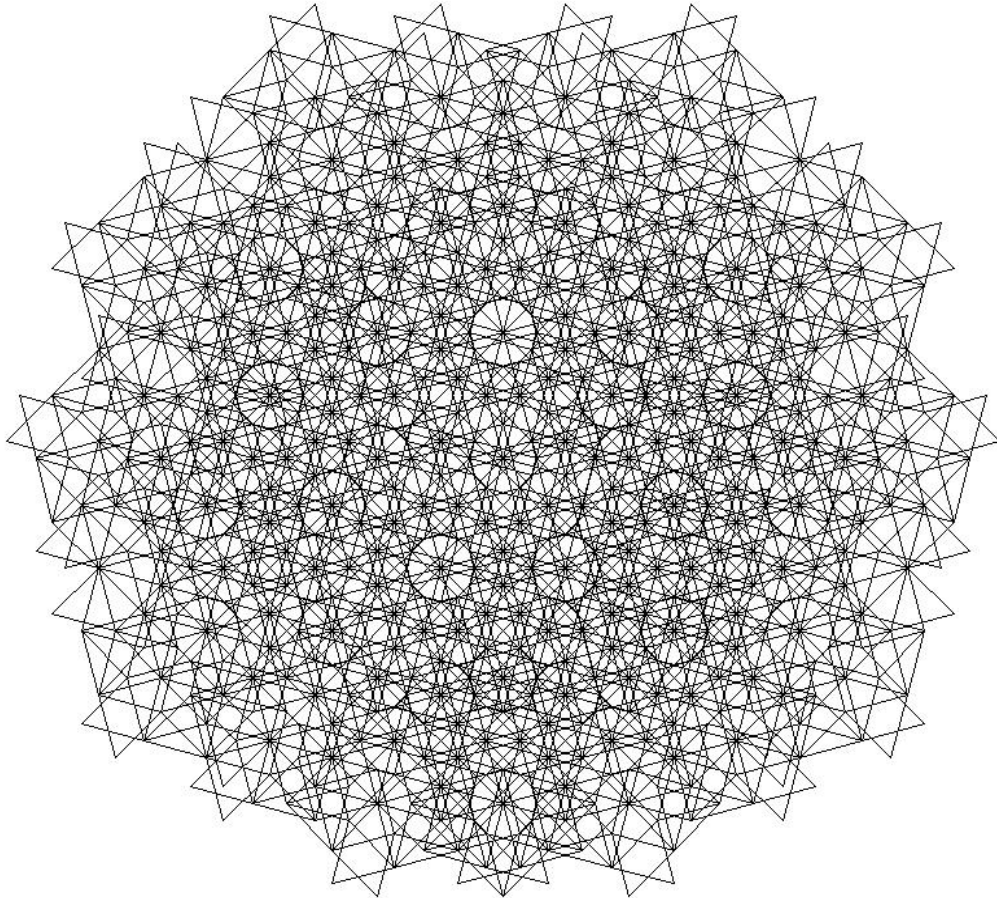
<sup>1</sup>In [1] it is called the infinite Fibonacci word.

$n \geq 1$ . As **Fib** is the limit of these strings, **Fib** is a fix point of this function  $f$  on boolean streams. So the function  $f$  and **Fib** satisfy the three equations

$$\begin{aligned} f(0 : \sigma) &= 0 : 1 : f(\sigma), \\ f(1 : \sigma) &= 0 : f(\sigma), \\ \mathbf{Fib} &= f(\mathbf{Fib}), \end{aligned}$$

for all boolean streams  $\sigma$ . In this paper we consider stream specifications consisting of such a set of equations. We address the most fundamental question one can think of: is the intended stream uniquely defined by these equations? More precisely, does such a set of equations admit a unique solution as constants and functions on streams? So in particular for **Fib**: is the boolean stream **Fib** uniquely defined by the three equations we gave? We will call this *well-defined*, and we will show that for the equations for **Fib** this indeed holds.

Although our specification of **Fib** only consists of a few very simple equations, the resulting stream is non-periodic and has remarkable properties. For instance, one can make a *turtle visualization* as follows. Choose an initial drawing direction and traverse the elements of the stream **Fib** as follows: if the symbol 0 is read then the drawing direction is moved 120 degrees to the right; if the symbol 1 is read then the drawing direction is moved 30 degrees to the left. In both cases after doing so a line of unit length is drawn. Then after 100.000 steps the following picture is obtained.



Another turtle visualization of **Fib** with different parameters was given in [19]. For turtle visualizations of similar stream specifications we refer to

<http://www.win.tue.nl/~hzantema/str.html>.

To show that well-definedness does not always hold, observe that the function  $f$  defined by

$$f(0 : \sigma) = 1 : f(\sigma), \quad f(1 : \sigma) = 0 : f(\sigma)$$

has no fixpoints, that is, adding an equation  $c = f(c)$  yields no solution for  $c$ . On the other hand, the function  $f$  defined by

$$f(0 : \sigma) = 0 : f(\sigma), \quad f(1 : \sigma) = 1 : f(\sigma)$$

is the identity with infinitely many fixpoints, yielding infinitely many solutions of the equation  $c = f(c)$ . Finally, the function  $f$  defined by

$$f(0 : \sigma) = 0 : 1 : f(\sigma), \quad f(1 : \sigma) = 1 : 0 : f(\sigma)$$

has exactly two fixpoints: the Thue-Morse stream and its inverse.

Our approach to prove well-definedness of stream specifications is based on the following idea. Derive rewrite rules from the equations in such a way that by these rules the  $n$ -th element of the stream can be computed for every  $n$ . The term rewriting systems (TRS) consisting of these rules will be orthogonal by construction, so if the computation yields a result, this result will be unique. So the remaining key point is to show that the computation always yields a result, which is the case if the TRS is terminating. The past ten years showed up a remarkable progress in techniques and implementations for proving termination of TRSs [2, 7, 14]. One of the objectives of this paper is to exploit this power for proving well-definedness of stream specifications. In our approach we introduce fresh operators **head** and **tail** intended to observe streams. We present a transformation of the specification to its *observational variant*. This is a TRS mimicking the stream specification in such a way that **head** or **tail** applied on any stream constant or stream function can always be rewritten. In particular for a stream term  $t$  it serves for computing **head**(**tail** <sup>$n-1$</sup> ( $t$ )), representing the  $n$ -th element of  $t$ . So not only a proof of well-definedness is provided, our approach also yields an algorithm to compute the  $n$ -th element of any stream term, for any  $n$ .

This transformation is straightforward and easy to implement; an implementation for boolean stream specifications is discussed in Section 6.

The main result of this paper states that if the observational variant of a stream specification is terminating, then the stream specification is well-defined. It turns out that for several interesting cases termination of the observational variant of a specification can be proved by termination tools like AProVE [6] or TTT2 [10]. This provides a new technique to prove well-definedness of stream specifications fully automatically, applying for cases where earlier approaches fail. Our main result appears in two variants:

- a variant restricting to ground terms for general stream specifications (Theorem 5.1), and
- a variant generalizing to all streams for stream specifications not depending on particular data elements (Theorem 7.1).

By an example we show that the approach does not work for general stream specifications and functions applied on all streams. Moreover, we show that our technique is not complete: the fix point definition of the Fibonacci stream **Fib** as we just gave is a well-defined stream specification for which the observational variant is non-terminating. However, we will also investigate transformations from stream specifications to stream specifications preserving

semantics, so also preserving well-definedness. Applying such a transformation to our specification of **Fib** gives an alternative specification specifying the same stream **Fib**, but for which the observational variant is terminating, to be proved automatically by a termination tool. In this way we prove well-definedness of **Fib** with respect to the original stream specification. More general, applying such semantics preserving transformations increases the power of our approach to prove well-definedness of stream specifications.

Proving well-definedness in stream specification is closely related to proving equality of streams. A standard approach for this is co-induction [16]: two streams or stream functions are equal if a bisimulation can be found between them. Finding such an arbitrary bisimulation is a hard problem in the general setting, but restricting to circular co-induction [8, 13] finding this automatically is tractable. A strong tool doing so is *Circ* [12, 11]. The tool *Circ* focuses on proving equality, but proving well-definedness of a function  $f$  can also be proved by equality as long as the equations for  $f$  are orthogonal: take a copy  $f'$  of  $f$  with the same equations, and prove  $f' = f$ . Here orthogonality is essential: if for instance a stream  $c$  has two rules  $c = 0 : c$  and  $c = 1 : c$ , then the system is non-orthogonal and admits every boolean stream as a solution, while by having a copy  $c'$  with the same rules one can prove  $c = c'$  by only using the rules  $c = 0 : c$  and  $c' = 0 : c'$ .

The input format of *Circ* differs from what we call stream specifications: in order to fit in the co-induction approach **head** and **tail** are already building blocks and the *Circ* input is essentially the same as what we call the observational variant. Our implementation as discussed in Section 6 offers the facility to transform a stream specification to *Circ* format, and also generate the equalities representing well-definedness in *Circ* format. For very simple examples the equalities can be proved automatically by *Circ*, but for several small stream specifications *Circ* fails while our approach succeeds in proving well-definedness. Conversely our approach can be helpful to prove equality of two streams: if one stream satisfies the specification of the other one, and both specifications are well-defined, then the streams are equal.

Another closely related topic is *productivity* of stream specifications, as studied by [4]. Productive stream specifications are always well-defined. Conversely we will give an example (Example 4) of a stream specification that is well-defined, but not productive. Our format of stream specifications is strongly inspired by [4]. In [4] a technique has been developed for establishing productivity of single ground terms fully automatically for a restricted class of stream specifications. In particular, only a mild type of nesting in the right-hand sides of the equation is allowed. If these restrictions hold, then the approach yields a full decision procedure for productivity, and provides a corresponding implementation by which for a wide range of examples productivity can be proved fully automatically. Productivity of a single ground term implies well-definedness of that single term. On the other hand, our technique often applies where their restrictions do not hold, or for proving well-definedness for systems that are not productive. Apart from the technique from [4] there are more results on productivity. An approach to prove productivity by means of outermost termination has been presented in [21]; a more recent approach using transformations and context-sensitive termination is presented in [20]. For both these approaches the power of present termination provers is exploited for proving productivity automatically, similar to what we do in this paper for proving well-definedness.

In [9] well-definedness of a stream specification is claimed if some particular syntactic conditions hold, like all right-hand sides of the equations have  $":$ " as its root. Their result both follows from our main theorem and from the productivity analysis of [4].

Both stream equality [15] and productivity [17] have been proved to be  $\Pi_2^0$ -complete, hence undecidable. By a similar Turing machine construction the same is expected to hold for stream well-definedness.

This paper is an extension of the RTA conference paper [19] and the corresponding tool description [18]. Compared to these papers

- some definitions have been slightly modified in order to cover a more general setting,
- the process of unfolding and other transformations preserving the semantics have been worked out in detail in Section 3 and Section 8, while in [19] only some of the ideas were sketched by examples,
- more examples are given, in particular specifying the paper folding stream and the Kolakoski stream.

The paper is structured as follows. In Section 2 we present the basics of stream specifications and their models. In Section 3 we show how a non-proper stream specification can be unfolded to a proper stream specification preserving semantics and well-definedness. In Section 4 we define the transformation of a proper stream specification to its observational variant. In Section 5 we present and prove the main theorem: if the observational variant is terminating then the specification is well-defined, that is, restricted to ground terms it has a unique model. In Section 6 we describe our implementation. In Section 7 we show that the restriction to ground terms in the main theorem may be removed in case the stream specification is data independent, that is, left-hand sides of equations do not contain data values. In Section 8 we present requirements on transformations on stream specifications for preserving semantics and well-definedness. In case the observational variant of a stream specification is not terminating, or the tools fail to prove termination, then we can apply such transformations. Often then the observational variant is terminating, proving not only well-definedness of the transformed specification, but also of the original one. One of the corresponding examples serves for proving incompleteness of our main theorem. We conclude in Section 9.

## 2. STREAMS: SPECIFICATIONS AND MODELS

In stream specifications we have two sorts:  $s$  (stream) and  $d$  (data). We assume the set  $D$  of data elements to consist of the unique normal forms of ground terms over some signature  $\Sigma_d$  with respect to some terminating orthogonal TRS  $R_d$  over  $\Sigma_d$ . Here all symbols of  $\Sigma_d$  are of type  $d^n \rightarrow d$  for some  $n \geq 0$ . We assume a particular symbol  $:$  having type  $d \times s \rightarrow s$ . For giving the actual stream specification we need a set  $\Sigma_s$  of stream symbols, each being of type  $d^n \times s^m \rightarrow s$  for  $n, m \geq 0$ . Now terms of sort  $s$  are defined inductively as follows:

- a variable of sort  $s$  is a term of sort  $s$ ,
- if  $f \in \Sigma_s$  is of type  $d^n \times s^m \rightarrow s$ ,  $u_1, \dots, u_n$  are terms over  $\Sigma_d$  and  $t_1, \dots, t_m$  are terms of sort  $s$ , then  $f(u_1, \dots, u_n, t_1, \dots, t_m)$  is a term of sort  $s$ ,
- if  $u$  is a term over  $\Sigma_d$  and  $t$  is a term of sort  $s$ , then  $u : t$  is a term of sort  $s$ .

Note that we do not allow function symbols with output sort  $d$  and input containing sort  $s$ . One reason for this is that we do not want that distinct data elements are made equal by stream equations.

An equation of sort  $s$  is a pair  $(\ell, r)$  of terms of sort  $s$ , usually written as  $\ell = r$ . An equation can also be considered as a rule in a TRS. For basic properties of TRSs we

refer to [3]. In particular, an orthogonal TRS is always confluent, from which it can be concluded that every term has at most one normal form. Here orthogonal means that the left-hand sides of the rules are non-overlapping, and every variable occurs at most once in any left-hand side.

As a notational convention variables of sort  $d$  will be denoted by  $x, y$ , terms of sort  $d$  by  $u, u_i$ , variables of sort  $s$  by  $\sigma, \tau$ , and terms of sort  $s$  by  $t, t_i$ .

**Definition 2.1.** A *stream specification*  $(\Sigma_d, \Sigma_s, R_d, R_s)$  consists of  $\Sigma_d, \Sigma_s, R_d$  as given before, and a set  $R_s$  of equations over  $\Sigma_d \cup \Sigma_s \cup \{:\}$  of sort  $s$ .

A stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  is called *proper* if all equations in  $R_s$  are of the shape

$$f(u_1, \dots, u_n, t_1, \dots, t_m) = t,$$

where

- $f \in \Sigma_s$  is of type  $d^n \times s^m \rightarrow s$ ,
- for every  $i = 1, \dots, m$  the term  $t_i$  is either a variable of sort  $s$ , or  $t_i = x : \sigma$  where  $x$  is a variable of sort  $d$  and  $\sigma$  is a variable of sort  $s$ ,
- $t$  is any term of sort  $s$ ,
- $R_s \cup R_d$  is orthogonal,
- Every term of the shape  $f(u_1, \dots, u_n, u_{n+1} : t_1, \dots, u_{n+m} : t_m)$  for  $f \in \Sigma_s$  of type  $d^n \times s^m \rightarrow s$ , and  $u_1, \dots, u_{n+m} \in D$  matches with the left-hand side of an equation from  $R_s$ .

Some parts in this definition allow modification, but for being a basis for the rest of our theory we fix this choice. All of our examples are on boolean streams, but by allowing data to be ground normal forms of a data TRS, the setting is much more general.

Sometimes we call  $R_s$  a stream specification: in that case  $\Sigma_d, \Sigma_s$  consist of the symbols of sort  $d, s$ , respectively, occurring in  $R_s$ , and  $R_d = \emptyset$ .

**Example 1.** For specifying the Thue-Morse sequence the data elements are 0, 1, and a data operation **not** is used. The data rewrite system  $R_d$  consists of the two rules **not**(0)  $\rightarrow$  1 and **not**(1)  $\rightarrow$  0. The set  $R_s$  consists of the equations

$$\begin{array}{ll} \mathbf{morse} = 0 : \mathbf{zip}(\mathbf{inv}(\mathbf{morse}), \mathbf{tail}(\mathbf{morse})) & \mathbf{tail}(x : \sigma) = \sigma \\ \mathbf{inv}(x : \sigma) = \mathbf{not}(x) : \mathbf{inv}(\sigma) & \mathbf{zip}(x : \sigma, \tau) = x : \mathbf{zip}(\tau, \sigma) \end{array}$$

This is a proper stream specification.

Definition 2.1 is closely related to the definition of stream specification in [4]. In fact there are two differences:

- We want to specify streams for every ground term of sort  $s$ , while in [4] there is a designated constant to be specified.
- Our restriction on left-hand sides of  $R_s$  in a proper stream specification is stronger than the exhaustiveness from [4]. However, by introducing fresh symbols and equations for defining these fresh symbols, every stream specification in the format of [4] can be unfolded to a proper stream specification in our format. This is worked out in Section 3.

Stream specifications are intended to specify streams for the constants in  $\Sigma_s$ , and stream functions for the other elements of  $\Sigma_s$ . The combination of these streams and stream functions is what we will call a *stream model*.

More precisely, a *stream* over  $D$  is a map from the natural numbers to  $D$ . Write  $D^\omega$  for the set of all streams over  $D$ . In case of  $D = \emptyset$  we have  $D^\omega = \emptyset$ ; in case of  $\#D = 1$  we have  $\#D^\omega = 1$ . So in non-degenerate cases we have  $\#D \geq 2$ .

It seems natural to require that stream functions in a stream model are defined on all streams. However, it turns out that several desired properties do not hold when requiring this. Therefore we allow stream functions to be defined on some set  $S \subseteq D^\omega$  for which every ground term can be interpreted in  $S$ .

**Definition 2.2.** A *stream model* is defined to consist of a set  $S \subseteq D^\omega$  and a set of functions  $[f]$  for every  $f \in \Sigma_s$ , where  $[f] : D^n \times S^m \rightarrow S$  if the type of  $f \in \Sigma_s$  is  $d^n \times s^m \rightarrow s$ .

For a ground term  $u$  over  $\Sigma_d$  write  $\mathbf{NF}(u)$  for its  $R_d$ -normal form. We write  $\mathbf{T}_s$  for the set of ground terms of sort  $s$  over  $\Sigma_d \cup \Sigma_s \cup \{:\}$ . For  $t \in \mathbf{T}_s$  the stream interpretation  $[t]$  in the stream model  $(S, ([f])_{f \in \Sigma_s})$  is defined inductively by:

$$\begin{aligned} [f(u_1, \dots, u_n, t_1, \dots, t_m)] &= [f]([u_1], \dots, [u_n], [t_1], \dots, [t_m]) && \text{for } f \in \Sigma_s \\ [f(u_1, \dots, u_n)] &= \mathbf{NF}(f(u_1, \dots, u_n)) && \text{for } f \in \Sigma_d \\ [u : t](0) &= [u] \\ [u : t](i) &= [t](i - 1) && \text{for } i > 0 \end{aligned}$$

for all ground terms  $u, u_i$  of sort  $d$  and all ground terms  $t, t_i$  of sort  $s$ .

So in a stream model:

- every data operator is interpreted by its corresponding term constructor, after which the result is reduced to normal form,
- every stream operator  $f$  is interpreted by the given function  $[f]$ , and
- the operator  $:$  applied on a data element  $d$  and a stream  $s$  is interpreted by putting  $d$  on the first position and shifting every stream element of  $s$  to its next position.

Any stream model  $(S, ([f])_{f \in \Sigma_s})$  can be restricted to a stream model  $(S', ([f])_{f \in \Sigma_s})$  satisfying  $S' \subseteq S$  and  $S' = \{[t] \mid t \in \mathbf{T}_s\}$ , note that from  $S' = \{[t] \mid t \in \mathbf{T}_s\}$  we conclude that  $S'$  is closed under  $[f]$  for every  $f \in \Sigma_s$ .

**Definition 2.3.** A stream model  $(S, ([f])_{f \in \Sigma_s})$  is said to *satisfy* a stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  if  $[\ell\rho] = [r\rho]$  for every equation  $\ell = r$  in  $R_s$  and every ground substitution  $\rho$ . We also say that the specification *admits* the model.

If a stream model  $(S, ([f])_{f \in \Sigma_s})$  satisfies a stream specification, then the stream model  $(S', ([f])_{f \in \Sigma_s})$  defined by  $S' = \{[t] \mid t \in \mathbf{T}_s\}$  satisfies the same stream specification by definition.

**Definition 2.4.** A stream specification is *well-defined* if there is exactly one stream model  $(S, ([f])_{f \in \Sigma_s})$  satisfying the stream specification for which  $S = \{[t] \mid t \in \mathbf{T}_s\}$ .

One can wonder why to restrict to  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . Another option would be simply state  $S = D^\omega$ . However, sometimes restricting to ground terms yields a unique model, while functions applied on arbitrary streams are not unique. In Example 3 we will see an example of this phenomenon. By restricting to interpretations of ground terms and ignoring unreachable streams, we arrived at our definition of well-definedness.

Not every proper stream specification is well-defined: if  $\#D > 1$  and  $R_s$  only consists of the equation  $c = c$  then every stream  $[c]$  satisfies the specification. Less trivial is the boolean stream specification

$$c = 0 : f(c), \quad f(x : \sigma) = \sigma,$$

in which  $[f]$  can be chosen to be the tail function and  $[c]$  be any stream starting with 0, yielding several stream models. There are also proper stream specifications with no model, for instance

$$\begin{aligned} c &= f(c), & f(x : \sigma) &= g(x, \sigma), \\ g(0, \sigma) &= 1 : \sigma, & g(1, \sigma) &= 0 : \sigma \end{aligned}$$

Here  $[f(c)]$  starts with 1 if  $[c]$  starts with 0, and conversely, contradicting  $[c] = [f(c)]$ .

### 3. UNFOLDING STREAM SPECIFICATIONS

The specification of the function  $f$

$$f(0 : \sigma) = 0 : 1 : f(\sigma), \quad f(1 : \sigma) = 0 : f(\sigma)$$

in the introduction to define **Fib** does not meet the requirements of a proper stream specification since the argument  $0 : \sigma$  in the left-hand side  $f(0 : \sigma)$  is not of the right shape. Introducing a fresh symbol  $g$  and unfolding yields

$$\begin{aligned} f(x : \sigma) &= g(x, \sigma) & g(0, \sigma) &= 0 : 1 : f(\sigma) \\ & & g(1, \sigma) &= 0 : f(\sigma) \end{aligned}$$

satisfying the requirements of a proper stream specification. In this section we precisely define this unfolding and show that it does not influence well-definedness.

Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  be a stream specification in which  $R_s$  contains an equation

$$f(u_1, \dots, u_n, t_1, \dots, t_m) = t,$$

where  $f \in \Sigma_s$  is of type  $d^n \times s^m \rightarrow s$ , and for some  $i \in \{1, \dots, m\}$  the term  $t_i$  is of the shape  $t_i = u : t'$  where not both  $u$  and  $t'$  are variables, so the stream specification is not proper. Then the *unfolded* stream specification on  $f$  on position  $i$ , denoted as  $\mathbf{Unf}_{f,i}(\Sigma_d, \Sigma_s, R_d, R_s)$ , is obtained by adding a fresh symbol  $g$  of type  $d^{n+1} \times s^m \rightarrow s$  to  $\Sigma_s$ , adding an equation

$$f(x_1, \dots, x_n, \sigma_1, \dots, \sigma_m) = g(x_1, \dots, x_{n+1}, \sigma_1, \dots, \sigma_m)$$

to  $R_s$ , where  $x_{n+1} : \sigma_i$  is in the  $i$ -th stream position of  $f$ , and in which every equation in  $R_s$  of the shape

$$f(u_1, \dots, u_n, t_1, \dots, u : t', \dots, t_m) = t$$

where  $u : t'$  is on the  $i$ -th stream position of  $f$ , is replaced by

$$g(u_1, \dots, u_n, u, t_1, \dots, t', \dots, t_m) = t,$$

where  $t'$  is on the  $i$ -th stream position of  $g$ .

Applying  $\mathbf{Unf}_{f,1}$  on the **Fib** stream specification from the introduction yields

$$\begin{aligned} f(x : \sigma) &= g(x, \sigma) & g(0, \sigma) &= 0 : 1 : f(\sigma) \\ \mathbf{Fib} &= f(\mathbf{Fib}) & g(1, \sigma) &= 0 : f(\sigma) \end{aligned}$$

which is indeed a proper stream specification.

In general, for every exhaustive stream specification in the sense of [4], by repeatedly applying  $\mathbf{Unf}_{f,i}$  for various  $f, i$ , as long as an equation of the shape  $f(u_1, \dots, u_n, t_1, \dots, u : t', \dots, t_m) = t$  exists for which not both  $u$  and  $t'$  are variables, a proper stream specification in our sense can be obtained.

In order to justify this unfolding it remains to prove that the original stream specification is well-defined if and only if the unfolded variant is well-defined, and in case of well-definedness they define the same. More precisely, we prove that the transformation  $\mathbf{Unf}_{f,i}$  *preserves* semantics, defined as follows.



**Definition 3.1.** A transformation  $\Phi$  mapping a stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  to  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$  satisfying  $\Sigma_s \subseteq \Sigma'_s$  is said to *preserve semantics* if

- $(\Sigma_d, \Sigma_s, R_d, R_s)$  is well-defined if and only if  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$  is well-defined, and
- If  $(\Sigma_d, \Sigma_s, R_d, R_s)$  is well-defined with corresponding model  $(S, [\cdot])$ , and  $(S', [\cdot]')$  is the model corresponding to  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$ , then  $[t] = [t]'$  for all ground terms of sort  $s$  over  $\Sigma_d \cup \Sigma_s$ .

Obviously, preservation of semantics is closed under composition of such transformations.

We prove that  $\text{Unf}_{f,i}$  preserves semantics in two steps: first we only add the equation for the fresh symbol  $g$ , and then we do the replacement of the  $f$ -equations by  $g$ -equations. For each of these two steps we show by a more general lemma that semantics is preserved.

**Lemma 3.2.** *Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  be a stream specification. Let  $g \notin \Sigma_s$  be of type  $d^{n+1} \times s^m \rightarrow s$ . Let  $R'_s$  be the union of  $R_s$  and an equation*

$$t = g(x_1, \dots, x_{n+1}, \sigma_1, \dots, \sigma_m)$$

*in which the symbol  $g$  does not occur in  $t$ , and  $t$  does not contain variables other than  $x_1, \dots, x_{n+1}, \sigma_1, \dots, \sigma_m$ . Then transforming  $(\Sigma_d, \Sigma_s, R_d, R_s)$  to  $(\Sigma_d, \Sigma_s \cup \{g\}, R_d, R'_s)$  preserves semantics.*

*Proof.* First assume that the stream model  $(S, ([f])_{f \in \Sigma_s})$  satisfies the stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  and  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . For  $s_1, \dots, s_m \in S$  choose  $t_1, \dots, t_m \in \mathbf{T}_s$  such that  $s_i = [t_i]$  for  $i = 1, \dots, m$ . Now for  $d_1, \dots, d_{n+1} \in D$  define

$$[g](d_1, \dots, d_{n+1}, s_1, \dots, s_m) = [t\rho]$$

for  $\rho$  defined by  $\rho(x_i) = d_i$  for  $i = 1, \dots, n+1$  and  $\rho(\sigma_i) = s_i$  for  $i = 1, \dots, m$ . Due to the compositional shape of the definition of  $[f]$  for  $f \in \Sigma_s$  this definition of  $g$  is independent of the choice of  $t_1, \dots, t_m \in \mathbf{T}_s$ . By construction this yields a stream model  $(S, ([f])_{f \in \Sigma_s \cup \{g\}})$  satisfying  $(\Sigma_d, \Sigma_s \cup \{g\}, R_d, R'_s)$  and  $S = \{[t] \mid t \in \mathbf{T}_s\}$ , where in the latter  $\mathbf{T}_s$  stands for ground terms including the symbol  $g$ .

Conversely, assume we have a stream model  $(S, ([f])_{f \in \Sigma_s \cup \{g\}})$  satisfying  $(\Sigma_d, \Sigma_s \cup \{g\}, R_d, R'_s)$  and  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . Then by ignoring  $g$  it is also a stream model satisfying  $(\Sigma_d, \Sigma_s, R_d, R_s)$ . Due to the shape of the equation containing  $g$ , for every ground term  $t'$  containing  $g$  there is a ground term  $t''$  not containing  $g$  satisfying  $[t''] = [t']$ . So we also have  $S = \{[t] \mid t \in \mathbf{T}_s\}$  for  $\mathbf{T}_s$  standing for the ground terms not containing  $g$ .

Summarizing, a model for  $(\Sigma_d, \Sigma_s, R_d, R_s)$  yields a model for  $(\Sigma_d, \Sigma_s \cup \{g\}, R_d, R'_s)$  and conversely, keeping the same set  $S$  and both satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . This proves the first requirement of semantics preservation. The second requirement holds since the interpretations of ground terms are the same in both models.  $\square$

In Lemma 3.2 the signature was extended by a fresh symbol, while except for adding one equation for this fresh symbol, the equations remained the same. In the next lemma it is the other way around: now the signature remains the same and the equations may be modified. For a set  $R$  of equations we write  $=_R$  for the congruence generated by  $R$ , that is, the closure of  $R$  under substitutions, contexts, reflexivity, symmetry and transitivity.

**Lemma 3.3.** *Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  and  $(\Sigma_d, \Sigma_s, R_d, R'_s)$  be stream specifications satisfying  $\ell =_{R'_s} r$  for all  $\ell = r$  in  $R_s$ , and  $\ell =_{R_s} r$  for all  $\ell = r$  in  $R'_s$ . Then transforming  $(\Sigma_d, \Sigma_s, R_d, R_s)$  to  $(\Sigma_d, \Sigma_s, R_d, R'_s)$  preserves semantics.*

*Proof.* From the given connection between  $R_s$  and  $R'_s$  it is immediate that a model satisfies  $(\Sigma_d, \Sigma_s, R_d, R_s)$  if and only if it satisfies  $(\Sigma_d, \Sigma_s, R_d, R'_s)$ . From this the lemma follows.  $\square$

**Theorem 3.4.** *The transformation  $\mathbf{Unf}_{f,i}$  preserves semantics on stream specifications on which it is defined.*

*Proof.* The operation  $\mathbf{Unf}_{f,i}$  consists of two steps: the addition of an equation generating  $g$  and the replacement of existing equations for  $f$ . The addition preserves semantics due to Lemma 3.2. For the replacement Lemma 3.3 applies, for both directions applying the equation

$$f(x_1, \dots, x_n, \sigma_1, \dots, \sigma_m) = g(x_1, \dots, x_{n+1}, \sigma_1, \dots, \sigma_m).$$

As both transformations preserve semantics, the same holds for the composition  $\mathbf{Unf}_{f,i}$ .  $\square$

#### 4. THE OBSERVATIONAL VARIANT

We define a transformation **Obs** transforming the original set of equations  $R_s$  in a proper stream specification to its *observational variant*  $\mathbf{Obs}(R_s)$ , being a TRS. The basic idea is that the streams are observed by two auxiliary operators **head** and **tail**, of which **head** picks the first element of the stream and **tail** removes the first element from the stream, and that for every  $t \in \mathbf{T}_s$  of type stream both **head**( $t$ ) and **tail**( $t$ ) can be rewritten by  $\mathbf{Obs}(R_s)$ .

The main result of this paper is that if  $\mathbf{Obs}(R_s) \cup R_d$  is terminating for a given proper stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$ , then the specification is well-defined, that is, it admits a unique model  $(S, ([f])_{f \in \Sigma_s})$  satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . As a consequence, the specification uniquely defines a corresponding stream  $[t]$  for every  $t \in \mathbf{T}_s$ .

We define  $\mathbf{Obs}(R_s)$  in two steps. First we define  $\mathbf{P}(R_s)$  obtained from  $R_s$  by modifying the equations as follows. By definition every equation of  $R_s$  is of the shape

$$f(u_1, \dots, u_n, t_1, \dots, t_m) = t$$

where for every  $i = 1, \dots, m$  the term  $t_i$  is either a variable of sort  $s$ , or  $t_i = x : \sigma$  where  $x$  is a variable of sort  $d$  and  $\sigma$  is a variable of sort  $s$ . In case  $t_i = x : \sigma$  then in the left-hand side of the equation the subterm  $t_i$  is replaced by  $\sigma$ , while in the right-hand side of the equation every occurrence of  $x$  is replaced by **head**( $\sigma$ ) and every occurrence of  $\sigma$  is replaced by **tail**( $\sigma$ ).

For example, the equation for **zip** in Example 1 will be replaced by

$$\mathbf{zip}(\sigma, \tau) \rightarrow \mathbf{head}(\sigma) : \mathbf{zip}(\tau, \mathbf{tail}(\sigma)).$$

Now we are ready to define **Obs**.

**Definition 4.1.** Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  be a proper stream specification; **tail**  $\notin \Sigma$ . Let  $\mathbf{P}(R_s)$  be defined as above. Then  $\mathbf{Obs}(R_s)$  is the TRS over  $(\Sigma_d \cup \Sigma_s) \cup \{:, \mathbf{head}, \mathbf{tail}\}$  consisting of

- the two rules

$$\mathbf{head}(x : \sigma) \rightarrow x, \quad \mathbf{tail}(x : \sigma) \rightarrow \sigma,$$

- for every rule in  $\mathbf{P}(R_s)$  of the shape  $\ell \rightarrow u : t$  the two rules

$$\mathbf{head}(\ell) \rightarrow u, \quad \mathbf{tail}(\ell) \rightarrow t,$$

- for every rule in  $\mathbf{P}(R_s)$  of the shape  $\ell \rightarrow r$  with  $\mathbf{root}(r) \neq :$  the two rules

$$\mathbf{head}(\ell) \rightarrow \mathbf{head}(r), \quad \mathbf{tail}(\ell) \rightarrow \mathbf{tail}(r).$$

The reason for first transforming  $R_s$  to  $\mathbf{P}(R_s)$  is that for the validity of the main theorem we need the special shape of the rules of  $\mathbf{Obs}(R_s)$  in which apart from the root symbol **head** or **tail** and one symbol from  $\Sigma_s$ , every left-hand sides only consists of variables.

**Example 2.** For the set  $R_s$  of equations given in Example 1 we rename the symbol **tail** by **tail0** in order to keep the symbol **tail** for the fresh symbol introduced in the **Obs** construction. Then the TRS  $\mathbf{Obs}(R_s)$  consists of the following rules:

$$\begin{array}{ll}
\mathbf{head}(x : \sigma) \rightarrow x & \mathbf{head}(\mathbf{tail0}(\sigma)) \rightarrow \mathbf{head}(\mathbf{tail}(\sigma)) \\
\mathbf{tail}(x : \sigma) \rightarrow \sigma & \mathbf{tail}(\mathbf{tail0}(\sigma)) \rightarrow \mathbf{tail}(\mathbf{tail}(\sigma)) \\
\mathbf{head}(\mathbf{morse}) \rightarrow 0 & \mathbf{head}(\mathbf{zip}(\sigma, \tau)) \rightarrow \mathbf{head}(\sigma) \\
\mathbf{tail}(\mathbf{morse}) \rightarrow \mathbf{zip}(\mathbf{inv}(\mathbf{morse}), \mathbf{tail0}(\mathbf{morse})) & \mathbf{tail}(\mathbf{zip}(\sigma, \tau)) \rightarrow \mathbf{zip}(\tau, \mathbf{tail}(\sigma)) \\
\mathbf{head}(\mathbf{inv}(\sigma)) \rightarrow \mathbf{not}(\mathbf{head}(\sigma)) & \\
\mathbf{tail}(\mathbf{inv}(\sigma)) \rightarrow \mathbf{inv}(\mathbf{tail}(\sigma)) &
\end{array}$$

Together with the rules  $\mathbf{not}(0) \rightarrow 1$  and  $\mathbf{not}(1) \rightarrow 0$  from  $R_d$  this TRS is terminating as can easily be proved fully automatically by AProVE [6] or TTT2 [10]. As a consequence, the result of this paper states that the specification uniquely defines a stream for every ground term of type  $s$ , in particular for **morse**.

## 5. THE MAIN THEOREM

We start this section by presenting our main theorem.

**Theorem 5.1.** *Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  be a proper stream specification for which the TRS  $\mathbf{Obs}(R_s) \cup R_d$  is terminating. Then the stream specification is well-defined.*

Recall that a stream specification is defined to be well-defined if it admits a unique model  $(S, ([f])_{f \in \Sigma_s})$  satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . Before proving the theorem we show by an example why it is essential to restrict to  $S = \{[t] \mid t \in \mathbf{T}_s\}$  rather than choosing  $S = D^\omega$ . A degenerate example is obtained if there are no constants of sort  $s$ , and hence  $\mathbf{T}_s = \emptyset$ . More interesting is the following.

**Example 3.** Consider the proper boolean stream specification with  $R_d = \emptyset$  and  $R_s$  consists of:

$$\begin{array}{ll}
c = 1 : c & f(x : \sigma) = g(x, \sigma) \\
g(0, \sigma) = f(\sigma) & \\
g(1, \sigma) = 1 : f(\sigma) &
\end{array}$$

obtained by unfolding

$$\begin{array}{ll}
c = 1 : c & f(0 : \sigma) = f(\sigma) \\
f(1 : \sigma) = 1 : f(\sigma) &
\end{array}$$

The function  $f$  has been specified in such a way that it tries to remove all 0's from its argument. So for streams specified by terms like  $f(c)$  there is nothing to remove, and we expect well-definedness: the term  $f(c)$  will uniquely be defined to be the stream of only ones. However, for streams containing only finitely many 1's this may be problematic. Note that by the symbols  $c, :, 0$  and  $1$  only the streams with finitely many 0's can be constructed, so for ground terms over the symbols occurring in the specification this problem does not arise. Indeed, it turns out that the TRS  $\mathbf{Obs}(R_s) \cup R_d$  is terminating, so by Theorem 5.1 the specification is well-defined. It is interesting to remark that the approach from [4] fails

to prove productivity, as this stream specification is not *data-obliviously productive*, i.e., the identity of the data is essential for productivity. Moreover, also Circ [11] fails to prove well-definedness of this stream specification.

We concluded that this example is well-defined: it admits a unique model  $(S, ([f])_{f \in \Sigma_s})$  satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . However, when extending to all streams the function  $[f] : D^\omega \rightarrow D^\omega$  is not uniquely defined, even if we strengthen the requirement of  $[\ell\rho] = [r\rho]$  for all equations  $\ell = r$  and all ground substitutions  $\rho$  to an open variant in which the  $\sigma$ 's in the equations are replaced by arbitrary streams. Write **ones** and **zeros** for the streams only consisting of ones, resp. zeros. Two distinct models  $[\cdot]_1$  and  $[\cdot]_2$  satisfying the stream specification are defined by:

$$[c]_1 = [f]_1(s) = [g]_1(u, s) = \mathbf{ones} \text{ for all } s \in D^\omega, u \in D,$$

and  $[c]_2 = \mathbf{ones}$ , and  $[f]_2(s) = [g]_2(u, s) = \mathbf{ones}$  for  $u \in D$  and streams  $s$  containing infinitely many ones, and  $[f]_2(s) = 1^n : \mathbf{zeros}$ ,  $[g]_2(u, s) = [f]_2(u : s)$  for  $u \in D$  and streams  $s$  containing  $n < \infty$  ones.

Now we arrive at the proof of Theorem 5.1. The plan of the proof is as follows.

- First we construct a function  $[\cdot]_1 : \mathbf{T}_s \rightarrow D^\omega$ , and choose  $S_1 = \{[t]_1 \mid t \in \mathbf{T}_s\}$ .
- Next we show that if  $[t_i]_1 = [t'_i]_1$  for  $i = 1, \dots, m$ , then

$$[f(u_1, \dots, u_n, t_1, \dots, t_m)]_1 = [f(u_1, \dots, u_n, t'_1, \dots, t'_m)]_1,$$

by which  $[f]_1$  is well-defined and we have a model  $(S_1, ([f]_1)_{f \in \Sigma_s})$ .

- We show this model satisfies the specification.
- We show that no other model  $(S, ([f])_{f \in \Sigma_s})$  with  $S = \{[t] \mid t \in \mathbf{T}_s\}$  satisfies the specification.

First we define  $[t]_1 \in D^\omega$  for any  $t \in \mathbf{T}_s$ . Since elements of  $D^\omega$  are functions from  $\mathbf{N}$  to  $D$ , a function  $[t]_1 \in D^\omega$  is defined by defining  $[t]_1(n)$  for every  $n \in \mathbf{N}$ . Due to the assumption of the theorem the TRS  $\mathbf{Obs}(R_s) \cup R_d$  is terminating. According to the definition of a proper stream specification the TRS  $R_s \cup R_d$  is orthogonal, and by the construction  $\mathbf{Obs}$  the TRS  $\mathbf{Obs}(R_s) \cup R_d$  is orthogonal, too. So it is confluent. Since we assume termination, we conclude that every ground term of sort  $d$  has a unique normal form with respect to  $\mathbf{Obs}(R_s) \cup R_d$ .

Assume such a normal form of sort  $d$  contains a symbol from  $\Sigma_s \cup \{:\}$ . Choose such a symbol with minimal position, that is, closest to the root. Since the term is of sort  $d$ , this symbol is not the root. Hence it has a parent. Due to minimality of position, this parent is either **head** or **tail**. Due to the shape of the rules of  $\mathbf{Obs}(R_s)$ , a rule of  $\mathbf{Obs}(R_s)$  is applicable on this parent position, contradicting the normal form assumption. So the normal form only contains symbols from  $\Sigma_d$ . Since it is also a normal form with respect to  $R_d$ , such a normal form is an element of  $D$ . Now for  $t \in \mathbf{T}_s$  and  $n \in \mathbf{N}$  we define

$$[t]_1(n) = \text{the normal form of } \mathbf{head}(\mathbf{tail}^n(t)) \text{ with respect to } \mathbf{Obs}(R_s) \cup R_d,$$

in this way defining  $[t]_1 \in D^\omega$ .

**Lemma 5.2.** *Let  $\mathbf{Obs}(R_s) \cup R_d$  be terminating. Let  $f \in \Sigma_s$  of type  $d^n \times s^m \rightarrow s$ . Let  $u_1, \dots, u_n \in D$  and  $t_1, \dots, t_m, t'_1, \dots, t'_m \in \mathbf{T}_s$  satisfying  $[t_i]_1 = [t'_i]_1$  for  $i = 1, \dots, m$ . Then*

$$[f(u_1, \dots, u_n, t_1, \dots, t_m)]_1 = [f(u_1, \dots, u_n, t'_1, \dots, t'_m)]_1.$$

*Proof.* First we extend the definition of  $[\cdot]_1$  to all ground terms over  $\Sigma_s \cup \Sigma_d \cup \{:, \mathbf{head}, \mathbf{tail}\}$ . For ground terms  $t$  of sort  $s$  we define it by  $[t]_1(n) = \text{the normal form of } \mathbf{head}(\mathbf{tail}^n(t))$  with

respect to  $\mathbf{Obs}(R_s) \cup R_d$ , and for ground terms  $u$  of sort  $d$  we define  $[u]_1$  to be the normal form of  $u$  with respect to  $\mathbf{Obs}(R_s) \cup R_d$ . We prove the following claim.

**Claim 1:** Let  $[t]_1 = [t']_1$  for  $t, t' \in \mathbf{T}_s$ . Let  $T$  be a ground term over  $\Sigma_s \cup \Sigma_d \cup \{:, \mathbf{head}, \mathbf{tail}\}$  of sort  $s$  containing  $t$  as a subterm. Let  $T'$  be obtained from  $T$  by replacing zero or more occurrences of the subterm  $t$  by  $t'$ . Then

$$[\mathbf{head}(T)]_1 = [\mathbf{head}(T')]_1.$$

Let  $>$  be the well-founded order on ground terms being the strict part of  $\geq$  defined by

$$v \geq v' \iff v' \text{ is a subterm of } v'' \text{ such that } v \rightarrow^*_{\mathbf{Obs}(R_s) \cup R_d} v''.$$

We prove the claim for every such term  $\mathbf{head}(T)$  by induction on  $>$ .

Claim 1 is trivial if  $t = T$ , so we may assume that  $T = f(u_1, \dots, u_n, t_1, \dots, t_m)$  such that  $t$  occurs in  $u_1, \dots, u_n, t_1, \dots, t_m$ , and either  $f \in \Sigma_s \cup \{:, \mathbf{tail}\}$ , and  $T' = f(u'_1, \dots, u'_n, t'_1, \dots, t'_m)$ . For every subterm of  $u_i$  of the shape  $\mathbf{head}(\dots)$  we may apply the induction hypothesis, yielding  $[u_i]_1 = [u'_i]_1 = d_i$  for all  $i$ , defining  $d_i \in D$ .

In case the root of  $T$  is not  $\mathbf{tail}$  we rewrite

$$\mathbf{head}(T) \rightarrow^*_{\mathbf{Obs}(R_s) \cup R_d} \mathbf{head}(f(d_1, \dots, d_n, t_1, \dots, t_m)),$$

and then continue by the rule  $\mathbf{head}(f(\dots)) \rightarrow \dots$  in  $\mathbf{Obs}(R_s)$ , yielding a term  $U$  of sort  $d$ . As  $\mathbf{head}$  is the only symbol of sort  $d$  having an argument of sort  $s$ , the only way such a term can contain  $t$  as a subterm is by  $U = C[\mathbf{head}(V_1), \dots, \mathbf{head}(V_k)]$  where  $t$  is a subterm of some of the  $V_i$  and  $C$  is composed from  $\Sigma_d$ . Similarly, we obtain

$$\mathbf{head}(T') \rightarrow^*_{\mathbf{Obs}(R_s) \cup R_d} \mathbf{head}(f(d_1, \dots, d_n, t'_1, \dots, t'_m)) \rightarrow C[\mathbf{head}(V'_1), \dots, \mathbf{head}(V'_k)],$$

for  $V'_i$  obtained from  $V_i$  by replacing zero or more occurrences of  $t$  by  $t'$ . By the induction hypothesis we obtain  $[\mathbf{head}(V_i)]_1 = [\mathbf{head}(V'_i)]_1$ . So  $[\mathbf{head}(V_i)]$  and  $[\mathbf{head}(V'_i)]$  rewrite to the same normal form for all  $i$ . Hence

$$[\mathbf{head}(T)]_1 = [C[\mathbf{head}(V_1), \dots, \mathbf{head}(V_k)]]_1 = [C[\mathbf{head}(V'_1), \dots, \mathbf{head}(V'_k)]]_1 = [\mathbf{head}(T')]_1,$$

which we had to prove.

In case the root of  $T$  is  $\mathbf{tail}$  then write

$$T = \mathbf{tail}^i(f(\dots)) \rightarrow^*_{\mathbf{Obs}(R_s) \cup R_d} \mathbf{tail}^i(f(d_1, \dots, d_n, t_1, \dots, t_m))$$

for  $f \in \Sigma_s \cup \{:\}$ . This can be rewritten by the rule  $\mathbf{tail}(f(\dots)) \rightarrow \dots$  in  $\mathbf{Obs}(R_s)$ , yielding  $V$ . Note that for applicability of this rule it is essential that the arguments of  $f$  in the left-hand side are variables, which was achieved by first applying the transformation  $\mathbf{P}$ .

On the same position using the same rule we can rewrite  $T' \rightarrow_{\mathbf{Obs}(R_s)} V'$  for  $V'$  obtained from  $V$  by replacing one or more occurrences of  $t$  by  $t'$ . Applying the induction hypothesis gives  $[\mathbf{head}(V)]_1 = [\mathbf{head}(V')]_1$  yielding

$$[\mathbf{head}(T)]_1 = [\mathbf{head}(V)]_1 = [\mathbf{head}(V')]_1 = [\mathbf{head}(T')]_1,$$

concluding the proof of Claim 1.

**Claim 2:** Let  $[t]_1 = [t']_1$  for  $t, t' \in \mathbf{T}_s$ . Let  $T$  be a ground term over  $\Sigma_s \cup \Sigma_d \cup \{:, \mathbf{head}, \mathbf{tail}\}$  of sort  $s$  containing  $t$  as a subterm. Let  $T'$  be obtained from  $T$  by replacing one or more occurrences of the subterm  $t$  by  $t'$ . Then  $[T]_1 = [T']_1$ .

Claim 2 easily follows from Claim 1 and the observation

$$[T]_1 = [T']_1 \iff \forall i \in \mathbf{N} : [\mathbf{head}(\mathbf{tail}^i(T))]_1 = [\mathbf{head}(\mathbf{tail}^i(T'))]_1.$$

Now the lemma follows by applying Claim 2 and replacing  $t_i$  by  $t'_i$  successively for  $i = 1, \dots, m$ .  $\square$

Define  $S_1 = \{[t]_1 \mid t \in \mathbf{T}_s\}$ . For any  $f \in \Sigma_s$  of type  $d^n \times s^m \rightarrow s$  for  $u_1, \dots, u_n \in D$  and  $t_1, \dots, t_m, t'_1, \dots, t'_m \in \mathbf{T}_s$  we now define  $[f]_1 : D^n \times S^m \rightarrow S$  by

$$[f]_1(u_1, \dots, u_n, [t_1], \dots, [t_m]) = [f(u_1, \dots, u_n, t_1, \dots, t_m)]_1;$$

Lemma 5.2 implies that this is well-defined: the result is independent of the choice of the representants in  $[t_i]_1$ . So  $(S_1, ([f]_1)_{f \in \Sigma_s})$  is a model.

Next we will prove that it satisfies the specification, and essentially is the only one doing so.

**Lemma 5.3.** *Let  $\ell \rightarrow r \in R_s$  and let  $\rho$  be a substitution. Then*

- *there is a term  $t$  such that  $\mathbf{head}(\ell\rho) \rightarrow^*_{\mathbf{Obs}(R_s)} t$  and  $\mathbf{head}(r\rho) \rightarrow^*_{\mathbf{Obs}(R_s)} t$ , and*
- *there is a term  $t$  such that  $\mathbf{tail}(\ell\rho) \rightarrow^*_{\mathbf{Obs}(R_s)} t$  and  $\mathbf{tail}(r\rho) \rightarrow^*_{\mathbf{Obs}(R_s)} t$ .*

*Proof.* Let  $f$  be the root of  $\ell$ . Define  $\rho'$  by  $\sigma\rho' = x\rho : \sigma\rho$  for every argument of the shape  $x : \sigma$  of  $f$  in  $\ell$ , and  $\rho'$  coincides with  $\rho$  on all other variables. Then  $\mathbf{head}(\ell\rho) = \ell'\rho'$  for some rule in  $\ell' \rightarrow r'$  in  $\mathbf{Obs}(R_s)$ . Now a common reduct  $t$  of  $r'\rho'$  and  $\mathbf{head}(r\rho)$  is obtained by applying the rule  $\mathbf{head}(x : \sigma) \rightarrow x$  zero or more times. This yields  $\mathbf{head}(\ell\rho) = \ell'\rho' \rightarrow^*_{\mathbf{Obs}(R_s)} r'\rho' \rightarrow^*_{\mathbf{Obs}(R_s)} t$  and  $\mathbf{head}(r\rho) \rightarrow^*_{\mathbf{Obs}(R_s)} t$ . The argument for  $\mathbf{tail}(\ell\rho)$  and  $\mathbf{tail}(r\rho)$  is similar.  $\square$

**Lemma 5.4.** *The model  $(S_1, ([f]_1)_{f \in \Sigma_s})$  satisfies the specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$ .*

*Proof.* We have to prove that  $[\ell\rho]_1(i) = [r\rho]_1(i)$  for every equation  $\ell = r$  in  $R_s$ , every ground substitution  $\rho$  and every  $i \in \mathbf{N}$ . By definition  $[\ell\rho]_1(i)$  is the unique normal form with respect to  $\mathbf{Obs}(R_s) \cup R_d$  of  $\mathbf{head}(\mathbf{tail}^i(\ell\rho))$ , and  $[r\rho]_1(i)$  is the similar normal form of  $\mathbf{head}(\mathbf{tail}^i(r\rho))$ . The terms  $\mathbf{head}(\mathbf{tail}^i(\ell\rho))$  and  $\mathbf{head}(\mathbf{tail}^i(r\rho))$  have a common  $\mathbf{Obs}(R_s)$ -reduct. For  $i = 0$  this follows from the first part of Lemma 5.3, for  $i > 0$  this follows from the second part of Lemma 5.3. As they have a common reduct, their unique normal forms  $[\ell\rho]_1(i)$  and  $[r\rho]_1(i)$  with respect to  $\mathbf{Obs}(R_s) \cup R_d$  are equal, which we had to prove.  $\square$

For concluding the proof of Theorem 5.1 we have to prove that  $(S_1, ([f]_1)_{f \in \Sigma_s})$  is the only model satisfying the specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  and  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . This follows from the following lemma.

**Lemma 5.5.** *Let  $(S, ([f])_{f \in \Sigma_s})$  be any model satisfying  $(\Sigma_d, \Sigma_s, R_d, R_s)$ , and  $t \in \mathbf{T}_s$ . Then  $[t] = [t]_1$ .*

*Proof.* By definition in the model for  $u \in D$  and  $s \in S$  we have

$$([\cdot](u, s))(0) = u, \quad ([\cdot](u, s))(i) = s(i-1) \text{ for } i > 0.$$

In the original stream specification the symbols  $\mathbf{head}, \mathbf{tail}$  do not occur, for these fresh symbols we now define functions  $[\mathbf{head}]$  and  $[\mathbf{tail}]$  on streams  $s$  by

$$[\mathbf{head}](s) = s(0), \quad ([\mathbf{tail}](s))(i) = s(i+1) \text{ for } i \geq 0.$$

If  $S \neq D^\omega$  then it is not clear whether  $[\mathbf{tail}](s) \in S$  for every  $s \in S$ . Therefore we extend  $S$  to  $D^\omega$  and define  $[f](\dots)$  to be any arbitrary value if at least one argument is in  $D^\omega \setminus S$ ; note that for the model satisfying the specification we only required  $[\ell\rho] = [r\rho]$  for ground substitutions to  $\mathbf{T}_s$  by which these junk values do not play a role.

Due to the definitions of  $[\cdot]$ ,  $[\mathbf{head}]$  and  $[\mathbf{tail}]$  this extended model satisfies the equations

$$\mathbf{E} = \begin{cases} \mathbf{head}(x : \sigma) & = x \\ \mathbf{tail}(x : \sigma) & = \sigma \\ \sigma & = \mathbf{head}(\sigma) : \mathbf{tail}(\sigma) \end{cases}$$

that is, for  $\rho$  mapping  $x$  to any term of sort  $d$  and  $\sigma$  to any term of sort  $s$  we have  $[\ell\rho] = [r\rho]$  for every  $\ell \rightarrow r \in \mathbf{E}$ . From the definition of  $\mathbf{Obs}(R_s)$  it is easily checked that any innermost step  $t \rightarrow_{\mathbf{Obs}(R_s)} t'$  on a ground term  $t$  is either an application of one of the first two rules of  $\mathbf{E}$ , or it is of the shape

$$t \rightarrow_{\mathbf{E}}^* \cdot \rightarrow_{R_s} \cdot \rightarrow_{\mathbf{E}}^* t'$$

where due to the innermost requirement the redex of the  $\rightarrow_{R_s}$  step does not contain the symbols  $\mathbf{head}$  or  $\mathbf{tail}$  so is in  $\mathbf{T}_s$ . Since the model is assumed to satisfy the specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$ , we conclude that  $[t] = [t']$  for every innermost ground step  $t \rightarrow_{\mathbf{Obs}(R_s)} t'$ .

For the lemma we have to prove that  $[t](i) = [t]_1(i)$  for every  $i \in \mathbf{N}$ . By definition  $[t]_1(i)$  is the normal form with respect to  $\mathbf{Obs}(R_s) \cup R_d$  of  $\mathbf{head}(\mathbf{tail}^i(t))$ . Now consider an innermost  $\mathbf{Obs}(R_s) \cup R_d$ -reduction of  $\mathbf{head}(\mathbf{tail}^i(t))$  to  $[t]_1(i)$ . By the above observation and the definitions of  $[\mathbf{head}]$  and  $[\mathbf{tail}]$  we conclude that

$$[t](i) = [\mathbf{head}(\mathbf{tail}^i(t))] = [[t]_1(i)] = [t]_1(i),$$

the last step since  $[t]_1(i) \in D$ . This concludes the proof, both of the lemma and Theorem 5.1.  $\square$

We conclude this section by an example of a well-defined proper stream specification that is not productive.

**Example 4.** Choose  $\Sigma_s = \{c, f, g\}$ ,  $\Sigma_d = \{0, 1\}$ ,  $R_d = \emptyset$ , and  $R_s$  consists of the following equations:

$$\begin{aligned} c &= 1 : c \\ f(x : \sigma) &= g(f(\sigma)) \\ g(x : \sigma) &= c. \end{aligned}$$

This is a valid proper stream specification for which  $\mathbf{Obs}(R_s)$  is terminating, as can be shown by AProVE [6] or TTT2 [10]. Hence by Theorem 5.1 it is well-defined. So the ground term  $f(c)$  has a unique interpretation: the stream only consisting of 1's. However,  $f(c)$  is not productive, as it only reduces to terms having  $f$  or  $g$  on top.

So the TRS  $R_s$  uniquely defines  $f(c)$ , but is not suitable to compute its interpretation.

## 6. IMPLEMENTATION

In <http://www.win.tue.nl/~hzantema/str.zip> we offer a prototype implementation automating proving well-definedness of boolean stream specifications by the approach we proposed. The main feature is to generate the observational variant for any given boolean stream specification. Being only a prototype, the focus is on testing simple examples as they occur in this paper. The default version runs under Windows with a graphical user interface and provides the following features:

- Boolean stream specifications can be entered, loaded, edited and stored. The format is the same as given here, with the only difference that for the operator ':' a prefix notation is chosen, in order to be consistent with the user defined symbols.
- By clicking a button the observational variant of the current stream specification is tried to be created. In doing so, all requirements of the definition of stream specification are checked. If they are not fulfilled, an appropriate error message is shown.
- If all requirements hold, then the resulting observational variant is shown on the screen by which it can be entered by cut and paste in a termination tool. Alternatively, it can be stored.
- Alternatively, the stream specification can be transformed to Circ format. This occurs in two variants:
  - a basic variant in which the Circ proof goal should be added manually, and
  - a version generating two copies of the specification and generating goals for these to be equivalent.

Again it is shown on the screen with cut and paste facility, or the result can be stored, both for entering the result in the tool Circ.

- A term can be entered, and an initial part of the stream represented by this term can be computed.
- For unary symbols the process of unfolding as described in Section 3 is supported.
- Several stream specifications, including the Fibonacci stream (the variant as we will present in Example 7), the Thue-Morse stream (Example 1), the paper folding stream (Example 5 below) and the Kolakoski stream (Example 9) are predefined. For all of these examples termination of the observational variant can be proved fully automatically both by AProVE [6] and TTT2 [10], proving well-definedness of the given stream specification.

Apart from this graphical Windows version there is also a command line version to be run under Linux. This provides the main facility, that is, generates the observational variant in term rewriting format in case the syntax is correct, and generates an appropriate error message otherwise.

None of the actions require substantial computation: for all features the result shows up instantaneously. On the other hand, proving termination of a resulting observational variant by a tool like AProVE or TTT2 may take some computation time, although never more than a few seconds for the given examples. This was one of the objectives of the project: the transformation itself should be simple and direct, while the real work to be done makes use of the power of current termination provers.

We conclude this section by an interesting stream specification that can be dealt with by our implementation. Just like in the introduction for **Fib**, and later in Section 8 we also show a turtle visualization. These and others are made by a few lines of code traversing a boolean array containing the first  $N$  elements of a stream. These first  $N$  elements are determined by executing outermost rewriting with respect to  $R_s$  starting in the constant representing the intended stream, until the first  $N$  elements have been computed.

**Example 5.** Start by a ribbon of paper. Fold it half lengthwise. Next fold the folded ribbon half lengthwise again, and repeat this a number of times, every time folding in the same direction. Now by unfolding the ribbon one sees a sequence of top-folds and valley-folds, and the question is what is the pattern in this sequence. A first observation is that this pattern is the first half of the pattern obtained when folding once more, so every such sequence is a proper prefix of the next sequence. As a consequence, we can take the limit,



obtaining a boolean stream  $P$ , called the *paper folding stream*, in which top folds and valley folds are represented by 0 and 1, respectively.

Imagine what happens if we do an extra fold. Then all existing folds remain, but between every two consecutive folds a new fold is created. These new folds are alternately top folds and valley folds. So the effect of folding once more is that the new sequence is the **zip** of 010101... and the old sequence. Taking the limit we obtain

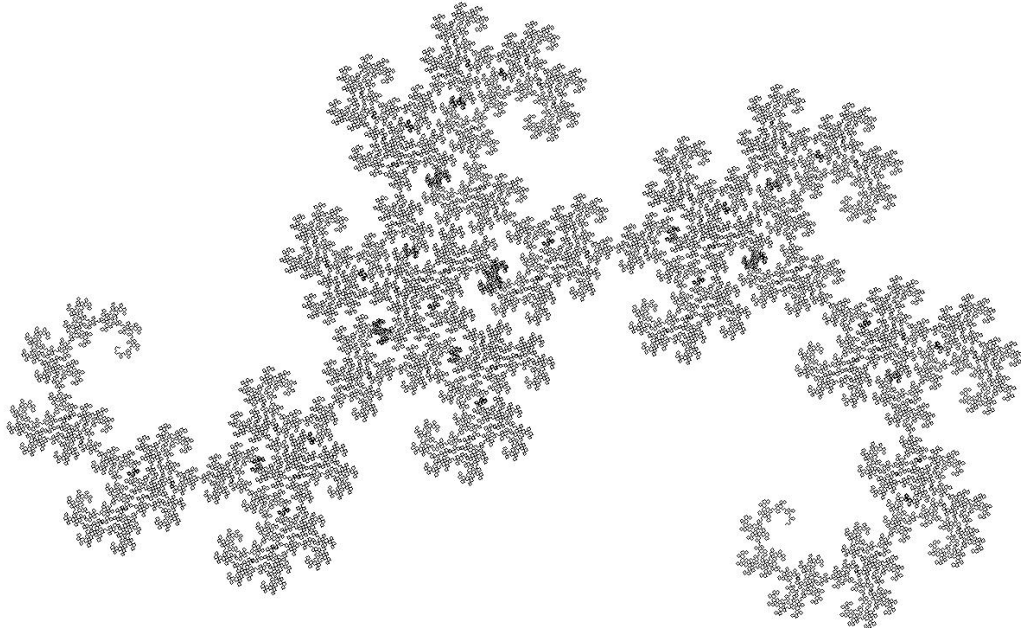
$$P = \mathbf{zip}(\mathbf{alt}, P),$$

where for **zip** and **alt** we have the equations

$$\mathbf{zip}(x : \sigma, \tau) = x : \mathbf{zip}(\tau, \sigma), \quad \mathbf{alt} = 0 : 1 : \mathbf{alt}.$$

One may wonder whether  $P$  is already fully defined by these three equations for  $P$ , **zip** and **alt**. It is, by Theorem 5.1, since the equations form a proper stream specification  $R_s$  for which termination of  $\mathbf{Obs}(R_s)$  is easily proved by TTT2 or AProVE.

Paper folding and many of its properties is folklore; we found this characterization of  $P$  independently. Turtle visualization of  $P$  is of particular interest, since the result is not just a visualization, but also the shape obtained if the ribbon is not fully unfolded, but only unfolded until the angles given as parameter of the turtle visualization. We only consider the case where the angles for 0 (top fold) and 1 (valley fold) are equal. In case this angle is 90 degrees, then the result is called the *dragon curve*; this curve touches itself, but does not intersect itself. Pictures are easily found on the Internet. When choosing turtle angles of less than 90 degrees, that is, the remaining paper fold is greater than 90 degrees, then the curve neither touches nor intersects itself. Doing this for 87 degrees and doing 15 folds, this yields the following turtle visualization of the first  $2^{15} - 1 = 32767$  elements of the stream  $P$ :



## 7. DATA INDEPENDENT STREAM FUNCTIONS

The reason that in Theorem 5.1 we have to restrict to models satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ , as we saw in Example 3, is in the fact that computations may be guarded by data elements in left-hand sides of equations. Next we show that we also get well-definedness for stream functions defined on all streams in case the left-hand sides of the equations do not contain data elements.

**Theorem 7.1.** *Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  be a proper stream specification for which the TRS  $\mathbf{Obs}(R_s) \cup R_d$  is terminating and the only subterms of left-hand sides of  $R_s$  of sort  $d$  are variables. Then the stream specification admits a unique model  $(S, ([f])_{f \in \Sigma_s})$  satisfying  $S = D^\omega$ .*

*Proof.* (sketch) We have to prove that for any  $f \in \Sigma_s$  of type  $d^n \times s^m \rightarrow s$  the function  $[f] : D^n \times (D^\omega)^m \rightarrow D^\omega$  is uniquely defined. For doing so we introduce  $m$  fresh constants  $c_1, \dots, c_m$  of sort  $s$ . Let  $k \in \mathbf{N}$  and  $u_1, \dots, u_n \in D$ . Due to termination and orthogonality of  $\mathbf{Obs}(R_s) \cup R_d$ , the term  $\mathbf{head}(\mathbf{tail}^k(f(u_1, \dots, u_n, c_1, \dots, c_m)))$  has a unique normal form with respect to  $\mathbf{Obs}(R_s) \cup R_d$ . Since it is of sort  $d$ , due to the shape of the rules it is a ground term of sort  $d$  over  $\Sigma_d \cup \{\mathbf{head}, \mathbf{tail}, c_1, \dots, c_m\}$ , that is, a ground term  $T$  composed from  $\Sigma_d$  and terms of the shape  $\mathbf{head}(\mathbf{tail}^i(c_j))$  for  $i \in \mathbf{N}$  and  $j \in \{1, \dots, m\}$ . For this observation it is essential that left-hand sides do not contain non-variable terms of sort  $d$ : terms of the shape  $f(\mathbf{head}(\dots), \dots)$  should be rewritten.

Let  $N$  be the greatest number  $i$  for which  $T$  has a subterm of the shape  $\mathbf{head}(\mathbf{tail}^i(c_j))$ . Let  $s_1, \dots, s_m \in D^\omega$ . Define  $t_j = s_j(0) : s_j(1) : \dots : s_j(N) : \sigma$ . Since the term  $\mathbf{head}(\mathbf{tail}^k(f(u_1, \dots, u_n, c_1, \dots, c_m)))$  rewrites to  $T$ ,  $\mathbf{head}(\mathbf{tail}^k(f(u_1, \dots, u_n, t_1, \dots, t_m)))$  rewrites to  $T'$  obtained from  $T$  by replacing every subterm of the shape  $\mathbf{head}(\mathbf{tail}^i(c_j))$  by  $\mathbf{head}(\mathbf{tail}^i(t_j))$ . Observe that  $\mathbf{head}(\mathbf{tail}^i(t_j))$  rewrites to  $s_j(i) \in D$ . So  $([f](u_1, \dots, u_n, s_1, \dots, s_m))(k)$  has to be the  $R_d$ -normal form of the ground term over  $\Sigma_d$  obtained from  $T$  by replacing every subterm of the shape  $\mathbf{head}(\mathbf{tail}^i(c_j))$  by  $s_j(i) \in D$ . Since this fixes  $([f](u_1, \dots, u_n, s_1, \dots, s_m))(k)$  for every  $k$ , this uniquely defines  $[f]$ .  $\square$

**Example 6.** It is easy to see that for the standard stream functions **zip**, **even** and **odd** defined by

$$\mathbf{even}(x : \sigma) = x : \mathbf{odd}(\sigma), \quad \mathbf{odd}(x : \sigma) = \mathbf{even}(\sigma), \quad \mathbf{zip}(x : \sigma, \tau) = x : \mathbf{zip}(\tau, \sigma),$$

there exists  $f : D^\omega \rightarrow D^\omega$  for every data set  $D$  satisfying

$$f(x : \sigma) = x : \mathbf{zip}(f(\mathbf{even}(\sigma)), f(\mathbf{odd}(\sigma))),$$

namely the identity. By Theorem 7.1 we can conclude it is the only one, since for  $R_d = \emptyset$  and  $R_s$  consisting of the above four equations, the resulting TRS  $\mathbf{Obs}(R_s)$  consisting of the rules

$$\begin{array}{ll} \mathbf{head}(\mathbf{even}(\sigma)) \rightarrow \mathbf{head}(\sigma) & \mathbf{head}(\mathbf{odd}(\sigma)) \rightarrow \mathbf{head}(\mathbf{even}(\mathbf{tail}(\sigma))) \\ \mathbf{tail}(\mathbf{even}(\sigma)) \rightarrow \mathbf{odd}(\mathbf{tail}(\sigma)) & \mathbf{tail}(\mathbf{odd}(\sigma)) \rightarrow \mathbf{tail}(\mathbf{even}(\mathbf{tail}(\sigma))) \\ \\ \mathbf{head}(f(\sigma)) \rightarrow \mathbf{head}(\sigma) & \\ \mathbf{tail}(f(\sigma)) \rightarrow \mathbf{zip}(f(\mathbf{even}(\mathbf{tail}(\sigma))), f(\mathbf{odd}(\mathbf{tail}(\sigma)))) & \end{array}$$

and the rules for  $'\cdot'$  and **zip** as in Example 2, is terminating as can be proved by AProVE [6] or TTT2 [10]. Other approaches seem to fail: the technique from [16] fails to prove that the identity is the only stream function satisfying the equation for  $f$ , while productivity of

stream specifications containing the rule for  $f$  cannot be proved to be productive by the technique from [4]. By essentially choosing  $\mathbf{Obs}(R_s)$  as the input and adding information about special contexts, the tool *Circ* [11] is able to prove that  $f$  is the identity.

## 8. MORE TRANSFORMATIONS PRESERVING SEMANTICS

Unfolding the Fibonacci stream specification as given in the introduction yields the proper stream specification  $R_s$  consisting of the equations

$$\begin{aligned} \mathbf{Fib} &= f(\mathbf{Fib}) & g(0, \sigma) &= 0 : 1 : f(\sigma) \\ f(x : \sigma) &= g(x, \sigma) & g(1, \sigma) &= 0 : f(\sigma). \end{aligned}$$

However, the TRS  $\mathbf{Obs}(R_s)$  is not terminating since it allows the infinite reduction

$$\mathbf{tail}(\mathbf{Fib}) \rightarrow \mathbf{tail}(f(\mathbf{Fib})) \rightarrow \mathbf{tail}(g(\mathbf{head}(\mathbf{Fib}), \underbrace{\mathbf{tail}(\mathbf{Fib})}_{\dots})) \rightarrow \dots,$$

so our method fails to prove well-definedness of  $\mathbf{Fib}$  in a direct way. In Lemma 3.2 and Lemma 3.3 we already saw two ways to modify stream specifications while preserving their semantics. In this section we will extend these lemmas to more general semantics preserving transformations, in particular by making use of the equations  $\mathbf{E}$  from the proof of Lemma 5.5 that hold in every model. As an example, we will apply such transformations to our original  $\mathbf{Fib}$  specification. The observational variant of the resulting stream specification will be terminating, so proving well-definedness of the transformed  $\mathbf{Fib}$  specification. But since the transformations are semantics preserving, this also proves well-definedness of the original  $\mathbf{Fib}$  specification.

In general we propose the following approach: in case for a stream specification the termination tools fail to prove termination of the observational variant, then try to apply semantics preserving transformations as discussed in Section 3 and this section until a transformed system has been found for which termination of the observational variant can be proved. If this succeeds, this not only proves well-definedness of the transformed specification, but also of the original one.

In this approach we have a symbol  $\mathbf{tail}$  in several variants of the specification, while in the construction of observational variant a fresh symbol  $\mathbf{tail}$  is required. So in the observational variant two versions of  $\mathbf{tail}$  occur: the original symbol  $\mathbf{tail}$  and the symbol  $\mathbf{tail}$  created by  $\mathbf{Obs}$ . However, if the observational variant happens to be terminating after identifying these two versions of  $\mathbf{tail}$ , then it is also terminating if they are distinguished, so identifying them will not yield wrong results. But it may happen that termination holds if the two versions of  $\mathbf{tail}$  are distinguished, and does not hold if they are identified. This is the case for Example 2.

Recall that mapping a stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  to  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$  with  $\Sigma_s \subseteq \Sigma'_s$  is said to *preserve semantics* if

- $(\Sigma_d, \Sigma_s, R_d, R_s)$  is well-defined if and only if  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$  is well-defined, and
- If  $(\Sigma_d, \Sigma_s, R_d, R_s)$  is well-defined with corresponding model  $(S, [\cdot])$ , and  $(S', [\cdot]')$  is the model corresponding to  $(\Sigma_d, \Sigma'_s, R_d, R'_s)$ , then  $[t] = [t]'$  for all ground terms of sort  $s$  over  $\Sigma_d \cup \Sigma_s$ .

For well-definedness we required the model to satisfy  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . For this section we need one more technical requirement:  $S$  should be closed under  $\mathbf{tail}$ . In order not to

change our definitions, throughout this section we assume the following extra assumptions to achieve this requirement:

- the symbol **tail** is in  $\Sigma_s$ , and
- the corresponding equation  $\mathbf{tail}(x : \sigma) = \sigma$  is in  $R_s$ .

Lemma 3.3 states that in keeping the same signature  $\Sigma_s$ , replacing  $R_s$  by  $R'_s$  preserves semantics as long as convertibility with respect to  $R_s$  coincides with convertibility with respect to  $R'_s$ . But as we are interested in preservation of semantics, this syntactical convertibility requirement may be weakened to a more semantic version: if  $R_s$  and  $R'_s$  do not have the same convertibility relation, but allow the same models, the same can be concluded. So now for a set  $R_s$  of equations we will introduce a congruence  $\sim_{R_s}$  being weaker than  $=_{R_s}$ , but still preserving semantics.

Recall the set  $\mathbf{E}$  of equations

$$\mathbf{E} = \begin{cases} \mathbf{head}(x : \sigma) = x \\ \mathbf{tail}(x : \sigma) = \sigma \\ \sigma = \mathbf{head}(\sigma) : \mathbf{tail}(\sigma) \end{cases}$$

For a set  $R_s$  of equations of sort  $s$  we define the relation  $\sim_{R_s}$  on terms over  $\Sigma_s \cup \Sigma_d \cup \{\mathbf{head}\}$  inductively by

- if  $\ell = r$  is in  $R_s$  then  $\ell \sim_{R_s} r$ ,
- $\sim_{R_s}$  is reflexive, symmetric and transitive,
- if  $C$  is a context and  $\rho$  is a substitution and  $t \sim_{R_s} t'$ , then  $C[t\rho] \sim_{R_s} C[t'\rho]$ ,
- if  $\ell = r$  is in  $\mathbf{E}$  then  $\ell \sim_{R_s} r$ ,
- if  $t, t'$  are terms that may contain a fresh variable  $x$  of type  $d$ , and  $t[x := u] \sim_{R_s} t'[x := u]$  for every  $u \in D$ , then  $t \sim_{R_s} t'$ .

Note that  $=_{R_s}$  is defined by the first three items, so  $\sim_{R_s}$  generalizes  $=_{R_s}$  by the additional last two items.

**Lemma 8.1.** *Let  $(\Sigma_d, \Sigma_s, R_d, R_s)$  and  $(\Sigma_d, \Sigma_s, R_d, R'_s)$  be stream specifications satisfying  $\ell \sim_{R'_s} r$  for all  $\ell = r$  in  $R_s$ , and  $\ell \sim_{R_s} r$  for all  $\ell = r$  in  $R'_s$ . Then transforming  $(\Sigma_d, \Sigma_s, R_d, R_s)$  to  $(\Sigma_d, \Sigma_s, R_d, R'_s)$  preserves semantics.*

*Proof.* In an arbitrary model  $(S, [\cdot])$  for a stream specification we define  $[\mathbf{head}](s) = s(0)$  for  $s \in S \subseteq D^\omega$ . By assuming the equation  $\mathbf{tail}(x : \sigma) = \sigma$  we conclude  $([\mathbf{tail}](s))(i) = s(i + 1)$  for  $i \geq 0$ . Combined with the definition of  $[\cdot]$  we conclude that  $\mathbf{E}$  holds in every model.

In case an equation  $t[x := u] = t'[x := u]$  holds in a model for every  $u \in D$ , then by definition the equation  $t = t'$  holds in the model, too.

Combining these observations we conclude by induction on the structure of  $\sim_{R_s}$  that if a model satisfies  $R_s$ , and  $t \sim_{R_s} t'$ , then the model satisfies the equation  $t = t'$  too. Applying this both for  $\sim_{R_s}$  and  $\sim_{R'_s}$ , and using the conditions of the lemma we conclude that a model satisfies  $(\Sigma_d, \Sigma_s, R_d, R_s)$  if and only if it satisfies  $(\Sigma_d, \Sigma_s, R_d, R'_s)$ . From this the lemma follows.  $\square$

**Example 7.** Our **Fib** specification completed by the **tail** equation reads

$$\begin{array}{ll} f(0 : \sigma) = 0 : 1 : f(\sigma) & \mathbf{Fib} = f(\mathbf{Fib}) \\ f(1 : \sigma) = 0 : f(\sigma) & \mathbf{tail}(x : \sigma) = \sigma. \end{array}$$

By Theorem 3.4 we know that unfolding this to

$$\begin{aligned} f(x : \sigma) &= g(x, \sigma) & \mathbf{Fib} &= f(\mathbf{Fib}), \\ g(0, \sigma) &= 0 : 1 : f(\sigma) & \mathbf{tail}(x : \sigma) &= \sigma \\ g(1, \sigma) &= 0 : f(\sigma) \end{aligned}$$

preserves semantics. Moreover, by Lemma 3.2 we may add a constant  $c$  to the signature and add the equation  $c = \mathbf{tail}(\mathbf{Fib})$ , still preserving semantics. Let  $R_s$  consist of these equations, and let  $R'_s$  consist of

$$\begin{aligned} f(x : \sigma) &= g(x, \sigma) & \mathbf{Fib} &= 0 : c \\ g(0, \sigma) &= 0 : 1 : f(\sigma) & c &= 1 : f(c) \\ g(1, \sigma) &= 0 : f(\sigma) & \mathbf{tail}(x : \sigma) &= \sigma. \end{aligned}$$

Now we will check the conditions of Lemma 8.1.

For proving that  $\ell \sim_{R'_s} r$  for all  $\ell = r$  in  $R_s$  we only need to consider the equations  $c = \mathbf{tail}(\mathbf{Fib})$  and  $\mathbf{Fib} = f(\mathbf{Fib})$ . We obtain

$$c =_{R'_s} \mathbf{tail}(0 : c) =_{R'_s} \mathbf{tail}(\mathbf{Fib})$$

and

$$\mathbf{Fib} =_{R'_s} 0 : c =_{R'_s} 0 : 1 : f(c) =_{R'_s} g(0, c) =_{R'_s} f(0 : c) =_{R'_s} f(\mathbf{Fib}).$$

For proving  $\ell \sim_{R_s} r$  for all  $\ell = r$  in  $R'_s$  we only need to consider the equations  $\mathbf{Fib} = 0 : c$  and  $c = 1 : f(c)$ . For this we need the congruence  $\sim_{R_s}$  rather than  $=_{R_s}$ . First observe

$$\mathbf{head}(g(0, \sigma)) \sim_{R_s} \mathbf{head}(0 : 1 : f(\sigma)) \sim_{R_s} 0,$$

and

$$\mathbf{head}(g(1, \sigma)) \sim_{R_s} \mathbf{head}(0 : f(\sigma)) \sim_{R_s} 0,$$

so by the last item of the definition of  $\sim_{R_s}$  we obtain  $\mathbf{head}(g(x, \sigma)) \sim_{R_s} 0$ . Using this we get

$$\begin{aligned} \mathbf{Fib} &\sim_{R_s} \mathbf{head}(\mathbf{Fib}) : \mathbf{tail}(\mathbf{Fib}) \\ &\sim_{R_s} \mathbf{head}(\mathbf{Fib}) : c \\ &\sim_{R_s} \mathbf{head}(f(\mathbf{Fib})) : c \\ &\sim_{R_s} \mathbf{head}(f(\mathbf{head}(\mathbf{Fib}) : \mathbf{tail}(\mathbf{Fib}))) : c \\ &\sim_{R_s} \mathbf{head}(g(\mathbf{head}(\mathbf{Fib}), \mathbf{tail}(\mathbf{Fib}))) : c \\ &\sim_{R_s} 0 : c. \end{aligned}$$

Using  $\mathbf{Fib} \sim_{R_s} 0 : c$ , for the remaining equation we have

$$\begin{aligned} c &\sim_{R_s} \mathbf{tail}(\mathbf{Fib}) \\ &\sim_{R_s} \mathbf{tail}(f(\mathbf{Fib})) \\ &\sim_{R_s} \mathbf{tail}(f(0 : c)) \\ &\sim_{R_s} \mathbf{tail}(g(0, c)) \\ &\sim_{R_s} \mathbf{tail}(0 : 1 : f(c)) \\ &\sim_{R_s} 1 : f(c). \end{aligned}$$

So the requirements of Lemma 8.1 are fulfilled and we conclude that transforming  $R_s$  to  $R'_s$  preserves semantics. By tools like AProVE or TTT2 one proves that  $\mathbf{Obs}(R'_s)$  is terminating, so by Theorem 5.1  $R'_s$  is well-defined. Due to preservation of semantics the same holds for  $R_s$ , and for the original  $\mathbf{Fib}$  specification.

As a consequence, we can conclude incompleteness of Theorem 5.1: the stream specification  $R_s$  is well-defined but  $\mathbf{Obs}(R_s)$  is not terminating, due to the infinite reduction of  $\mathbf{Obs}(R_s)$  we saw before.

The argument for the **Fib** example can be given in a more sloppy way as was done in [19] as follows. Identify ground terms with their interpretations in a model. The result of  $g$  always starts by 0, so we can write  $\mathbf{Fib} = f(\mathbf{Fib}) = g(\dots) = 0 : c$  for some stream  $c$ . Using this equality  $\mathbf{Fib} = 0 : c$  we obtain

$$0 : c = \mathbf{Fib} = f(\mathbf{Fib}) = f(0 : c) = 0 : 1 : f(c),$$

so  $c = 1 : f(c)$ . So any model for the original specification also satisfies  $R'_s$  which is obtained by replacing the equation  $\mathbf{Fib} = f(\mathbf{Fib})$  by the two equations  $\mathbf{Fib} = 0 : c$  and  $c = 1 : f(c)$ . As  $R'_s$  satisfies our format and  $\mathbf{Obs}(R'_s)$  is terminating we conclude well-definedness of  $\mathbf{Fib}$ .

For justifying the steps  $f(\mathbf{Fib}) = g(\dots) = 0 : c$  in this argument we need the last two items of the definition of  $\sim_{R_s}$ :

- for the step  $f(\mathbf{Fib}) = g(\dots)$  we need **E** to rewrite  $\mathbf{Fib}$  to a term with ":" on top, and
- for the step  $g(\dots) = 0 : c$  we need the case analysis on the data element in "... " as expressed by the last item in the definition of  $\sim_{R_s}$ ,

exactly as we did in our detailed proof. Note that the sloppy argument only shows that the new equations in  $R'_s$  are implied by original equations, and not the other way around. The following example shows that it is essential also to prove the other direction.

**Example 8.** Consider the proper stream specification  $R_s$  consisting of

$$\begin{array}{ll} f(x : \sigma, y : \tau) = g(x, y) & \mathbf{zeros} = 0 : \mathbf{zeros} \\ g(0, 0) = \mathbf{ones} & \mathbf{ones} = 1 : \mathbf{ones} \\ g(0, 1) = \mathbf{zeros} & c = f(c, c) \\ g(1, x) = \mathbf{zeros} & \mathbf{tail}(x : \sigma) = \sigma \end{array}$$

If  $[c]$  starts with 0, then  $[f(c, c)] = [g(0, 0)] = [\mathbf{ones}]$  starts with 1, and if  $[c]$  starts with 1, then  $[f(c, c)] = [g(1, 1)] = [\mathbf{zeros}]$  starts with 0, so  $R_s$  does not admit a model and is not well-defined. However, the proper stream specification  $R'_s$  obtained from  $R_s$  by replacing  $c = f(c, c)$  by  $c = f(f(c, c), c)$  is well-defined, while this new equation satisfies  $c =_{R_s} f(f(c, c), c)$ . Well-definedness of  $R'_s$  can be proved by proving termination of  $\mathbf{Obs}(R'_s)$ , where  $R''_s$  is obtained from  $R'_s$  by replacing the equation for  $c$  by  $c = \mathbf{zeros}$ . The transformation from  $R'_s$  to  $R''_s$  satisfies the requirements of Lemma 8.1; for checking this one shows that

$$f(f(0 : \sigma, 0 : \sigma), 0 : \sigma) \sim_{R'_s} f(g(0, 0), 0 : \sigma) \sim_{R'_s} f(1 : \mathbf{ones}, 0 : \sigma) \sim_{R'_s} g(1, 0) \sim_{R'_s} \mathbf{zeros}$$

and

$$f(f(1 : \sigma, 1 : \sigma), 1 : \sigma) \sim_{R'_s} f(g(1, 1), 1 : \sigma) \sim_{R'_s} f(0 : \mathbf{zeros}, 1 : \sigma) \sim_{R'_s} g(0, 1) \sim_{R'_s} \mathbf{zeros}$$

by which from the last item of the definition of  $\sim_{R'_s}$  one concludes

$$f(f(x : \sigma, x : \sigma), x : \sigma) \sim_{R'_s} \mathbf{zeros},$$

hence

$$c \sim_{R'_s} f(f(c, c), c) \sim_{R'_s} f(f(\mathbf{head}(c) : \mathbf{tail}(c), \mathbf{head}(c) : \mathbf{tail}(c)), \mathbf{head}(c) : \mathbf{tail}(c)) \sim_{R'_s} \mathbf{zeros}.$$

Next we show how we can use the combination of Lemmas 3.2 and 8.1 and Theorem 5.1 to prove that the following stream specification admits exactly *two* models  $(S, [\cdot])$  with  $S = \{[t] \mid t \in \mathbf{T}_s\}$ :

$$\begin{array}{ll} f(0 : \sigma) = 0 : 1 : f(\sigma) & m = f(m) \\ f(1 : \sigma) = 1 : 0 : f(\sigma) & \mathbf{tail}(x : \sigma) = \sigma. \end{array}$$

Assume we have a model  $(S, [\cdot])$  of this specification. Then either  $[m](0) = 0$  or  $[m](0) = 1$ . In the former case the equation  $m = 0 : \mathbf{tail}(m)$  holds, in the latter case the equation  $m = 1 : \mathbf{tail}(m)$  holds. First assume we are in the former case. Then we may add the equation  $m = 0 : \mathbf{tail}(m)$ . For this extended system we will prove well-definedness. Note that the specification is not orthogonal, but for applying Lemmas 3.2 and 8.1 and Theorem 3.4 this is not required. After applying Lemma 3.2 and Theorem 3.4 we arrive at the (non-proper) specification  $R_s$  consisting of

$$\begin{array}{ll} f(x : \sigma) = g(x, \sigma) & m = f(m) \\ g(0, \sigma) = 0 : 1 : f(\sigma) & m = 0 : \mathbf{tail}(m) \\ g(1, \sigma) = 1 : 0 : f(\sigma) & c = \mathbf{tail}(m) \\ & \mathbf{tail}(x : \sigma) = \sigma. \end{array}$$

Now we transform this to the proper specification  $R'_s$  consisting of

$$\begin{array}{ll} f(x : \sigma) = g(x, \sigma) & m = 0 : c \\ g(0, \sigma) = 0 : 1 : f(\sigma) & c = 1 : f(c) \\ g(1, \sigma) = 1 : 0 : f(\sigma) & \mathbf{tail}(x : \sigma) = \sigma. \end{array}$$

One easily checks that  $\ell =_{R'_s} r$  for all equations  $\ell = r$  in  $R_s$  and conversely, so by Lemma 8.1 (or even Lemma 3.3) one concludes that this transformation is semantics preserving. Since  $\mathbf{Obs}(R'_s)$  is easily checked to be terminating, this shows that adding  $m = 0 : \mathbf{tail}(m)$  to the original specification yields exactly one model with  $S = \{[t] \mid t \in \mathbf{T}_s\}$ . By symmetry the same holds for the other case, where the equation  $m = 1 : \mathbf{tail}(m)$  is added. Without a proof we mention that the two solutions for  $m$  are exactly the Thue-Morse stream **morse** from Example 1 and its inverse.

We conclude this section by an elaboration of the Kolakoski stream.

**Example 9.** The *Kolakoski stream* **Kol** is the unique fix point of  $g$  defined by

$$\begin{array}{ll} g(0 : \sigma) = 1 : 1 : f(\sigma) \\ g(1 : \sigma) = 1 : f(\sigma) \\ f(0 : \sigma) = 0 : 0 : g(\sigma) \\ f(1 : \sigma) = 0 : g(\sigma). \end{array}$$

So both for  $f$  and  $g$  its result on a stream is defined as follows. If a 1 is read, then a single symbol is produced, and if a 0 read, then two copies of a symbol are produced. This producing is done in such a way that the produced elements are alternately 0's and 1's, for  $f$  starting with 0 and for  $g$  starting with 1. Due to this procedure in some presentations instead of 0 the number 2 is written.

Of course we have to prove that  $g$  has a unique fix point **Kol**. Similar to what we saw for **Fib**, the fix point equation  $\mathbf{Kol} = g(\mathbf{Kol})$  causes non-termination in the observational variant so we cannot apply our approach directly. In order to prove well-definedness, we follow the same lines as we did for **Fib**, with the difference that now we do not start by unfolding, but postpone unfolding to the end. Start by the four equations for  $f$  and  $g$ , and the equations  $\mathbf{Kol} = g(\mathbf{Kol})$  and  $\mathbf{tail}(x : \sigma) = \sigma$ . According to Lemma 3.2 addition of

the equation  $K = \text{tail}(\text{tail}(\mathbf{Kol}))$  is semantics preserving. So let  $R_s$  consist of all of these equations for  $f, g, \mathbf{Kol}, \text{tail}, K$ . We will transform this to  $R'_s$  consisting of the equations for  $f, g, \text{tail}$ , and the two equations

$$\mathbf{Kol} = 1 : 0 : K, \quad K = 0 : g(K).$$

Applying unfolding (Theorem 3.4) to  $R'_s$  yields a proper stream specification for which TTT2 and AProVE succeed in proving termination of the observational variant, so by Lemma 8.1 it remains to show that  $\ell \sim_{R'_s} r$  for all  $\ell = r$  in  $R_s$ , and  $\ell \sim_{R_s} r$  for all  $\ell = r$  in  $R'_s$ . For doing so, first we show that  $\text{head}(g(0 : \sigma)) \sim_{R_s} 1$  and  $\text{head}(g(1 : \sigma)) \sim_{R_s} 1$ , so  $\text{head}(g(x : \sigma)) \sim_{R_s} 1$ , and hence  $\text{head}(g(\sigma)) \sim_{R_s} \text{head}(g(\text{head}(\sigma) : \text{tail}(\sigma))) \sim_{R_s} 1$ . Similarly we obtain  $\text{head}(f(\sigma)) \sim_{R_s} 0$ . Using this we derive

$$\mathbf{Kol} \sim_{R_s} \text{head}(\mathbf{Kol}) : \text{tail}(\mathbf{Kol}) \sim_{R_s} \text{head}(g(\mathbf{Kol})) : \text{tail}(\mathbf{Kol}) \sim_{R_s} 1 : \text{tail}(\mathbf{Kol}),$$

and

$$\begin{aligned} \text{head}(\text{tail}(\mathbf{Kol})) &\sim_{R_s} \text{head}(\text{tail}(g(\mathbf{Kol}))) \sim_{R_s} \text{head}(\text{tail}(g(1 : \text{tail}(\mathbf{Kol})))) \sim_{R_s} \\ &\text{head}(\text{tail}(1 : f(\text{tail}(\mathbf{Kol})))) \sim_{R_s} \text{head}(f(\text{tail}(\mathbf{Kol}))) \sim_{R_s} 0 \end{aligned}$$

from which  $\mathbf{Kol} \sim_{R_s} 1 : 0 : K$  follows. Moreover, we obtain

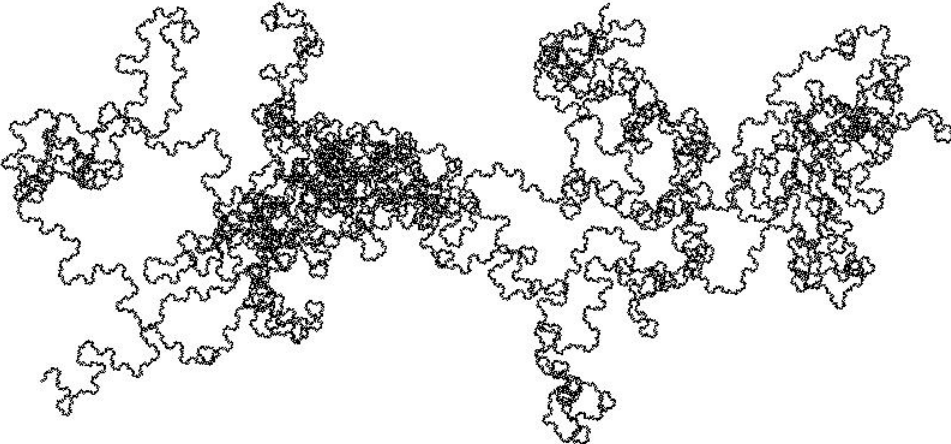
$$\begin{aligned} K &\sim_{R_s} \text{tail}(\text{tail}(\mathbf{Kol})) \\ &\sim_{R_s} \text{tail}(\text{tail}(g(\mathbf{Kol}))) \\ &\sim_{R_s} \text{tail}(\text{tail}(g(1 : 0 : K))) \\ &\sim_{R_s} \text{tail}(\text{tail}(1 : f(0 : K))) \\ &\sim_{R_s} \text{tail}(f(0 : K)) \\ &\sim_{R_s} \text{tail}(0 : 0 : g(K)) \\ &\sim_{R_s} 0 : g(K). \end{aligned}$$

For the other direction we have

$$g(\mathbf{Kol}) \sim_{R'_s} g(1 : 0 : K) \sim_{R'_s} 1 : f(0 : K) \sim_{R'_s} 1 : 0 : 0 : g(K) \sim_{R'_s} 1 : 0 : K \sim_{R'_s} \mathbf{Kol}$$

and  $K \sim_{R'_s} \text{tail}(\text{tail}(1 : 0 : K)) \sim_{R'_s} \text{tail}(\text{tail}(\mathbf{Kol}))$ , concluding the proof.

Although this stream  $\mathbf{Kol}$  has a very simple and regular definition, the stream seems to behave remarkably irregular. In contrast to earlier streams we saw, turtle visualizations of  $\mathbf{Kol}$  show up hardly any regular pattern: they seem to behave just like randomly generated boolean streams. For instance, by choosing the angle to be 90 degrees both for 0 and 1, taking the first 50000 elements of  $\mathbf{Kol}$  yields the following turtle visualization:





## 9. CONCLUSIONS AND FURTHER RESEARCH

We presented a technique by which well-definedness of stream specifications like Example 3 can be proved fully automatically, where a tool like Circ [12, 11] fails, and the productivity tool [4] fails to prove productivity. The main idea is to prove well-definedness by proving termination of a transformed system  $\mathbf{Obs}(R_s)$ , in this way exploiting the power of present termination provers.

We observed that productivity of the stream specification cannot be concluded from termination of  $\mathbf{Obs}(R_s)$ . Intuitively, productivity is closely related to termination; we leave as a challenge to further relate termination with productivity of stream specifications. A first step in this direction was made in [21]. There it was proved that productivity of a stream specification is equivalent to *balanced outermost termination* of the specification extended by an extra rule  $x : \sigma \rightarrow \mathbf{overflow}$ . Here an infinite reduction is called balanced outermost if only outermost redexes are reduced, and in the choice of them some fairness condition holds. As there are powerful techniques to prove outermost termination automatically [5], this can be used to prove productivity fully automatically. Unfortunately, as soon as binary operations like **zip** come in, typically the notions outermost termination and balanced outermost termination do not coincide: for many productive stream specifications the extension by the overflow rule is not outermost terminating, by which this approach fails. Instead in [20] some basic criteria for productivity have been investigated together with relationship with context-sensitive termination. Combined with a number of transformations and corresponding heuristics, this yields a powerful technique for proving productivity automatically, supported by an implementation. This approach exploits the power of present termination provers for proving productivity, just like we do in this paper for proving well-definedness.

We offer an implementation for computing  $\mathbf{Obs}(R_s)$  automatically, by which proving well-definedness can be done fully automatically in case the approach applies directly. For cases for which the approach does not apply directly, in Section 3 and Section 8 we developed techniques to transform stream specifications in such a way that semantics and well-definedness is preserved, and often our approach applies to the transformed specifications. Among these techniques only unfolding (Theorem 3.4) is supported by our implementation. For the other techniques some heuristics will be required. For the Fibonacci stream (Example 7) and the Kolakoski stream (Example 9) the following heuristics turned out to be successful:

- Identify a non-productive constant  $c$ . In both mentioned examples this is the stream to be defined, for which the equation is of the shape  $c = f(c)$ .
- Determine the first element  $d$  of the stream represented by  $c$ .
- Introduce a fresh constant  $c'$ , and introduce the equation  $c = d : c'$ .
- Using both the original equations and this new equation  $c = d : c'$  try to find a sound equation  $c' = t$  in which  $t$  is a term containing  $c'$ , but not  $c$ .
- Replace the original equation  $c = \dots$  by the two new equations  $c = d : c'$  and  $c' = t$ , and check whether this transformation is semantics preserving.
- In case this approach fails, try the generalization in which the first  $n$  elements  $d_1, \dots, d_n$  of the stream represented by  $c$  are determined for some small value  $n$ , and the equation  $c = d_1 : d_2 : \dots : d_n : c'$  is introduced for a fresh constant  $c'$ .

Another approach of using the techniques of Section 3 and Section 8 is proving well-definedness of a stream specification by proving productivity of all ground terms in a

transformed specification, e.g., by the approach of [20]. Since productivity implies well-definedness and the transformation preserves semantics, this implies well-definedness of the original specification.

In Section 8 we used the technical assumption that the model is closed under **tail**. This was forced by assuming the equation  $\mathbf{tail}(x : \sigma) = \sigma$ . We conjecture that for the validity of the approach this is not essential. More precisely, we conjecture that a stream specification  $(\Sigma_d, \Sigma_s, R_d, R_s)$  with  $\mathbf{tail} \notin \Sigma_s$  is well-defined if and only if the extended specification  $(\Sigma_d, \Sigma_s \cup \{\mathbf{tail}\}, R_d, R_s \cup \{\mathbf{tail}(x : \sigma) = \sigma\})$  is well-defined. This looks trivial as **tail** does not occur in the original specification, so is not expected to influence anything. However, giving a formal proof causes problems. The reason is that the model for  $(\Sigma_d, \Sigma_s, R_d, R_s)$  may not be closed under  $[\mathbf{tail}]$ . In fact we can even prove that in the model  $(S, [\cdot])$  for the **Fib** example satisfying  $S = \{[t] \mid t \in \mathbf{T}_s\}$ , the tail of **Fib** is *not* contained in  $S$ . A problem is how to lift  $[f]$  defined on  $S$  to the larger model that is closed under **tail**. For the particular **Fib** example a solution can be given, but for the general setting we failed.

This paper purely focuses on streams over a fixed data set  $D$ ; in all examples even  $D$  consists of the booleans. It is expected that the approach can be generalized to other infinite data types like infinite binary trees. A suitable format for this more general kind of infinite data structures has been given in [20]. In such a setting destructors can be defined as inverses of the constructors, just like in this paper we introduced  $(\mathbf{head}, \mathbf{tail})$  as the inverse of  $'\cdot'$ . Similar to what we did in this paper for streams, in this more general setting a specification consisting of equations on terms over constructors and user defined symbols will be transformed to an observational variant, being a rewrite system over destructors and the user defined symbols. Just like we did in this paper for the special case of streams, this rewrite system serves for observing data. It is orthogonal by construction, and well-definedness can be concluded from termination. Although the agenda for this approach for other infinite data structures is similar to what we did in this paper, this has not been elaborated in detail.

## REFERENCES

- [1] J.-P. Allouche and J. Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, 2003.
- [2] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
- [3] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [4] J. Endrullis, C. Grabmayer, and D. Hendriks. Data-oblivious stream productivity. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Computer Science*, pages 79–96. Springer, 2008. webinterface tool: <http://fspc282.few.vu.nl/productivity/>.
- [5] J. Endrullis and D. Hendriks. From outermost to context-sensitive rewriting. In R. Treinen, editor, *Proceedings of the 20th Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2009.
- [6] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, volume 4130 of *Lecture Notes in Computer Science*, pages 281–286. Springer, 2006. Tool available at <http://aprove.informatik.rwth-aachen.de/>.
- [7] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In Franz Baader and Andrei Voronkov, editors, *Proceedings*

- of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04), volume 3452 of *Lecture Notes in Artificial Intelligence*, pages 301–331, Montevideo, Uruguay, 2005. Springer.
- [8] J. Goguen, K. Lin, and G. Rosu. Circular coinductive rewriting. In *Proceedings, 15th International Conference on Automated Software Engineering (ASE'00)*. Institute of Electrical and Electronics Engineers Computer Society, 2000. Grenoble, France, 11-15 September 2000, webinterface tool CIRC: <http://fs1.cs.uiuc.edu/index.php/Special:CircOnline>.
- [9] R. Hinze. Functional pearl: streams and unique fixed points. In J. Hook and P. Thiemann, editors, *Proceeding of the 13th ACM SIGPLAN international conference on Functional programming, ICFP 2008*, pages 189–200. ACM, 2008.
- [10] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In R. Treinen, editor, *Proceedings of the 20th Conference on Rewriting Techniques and Applications (RTA)*, Lecture Notes in Computer Science. Springer, 2009. Tool available at <http://colo6-c703.uibk.ac.at/ttt2/>.
- [11] D. Lucanu, E.-I. Goriac, G. Caltais, and G. Rosu. CIRC: A behavioral verification tool based on circular coinduction. In *Proceedings of the 3rd Conference on Algebra and Coalgebra in Computer Science (CALCO'09)*, volume 5728 of *Lecture Notes in Computer Science*, pages 433–442. Springer, 2009.
- [12] D. Lucanu and G. Rosu. CIRC: A circular coinductive prover. In *CALCO'07*, volume 4624 of *Lecture Notes in Computer Science*, pages 372 – 378, 2007.
- [13] D. Lucanu and G. Rosu. Circular coinduction: A proof theoretical foundation. In *Proceedings of the 3rd Conference on Algebra and Coalgebra in Computer Science (CALCO'09)*, volume 5728 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 2009.
- [14] C. Marche and H. Zantema. The termination competition. In F. Baader, editor, *Proceedings of the 18th Conference on Rewriting Techniques and Applications (RTA)*, volume 4533 of *Lecture Notes in Computer Science*, pages 303–313. Springer, 2007.
- [15] G. Roşu. Equality of streams is a  $\Pi_2^0$ -complete problem. In *Proceedings of the 11th ACM SIGPLAN International Conference on Functional Programming (ICFP'06)*. ACM, 2006.
- [16] J.J.M.M. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15:93–147, 2005.
- [17] J. G. Simonsen. The  $\Pi_2^0$ -completeness of most of the properties of rewriting systems you care about (and productivity). In R. Treinen, editor, *Proceedings of the 20th Conference on Rewriting Techniques and Applications (RTA)*, Lecture Notes in Computer Science. Springer, 2009.
- [18] H. Zantema. A tool proving well-definedness of streams using termination tools. In *Proceedings of the 3rd Conference on Algebra and Coalgebra in Computer Science (CALCO'09)*, volume 5728 of *Lecture Notes in Computer Science*, pages 449–456. Springer, 2009.
- [19] H. Zantema. Well-definedness of streams by termination. In R. Treinen, editor, *Proceedings of the 20th Conference on Rewriting Techniques and Applications (RTA)*, volume 5595 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2009.
- [20] H. Zantema and M. Raffelsieper. Proving productivity in infinite data structures. In C. Lynch, editor, *Proceedings of the 21st Conference on Rewriting Techniques and Applications (RTA)*, Leibniz International Proceedings in Informatics. Schloss Dagstuhl Publishing, 2010.
- [21] H. Zantema and M. Raffelsieper. Stream productivity by outermost termination. In *Proceedings of the 9th International Workshop in Reduction Strategies in Rewriting and Programming (WRS 09)*, volume 15 of *Electronic Proceedings in Theoretical Computer Science*, 2010.