The following full text is a preprint version which may differ from the publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/84001

Please be advised that this information was generated on 2021-08-13 and may be subject to change.
Stream Productivity by Outermost Termination

Hans Zantema
Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
H. Zantema@tue.nl

Institute for Computing and Information Sciences, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

Matthias Raffelsieper
Department of Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
M. Raffelsieper@tue.nl

Streams are infinite sequences over a given data type. A stream specification is a set of equations intended to define a stream. A core property is productivity: unfolding the equations produces the intended stream in the limit. In this paper we show that productivity is equivalent to termination with respect to the balanced outermost strategy of a TRS obtained by adding an additional rule. For specifications not involving branching symbols balancedness is obtained for free, by which tools for proving outermost termination can be used to prove productivity fully automatically.

1 Introduction

Streams are among the simplest data types in which the objects are infinite: they can be seen as maps from the natural numbers to some data type $D$. 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 Thue Morse sequence $\text{morse}$ over the data elements 0, 1 can be specified by the rules

$\text{morse} \rightarrow 0 : \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse}))$
$\text{inv}(x : \sigma) \rightarrow \text{not}(x : \text{inv} (\sigma))$
$\text{tail}(x : \sigma) \rightarrow \sigma$
$\text{zip}(x : \sigma, \tau) \rightarrow x : \text{zip}(\tau, \sigma)$

together with the two rules $\text{not}(0) \rightarrow 1$ and $\text{not}(1) \rightarrow 0$.

This stream specification is productive: for every $n \in \mathbb{N}$ there is a rewrite sequence $\text{morse} \rightarrow^* u_1 : u_2 : \cdots : u_n : t$, that is, by these rules every $n$-th element of the stream can be computed. This notion of productivity goes back to Sijtsma [6]. In [2] a nice and powerful approach has been described to prove productivity automatically for a restricted class of stream specifications. Here we follow a completely different approach: we do not have these restrictions, but show that productivity is equivalent to termination with respect to a particular kind of outermost rewriting, after adding the rule $x : \sigma \rightarrow \text{overflow}$. The intuition of this equivalence is clear: productivity is equivalent to the claim that every ground term rewrites to a term with `: ' on top. This kind of rewriting is forced by doing outermost rewriting, and as soon as `: ' is on top, the reduction to $\text{overflow}$ is forced, blocking further rewriting.

However, there are some pitfalls. In the above example the term $\text{tail}(\text{morse})$ admits an infinite outermost reduction starting by

$\text{tail}(\text{morse}) \rightarrow \text{tail}(0 : \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse})))$
$\rightarrow \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse})))$
and then repeating this reduction forever on the created subterm \text{tail}(\text{morse}). So the outermost strategy to be considered needs an extra requirement disallowing this reduction. This requirement is what we call \textit{balanced}: we require every redex in the reduction either to be reduced eventually, or rewritten by a redex closer to the root. In the given example the redex \text{morse} in \text{zip(inv(\text{morse})}, \cdots) is never reduced, nor rewritten by a higher redex, so the resulting infinite outermost reduction is not balanced.

Our main result states that a stream specification given by a TRS \( R \) is productive for all ground terms if and only if \( R \cup \{ x : \sigma \rightarrow \text{overflow} \} \) does not admit an infinite balanced outermost reduction.

For the special case without rewrite rules for the data and without symbols having more than one argument of stream type, balancedness is obtained for free, and productivity of \( R \) on all ground terms is equivalent to outermost termination of \( R \cup \{ x : \sigma \rightarrow \text{overflow} \} \). For this fully automatic tools can be used, for instance based on the approaches of [3, 5, 7].

As an example consider
\[
\begin{align*}
    c &= 1 : c \\
    f(0 : \sigma) &= f(\sigma) \\
    f(1 : \sigma) &= 1 : f(\sigma)
\end{align*}
\]
by which we want to compute \( f(c) \). Clearly \( c \) only consists of ones, and \( f \) only removes zeros, so the result of \( f(c) \) will be the infinite stream of ones. Every 1 in this stream is easily produced by the reduction
\[
f(c) \rightarrow f(1 : c) \rightarrow 1 : f(c) \rightarrow \cdots,
\]
proving productivity of \( f(c) \). However, the approach from [2] fails, as this stream specification is not \textit{data-obliviously productive}, i.e., the identity of the data is essential for productivity. As far as we know, and confirmed by the authors of [2], until now there were no techniques for proving productivity automatically if the productivity is not data-oblivious. This has changed by the approach we present in this paper. The above example does not directly fit the basic format of our approach. However, it is easily (and automatically) unfolded to the system \( R \) consisting of the rules
\[
\begin{align*}
    c &= 1 : c \\
    f(x : \sigma) &= g(x, \sigma) \\
    g(0, \sigma) &= f(\sigma) \\
    g(1, \sigma) &= 1 : f(\sigma)
\end{align*}
\]
fitting the basic format of our approach. Now outermost termination of \( R \cup \{ x : \sigma \rightarrow \text{overflow} \} \) can be proved by a tool. Due to the shape of the symbols and the fact that there are no rewrite rules for the data, also balanced outermost termination of \( R \cup \{ x : \sigma \rightarrow \text{overflow} \} \) can be concluded. Then the main theorem of our paper states productivity, not only for \( f(c) \) but for all ground terms of sort stream.

The approach works for several other examples, for instance for an alternative definition of the morse stream.

In [9] a related approach is described, while an implementation of that technique is described in [8]. However, there the result is on well-definedness of stream specifications, which is a slightly weaker notion than productivity. The main result of [9] is that well-definedness of a stream specification can be concluded from termination of some transformed system: the observational variant.

### 2 The Main Result

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 rewrite system $R_d$ over $\Sigma_d$. Here all symbols of $\Sigma_d$ are of type $d^n \to d$ for some $n \geq 0$. In the actual stream specification we have a set $\Sigma_s$ of stream symbols, each being of type $d^n \times s^m \to s$ for $n, m \geq 0$. Apart from that, we assume a particular symbol $: \notin \Sigma_s$ having type $d \times s \to s$.

As a notational convention variables of sort $d$ will be denoted by $x,y$, terms of sort $d$ by $u_1,u_2$, variables of sort $s$ by $\sigma, \tau$, and terms of sort $s$ by $t_1,t_2$.

**Definition 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 rewrite rules over $\Sigma_d \cup \Sigma_s \cup \{:\}$ of the shape

$$f(u_1, \ldots, u_n, t_1, \ldots, t_m) \to t,$$

where

- $f \in \Sigma_s$ is of type $d^n \times s^m \to s$,
- for every $i = 1, \ldots, m$ 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 well-sorted term of sort $s$,
- $R_s \cup R_d$ is orthogonal,
- Every term of the shape $f(u_1, \ldots, u_n, u_{n+1} : t_1, \ldots, u_{n+m} : t_m)$ for $f \in \Sigma_s$ of type $d^n \times s^m \to s$, and $u_1, \ldots, u_{n+m} \in D$ matches with the left hand side of a rule from $R_s$.

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 $R_d = \emptyset$. Rules $\ell \to r$ in $R_s$ are often written as $\ell = r$.

Definition 1 is nearly the same as in [9]. It is closely related to the definition of stream specification in [2]: by introducing fresh symbols and rules for defining these fresh symbols, every stream specification in the format of [2] can be unfolded to a stream specification in our format. In the end of the introduction, where we unfolded $f(x : \sigma)$ to $g(x, \sigma)$, we already saw an example of this.

For defining productivity we follow the definition from [2]: a stream specification is called productive for a ground term $t$ if for every $n \in \mathbb{N}$ there exists a reduction of the shape $t \rightarrow u_1 : u_2 : \cdots : u_n : t'$. Instead of fixing the start ground term $t$ we prefer to require this for all ground terms of sort $s$. In practice this will make hardly any difference: typically a stream specification consists of an intended stream to be defined and a few auxiliary functions for which productivity not only holds for the single stream to be defined but also for any ground term built from it and the auxiliary functions.

Taking all ground terms of sort $s$ instead of only one has a strong advantage: then for proving productivity it is sufficient to prove that the first element is produced, rather than all elements. This is expressed in the following proposition that will serve as our characterization of productivity:

**Proposition 2.** A stream specification $(\Sigma_d, \Sigma_s, R_d, R_s)$ is productive for all ground terms of sort $s$ if and only if every ground term $t$ of sort $s$ admits a reduction $t \rightarrow R_d \cup R_s u' : t'$.

**Proof.** The “only if” direction of the proposition is obvious. To show the “if” direction, we show that if for all ground terms of sort $s$ we have $t \rightarrow R_d \cup R_s u_1 : u_2 : \cdots : u_n : t_n$ for all $n \in \mathbb{N}$. This is done by induction on $n$.

If $n = 0$, then the proposition directly holds.

Otherwise, we get from the induction hypothesis that $t \rightarrow R_d \cup R_s u_1 : u_2 : \cdots : u_{n-1} : t_{n-1}$ and $t_{n-1} \rightarrow R_d \cup R_s u_1 : u_2 : \cdots : u_{n-1} : u' : t'$ by assumption. Hence, $t \rightarrow R_d \cup R_s u_1 : u_2 : \cdots : u_{n-1} : u' : t'$, proving the proposition. □.
From now on we omit the subscript $R_s \cup R_d$ in rewrite steps $\rightarrow$. Given a term $t$, we define the set of positions $\text{Pos}(t) \subseteq \mathbb{N}^*$ as the smallest set such that $e \in \text{Pos}(t)$ and if $t = f(t_1, \ldots, t_n)$, then $i, p \in \text{Pos}(t)$ for all $1 \leq i \leq n$ and $p_i' \in \text{Pos}(t_i)$. The replacement of the subterm of $t$ at some position $p$, denoted $|_p t'$, by another term $t'$ is denoted $t[|_p t']$ and defined by $t[|_p t']_e = t'_e$ and $f(t_1, \ldots, t_n)[|_p t']_p = f(t_1[|_p t']_p, \ldots, t_n)$. A context $C$ is a special term, in which the variable $\square$ occurs exactly once. Then, we write $C[|_p t]$ to denote the term that is obtained by replacing $\square$ with the term $t$. If in a rewrite step $t \rightarrow t'$ the redex is on position $p \in \text{Pos}(t)$, we write $t \rightarrow_p t'$. We also write $t \rightarrow_p$ to indicate that the term $t$ has a redex at position $p$. For two positions $p, q$ we write $p \leq q$ if $p$ is a prefix of $q$, and $p < q$ if $p$ is a proper prefix of $q$, that is, the position $p$ is above $q$. If neither $p < q$ nor $q < p$, then we call the two positions independent, which is denoted $p \parallel q$. A rewrite step $t \rightarrow_p t'$ is called outermost if $t$ does not contain a redex in a position $q$ with $q < p$. A reduction is called outermost if every step is outermost. Such an infinite outermost reduction is called balanced outermost, if every redex is eventually either reduced or consumed by a redex at a higher position, as formally defined below.

**Definition 3.** Let $R$ be an arbitrary TRS. An infinite outermost reduction

$$l_1 \rightarrow_p l_2 \rightarrow_p l_3 \rightarrow_p \cdots$$

with respect to $R$ is called balanced outermost if for every $i$ and every redex of $t_i$ on position $q$ there exists $j \geq i$ such that $p_j \leq q$. The TRS $R$ is called balanced outermost terminating if it does not admit an infinite balanced outermost reduction.

A direct consequence is that for any infinite outermost reduction that is not balanced and contains a redex on position $p$ in some term, every term later in the reduction has a redex on position $p$, too.

As an example we consider the stream specification for the Thue Morse sequence from the introduction. The infinite reduction

$$\text{tail}(\text{morse}) \rightarrow \text{tail}(0 : \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse}))) \rightarrow \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse}))$$

continued by repeating this reduction forever on the created subterm $\text{tail}(\text{morse})$, is outermost, but not balanced, since the redex $\text{morse}$ on position 1.1 in the term $\text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse}))$ is never rewritten, and neither a higher redex. By forcing the infinite outermost reduction to be balanced, this redex should be rewritten, after which the rule for $\text{inv}$ can be applied, and has to be applied due to balancedness, after which the first argument of $\text{zip}$ will have $\cdot$ as its root, after which outermost reduction will choose the $\text{zip}$ rule and create a $\cdot$ as the root.

Now we arrive at the main theorem, showing that productivity of a stream specification is equivalent to balanced outermost termination of the stream specification extended with the rule $x : \sigma \rightarrow \text{overflow}$.

**Theorem 4.** A stream specification $(\Sigma_d, \Sigma_s, R_d, R_s)$ is productive for all ground terms of sort $s$ if and only if

$$R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}$$

is balanced outermost terminating.

### 3 Soundness

In this section we show soundness of Theorem 4, i.e., balanced outermost termination of the extended TRS implies productivity of the corresponding stream specification.

For doing so, using the special shape of stream specifications, first we prove a lemma stating that any ground term not having $\cdot$ as root symbol contains a redex that is not below a $\cdot$ symbol.
Lemma 5. Let \((\Sigma_d, \Sigma_s, R_d, R_s)\) be a stream specification, and let \(t\) be a ground term of sort \(s\) with \(\text{root}(t) \neq \cdot\). Then there exists a position \(p \in \text{Pos}(t)\) such that \(t \rightarrow p\) and for all \(p' < p\), \(\text{root}(t|_{p'}) \neq \cdot\).

Proof. This lemma is proven by structural induction on \(t\).

If \(t\) is a constant \(c \in \Sigma_c\), then by requirement there is a rule \(c \rightarrow r \in R_s\) for some term \(r\).

Otherwise, \(t = f(t_1, \ldots, t_m)\) for some symbol \(f \neq \cdot\), ground terms \(u_1, \ldots, u_n\) of sort \(d\), and ground terms \(t_1, \ldots, t_m\) of sort \(s\). If \(t \rightarrow \epsilon\), then the lemma holds. Therefore, we assume in the rest of the proof that this is not the case.

If there is a \(u_i\) such that \(u_i \rightarrow\) then this reduction is not below a \(\cdot\) since \(f \neq \cdot\). Otherwise, assume that \(u_i \in \text{NF}(R_d)\) for all \(1 \leq i \leq m\). If there is a \(i\) such that \(i\) is not below a \(\cdot\) then this reduction is not below a \(\cdot\) since \(f \neq \cdot\). Finally, we have to consider the case where \(u_i \in \text{NF}(R_d)\) and \(t_j = u_j : t_j'\) for all \(1 \leq j \leq n\) and some terms \(u_j, t_j'\). However, in this case it is required by stream specifications that \(t \rightarrow \epsilon\), giving a contradiction to our assumption. □

Using the above lemma, we can now prove soundness of our main result, i.e., we can show a stream specification \((\Sigma_d, \Sigma_s, R_d, R_s)\) to be productive by showing \(R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}\) to be balanced outermost terminating.

Using the above lemma, we can now prove soundness of our main result, i.e., we can show a stream specification \((\Sigma_d, \Sigma_s, R_d, R_s)\) to be productive by showing \(R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}\) to be balanced outermost terminating.

Proof of Soundness of Theorem 4. Assume \(t\) is not productive, i.e., it does not rewrite to a term with \(\cdot\) as its root symbol. This allows us to construct an infinite balanced outermost reduction w.r.t. \(R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}\): According to Lemma 5, there exists a position \(p\) such that \(t \rightarrow p\) and for all \(p' < p\), \(\text{root}(t|_{p'}) \neq \cdot\). Hence, there exists a position \(q_1 \leq p\) such that for some term \(t_1\), \(t \rightarrow q_1 t_1\) is an outermost step w.r.t. \(R_d \cup R_s\). Since also for all \(q' < q_1\), \(\text{root}(t|_{q'}) \neq \cdot\), this is also an outermost step w.r.t. \(R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}\). Also \(t_1\) is not productive, otherwise, if \(t_1\) would rewrite to a term with \(\cdot\) as its root symbol, then so would \(t\). Hence, we can repeat this argument to obtain an infinite outermost reduction \(t = t_0 \rightarrow q_1 t_1 \rightarrow q_2 t_2 \rightarrow \cdots\).

There might however be a term \(t_i\) and a redex on a position \(p \in \text{Pos}(t_i)\) that is never reduced or consumed in the constructed infinite outermost reduction. However, then there is never a reduction step above \(p\) in the remaining reduction, i.e., for all \(j > i\), \(q_j \leq p\). Since the reduction consists of outermost steps, we furthermore can conclude that \(q_j \neq p\), otherwise \(t_{j-1} \rightarrow q_j t_j\) would not be outermost. Hence, \(q_j \parallel p\) for all \(j > i\). Let \(p' \leq p\) such that \(t_i \rightarrow p'\) is an outermost step. Then also \(p' \parallel q_j\) for all \(j > i\), since \(q_j \leq p' \leq p\) would contradict the assumption that \(q_j \neq p\) and \(q_j > p'\) would contradict the assumption that \(t_{j-1} \rightarrow q_j t_j\) is an outermost step. Therefore, we can reduce the redex at position \(p'\) at any time, without affecting reducibility of the redexes at positions \(q_j\). These however might now become non-outermost steps. So let \(l_0 \rightarrow^* t_i \rightarrow q_i^{k+1} \cdots \rightarrow q_i l_k \rightarrow p' t_{k+1}^i\) for some \(k > i\) such that \(t_{k+1}^i \rightarrow q_{k+1}\) is not an outermost step. But then we can again apply the above reasoning that there is a redex on a position not below a \(\cdot\) symbol in \(t_{k+1}^i\) and following terms, yielding another infinite outermost reduction for which the redex of \(t_i\) at position \(p\) is reduced or consumed. Repeating this construction gives an infinite balanced outermost reduction, which shows soundness of the theorem. □

4 Completeness

In this section we show completeness of Theorem 4, i.e., disproving balanced outermost termination allows us to conclude non-productivity. Before we can prove this however, we first have to introduce some notation that allows us to distinguish between outermost and non-outermost rewrite steps.
**Definition 6.** For a TRS $R$, we define $t \xrightarrow{\beta} t'$ if $t \xrightarrow{\alpha} t'$ is an outermost rewrite step. Otherwise, if $t \xrightarrow{\alpha} t'$ is not an outermost rewrite step, we define $t \xrightarrow{\text{no}} t'$.

By convention, we will denote substitutions with $\xi, \rho$, which are mappings from variables to terms, written as $\{x_1 := t_1, \ldots, x_n := t_n\}$. Application of a substitution $\xi$ to a term $t$ is denoted $t[\xi]$. Given a TRS $R$, $c$ is called a **constructor** if $\text{root}(t) \neq c$ for all rules $t \rightarrow r \in R$. Furthermore, given a term $t$, the **tail** of a position $p \in \text{Pos}(t)$ w.r.t. another position $p' \in \text{Pos}(t)$ with $p' \leq p$ is denoted $p \setminus p'$. Thereby, $p \setminus p'$ is $p$ after removing the prefix $p'$. Finally, we define the concept of parallel rewrite steps.

**Definition 7.** For a TRS $R$ we define the **parallel rewrite step** $t \parallel t'$ if there exists a set of positions $\{p_1, \ldots, p_n\} \subseteq \text{Pos}(t)$ such that for all $1 \leq i, j \leq n$ with $i \neq j$, $p_i \parallel p_j$ and $t \rightarrow p_i t_i \rightarrow p_j t_j \cdots \rightarrow p_n t_n$.

A standard lemma that we will use is the **Parallel Moves Lemma**, which is for example presented and proved in [1, Lemma 6.4.4]. We will however use a slightly different form than presented there, but the proof of [1] easily shows this to be true.

**Parallel Moves Lemma.** Let $R$ be a TRS and $\ell \rightarrow r \in R$ a left-linear rule. If for two substitutions $\xi, \xi'$ we have that $x \xi = \xi' \xi'$ for all variables $x$, then $\ell \xi \rightarrow \ell' \xi' \rightarrow r \xi' \rightarrow r \xi'$.

It is easy to see that for an orthogonal TRS, the Parallel Moves Lemma is always applicable in case a term is reducible at two different positions. This holds, since there are no overlaps of the rules, i.e., any redex contained in another redex must be below some variable position, hence in the substitution part.

We will now show that a non-outermost reduction step followed by an outermost reduction step is either on an independent position or on a position below the outermost step.

**Lemma 8.** Let $R$ be an orthogonal TRS. If $t_1 \xrightarrow{\text{no}} t_2 \xrightarrow{\beta} t_3$, then $p \parallel q$ or $p < q$.

**Proof.** Let $t_1 \xrightarrow{\text{no}} t_1 \rightarrow t_2 \xrightarrow{\beta} t_2 \rightarrow t_3$. Therefore, a position $q < q$ exists such that $t_1 \rightarrow t_2 \parallel q$. Assume that $p \parallel q$ and $q \leq p$ (where the latter implies the former). Then $t_1 = t_1[\ell' \xi'[\ell_1 \xi_1][q \xi q]'q]$.

Since $R$ is orthogonal, there exists a variable $x$ and a context $C$ such that $t_1 = t_1[\ell' \xi''[x := C[\ell_1 \xi_1]][q \xi q]'q]$. Since $\xi''$ is like $\xi'$ except that $\xi''(x) = x$, therefore, $t_1 = t_1[\ell' \xi''[x := C[\ell_1 \xi_1]][q \xi q]'q] \xrightarrow{\text{no}} t_1[\ell' \xi''[x := C[\ell_1 \xi_1]][q \xi q]'q] = t_2$.

In this last term, the redex at position $p$ is contained, i.e., $t_2 = t_1[\ell' \xi''[x := C[\ell_1 \xi_1]][q \xi q]'q] = t_1[\ell' \xi''[x := C[\ell_1 \xi_1]][q \xi q]'q]$.

If $q \parallel p$ for all $q \in Q$, then we can swap the two reductions, i.e., $t_1 \xrightarrow{\beta} t_1 \rightarrow t_2 \rightarrow t_3$. Otherwise, a maximal $\theta \neq Q \subseteq Q$ exists such that $p < q$ for all $q \in Q$. Let $t_1 \xrightarrow{\text{no}} Q t_2 \xrightarrow{\beta} t_3$. Then, since $R$ is orthogonal, we can apply the Parallel Moves Lemma, showing that $t_1 \xrightarrow{\beta} t_2 \rightarrow t_3$ for some $t$. 

The above lemma allows us to show that for such a sequence of steps, i.e., a non-outermost step followed by an outermost step, we can swap the evaluation order and still reach the same term. In the remainder of this section we denote with $\parallel_{\text{no}}$ parallel non-outermost steps, i.e., a parallel reduction where all positions in the set $P$ are on non-outermost positions.

**Lemma 9.** For an orthogonal TRS $R$, if $t_1 \xrightarrow{\text{no}} t_2 \xrightarrow{\beta} t_3$, then $p \parallel q$ or $q > p$ for all $q \in Q$.

**Proof.** Let $t_1 \xrightarrow{\text{no}} Q t_2 \xrightarrow{\beta} t_3$. If $q \parallel p$ for all $q \in Q$, then we can swap the two reductions, i.e., $t_1 \xrightarrow{\beta} t_1 \rightarrow q t_3$.

If $t \xrightarrow{\text{no}} t_3$, then we have the required shape. Otherwise, if $t \xrightarrow{\beta} t_3$ then we also have the required shape, since $t_1 \rightarrow t_2 \parallel_{\text{no}} t_3$. Otherwise, a maximal $\theta \neq Q \subseteq Q$ exists such that $p < q$ for all $q \in Q$. Let $t_1 \xrightarrow{\text{no}} Q t_2 \xrightarrow{\beta} t_3$. Then, since $R$ is orthogonal, we can apply the Parallel Moves Lemma, showing that $t_1 \xrightarrow{\beta} t_2 \rightarrow t_3$ for some $t$. 


All redexes in the reduction \( t \xrightarrow{1} t' \) are on independent positions, hence we can first reduce all outermost ones, then all non-outermost ones. Therefore, a term \( t' \) exists such that \( t \xrightarrow{p} t \xrightarrow{*} t' \xrightarrow{q_0} t' \xrightarrow{Q} t_3 \) for some set \( Q \), where \( p < q' \) for all \( q'' \in Q' \). Because all positions in \( Q \setminus Q' \) are independent from the position \( p \), they are also independent from the positions in \( Q' \). Thus, we get that \( t \xrightarrow{Q} t' \xrightarrow{Q' \cup (Q, Q)} t_3 \). \( \square \)

Using the above lemma, we can prove that any reduction can be split into an outermost and a non-outermost reduction.

**Lemma 10.** Let \( R \) be an orthogonal TRS.

If \( t \xrightarrow{*} t' \), then \( t \xrightarrow{Q} t' \xrightarrow{Q} t' \) for some \( \bar{t} \).

**Proof.** Let \( t \xrightarrow{n} t' \). We perform induction on the length \( n \) of this reduction.

If \( n = 0 \), then \( t = t' \) and nothing has to be shown.

Otherwise, let \( t \xrightarrow{n-1} t_{n-1} \xrightarrow{} t' \). We get from the induction hypothesis that \( t \xrightarrow{Q} t' \xrightarrow{Q} t' \) for some \( \bar{t} \). If \( t_{n-1} \xrightarrow{Q} t' \) the then lemma holds. So assume \( t_{n-1} \xrightarrow{Q} t' \). Then \( \bar{t} \xrightarrow{Q} t' \) and therefore \( t' \xrightarrow{Q} t_{n-1} \xrightarrow{Q} t' \). Repeated application of Lemma 9 shows that for some \( \bar{t} \), \( \bar{t} \xrightarrow{Q} \bar{t} \xrightarrow{Q} \bar{t} \), hence \( t \xrightarrow{Q} t \xrightarrow{Q} t \xrightarrow{Q} t \) by unfolding the parallel non-outermost steps, which proves the lemma. \( \square \)

This allows us to show that for checking the productivity criterion of Proposition 2, we only have to consider outermost reductions.

**Lemma 11.** Let \( R \) be an orthogonal TRS having a binary symbol \( : \) in its signature.

If \( t \xrightarrow{*} u, t' \), then \( t \xrightarrow{Q} u : t' \xrightarrow{Q} u : t' \) for some terms \( u', t' \).

**Proof.** Let \( t \xrightarrow{*} u : t_u \). Then by Lemma 10, \( t \xrightarrow{Q} u : t_u \). If \( \root(\bar{t}) = : \), then the lemma holds. Otherwise, \( \root(\bar{t}) \neq : \). Let \( \bar{t} = t_0 \xrightarrow{q_0} t_1 \xrightarrow{q_1} \cdots xrightarrow{q_k} t_k = u : t_u \). Then for all \( 1 \leq i \leq k \), \( \root(t_i) = \root(t_{i-1}) \), since none of the terms can be reduced at the root position as this would be an outermost reduction step. This however gives a contradiction, because \( \neq \root(\bar{t}) = \root(t_0) = \root(t_1) = \cdots = \root(t_k) = \root(u : t_u) = : \). \( \square \)

Next, we prove two technical lemmas that will be used to prove completeness of our main theorem.

In the first we handle the case where a redex in a term that starts an infinite balanced outermost reduction is also reduced at that position later in the infinite balanced outermost reduction. In this case, we can bring forward this step and still get an infinite balanced outermost reduction.

**Lemma 12.** Let \( R \) be an orthogonal TRS for which \( : \) is a constructor.

If \( t_0 \xrightarrow{p_1} t_1 \xrightarrow{p_2} \cdots xrightarrow{p_j} t_j \) is an infinite balanced outermost reduction, where for all \( i \in \mathbb{N} \), \( \root(t_i) \neq : \), \( p_i \neq p_j \), and \( p_i \neq p_j \) for all \( 1 \leq i < j \), then an infinite balanced outermost reduction \( t_0 \xrightarrow{p_j} t'_1 \xrightarrow{p_j} \cdots \) exists, where for all \( i \in \mathbb{N} \), \( \root(t'_i) \neq : \).

**Proof.** First we show that \( p_i \neq p_j \) for all \( 1 \leq i < j \). For this, we perform induction on \( j-i \) and prove that if \( i < j \) and \( t_{i-1} \xrightarrow{} p_j \), then \( p_i \neq p_j \) \( t_i \rightarrow p_j \).

If \( j-i = 0 \), then \( i = j \) and the claim vacuously holds. Hence, we have \( t_{i-1} \xrightarrow{p_j} t_i \) and \( t_{i-1} \xrightarrow{} p_j \). If \( p_i \neq p_j \), we have a contradiction to the requirement \( p_i \neq p_j \), since \( i < j \). If \( p_i > p_j \), then we also have a contradiction, since then \( t_{i-1} \xrightarrow{p_j} \). Hence, \( p_i \neq p_j \) and therefore also \( i \rightarrow p_j \).

This shows that all positions \( p_j \) with \( 1 \leq j \) are on independent positions from \( p_j \), since \( t_0 \rightarrow p_j \) by assumption. Therefore, we can swap their order and get a reduction \( t_0 \xrightarrow{p_j} t'_1 \xrightarrow{p_j} t_2 \xrightarrow{p_j} \cdots xrightarrow{p_j} t_j \).

Due to Lemma 10, there exists a \( \hat{t}_1 \) such that \( t_0 \xrightarrow{p_j} t'_1 \xrightarrow{p_j} \hat{t}_1 \xrightarrow{Q} t_j \). Let \( t_0 = t_0 \xrightarrow{q_1} t_1 \xrightarrow{q_1} \cdots xrightarrow{q_k} t_k = \hat{t}_1 \xrightarrow{q_k} \cdots xrightarrow{q_k} t_j = t_j \), where \( q_k = p_j \). Furthermore, let \( q_{i+m} = p_{j+m} \) and \( t'_j = t_{j+m} \) for all \( m \geq 1 \). We will now show that every redex in this reduction is eventually reduced or consumed by a higher redex.
Assume not, i.e., there exists $i \geq 0$ such that for some $q \in \text{pos}(t_i)$, $t'_i \rightarrow q$ and for all $m > i$, $q_m \not\leq q$, i.e., either $q_m > q$ or $q_m \parallel q$. We can conclude that $i < l$, since $t'_i = t_i$ and $t_j$ is part of the balanced outermost reduction $t_0 \rightarrow^{o}_{p_1} \cdots$ If $0 \leq i < k$, then $t'_i \rightarrow^{o}_{q_{i+1}} t'_{i+1} \rightarrow^{o}_{q_{i+2}} \cdots \rightarrow^{o}_{q_k} t'_k$. If $q_{i+1} > q$, then because of $t_i \rightarrow q$, we would have $t_i \not\rightarrow^{o}_{q_{i+1}}$; therefore this cannot occur. If $q_{i+1} \parallel q$, then we also have $t'_{i+1} \rightarrow q$. Applying this repeatedly shows that $t'_k \rightarrow q$, i.e., it suffices to investigate the case where $i \geq k$. In this case, we have $t'_i \rightarrow^{no}_{q_{i+1}} \cdots \rightarrow^{no}_{q_k} t'_k = t_j$. If $q_{i+1} || q$, then also $t'_{i+1} \rightarrow q$. Otherwise, if $q_{i+1} > q$, then due to the Parallel Moves Lemma, we also have $t'_{i+1} \rightarrow q$. Applying this repeatedly shows that $t'_k = t_j \rightarrow q$ and for all $m > i$ we have $q_m \not\leq q$. This however is a contradiction, since $t_j$ was contained in the initial balanced outermost reduction. This shows our claim.

Furthermore, any non-outermost step of the above reduction, i.e., any step $t'_i \rightarrow^{no}_{q_{i+1}} t'_j, q_{i+1}, t'_{i+1}$ for $k \leq i < l$, is below some position $p_{m}$ for $m > j$. To show this, let $t'_i = t'_j[p_{i+1} \rightarrow^{o}_{q_{i+1}}]$. Then a position $q'_j < q_{i+1}$ exists such that $t'_i = t'_j[q'_i[x := C[q_{i+1} \rightarrow^{o}_{q_{i+1}}]]] \rightarrow q$. For some $\ell \rightarrow r \in R$. Since $R$ is orthogonal, there must be a variable $x$ and a context $C$ such that $t'_j = t'_j[x := C[q_{i+1} \rightarrow^{o}_{q_{i+1}}]]$. Since $\ell \rightarrow r \in R$, there is a constructor of $\ell$, i.e., $\ell \rightarrow r \in R$. Repeating this argument, we see that for every reduced non-outermost redex, there is a still a redex above it in the term $t'_j$. However, for every such redex at some position $q'_j$, there is a position $p_m$ with $m > j$ such that $p_m \leq q'$ due to the initial balanced outermost reduction, showing our claim.

To the reduction $t_0 = t_0 \rightarrow^{o}_{p_1} t_1 \rightarrow^{o}_{p_2} \cdots$ we can now repeatedly apply Lemma 9 to get the outermost reduction $t_0 \rightarrow^{o}_{p_1} t_1 \rightarrow^{o}_{p_2} \cdots$ This is a balanced outermost reduction due to the above observations, since every redex in a reduction $t'_0 \rightarrow^{o}_{i} \cdots$ is eventually reduced or consumed and every redex in a reduction $t'_0 \rightarrow^{o}_{i} \cdots$ is below some position $p_m$ that is reduced later in the reduction.

Finally, we have to show that none of the terms in the constructed infinite balanced outermost reduction has a `::` symbol as its root. If this was not the case, there would be a term $t''$ with root($t''$) = and $t'' \rightarrow^{o}_{i} \cdots$ for some $m$. However, for every such term $t_m$, we have that $t_{m} \rightarrow^{o}_{i} \cdots$ is a constructor of $R$, we would have that $\vdash$ root($t''$) = root($t_m$) = root($t$) = , giving a contradiction and hence showing the desired property.

The second case we have to consider is that a redex in a term starting an infinite balanced outermost reduction is strictly below some reduction step. But also in this case, we will show that we can reduce the redex and still get an infinite balanced outermost reduction.

**Lemma 13.** Let $R$ be an orthogonal TRS for which $\vdash$ is a constructor, $t_0 \rightarrow^{o}_{p_1} t_1 \rightarrow^{o}_{p_2} \cdots$ be an infinite balanced outermost reduction with root($t_i$) $\not\leq$:\ for all $i \geq 0$, $t_0 \rightarrow^{o}_{\ell \rightarrow q_i} 0 [r_{q_i}]_q = t'_{i}$, and let $p_j \leq q_i$ be minimal, with $p_j < q_i$.

Then an infinite balanced outermost reduction $t_0[r_{q_i}]_q = t'_i \rightarrow^{o}_{p_1} t'_2 \rightarrow^{o}_{p_2} \cdots$ exists with root($t'_i$) $\not\leq$:\ for all $i \geq 1$.

**Proof.** Let $t_{j-1} = t_0[r_{q_1}]_p \cdots \rightarrow^{o}_{r_{j-1} \rightarrow q_{j-1}]}_p \rightarrow^{o}_{j-1} \rightarrow^{o}_{q_{j-1}]}_p$. Then for some variable $x \in V(r_j)$ and some context $C$ we have that $t_{j-1} = t_0[r_{q_1}]_p \cdots \rightarrow^{o}_{r_{j-1} \rightarrow q_{j-1}]}_p \rightarrow^{o}_{j-1} \rightarrow^{o}_{q_{j-1}]}_p$.

Then $j \rightarrow^{o}_{\ell \rightarrow q_j}$ for all $q \in Q_j$. Furthermore, we define for all $k > j$, $Q_k = \{ q_1^{k}, \ldots, q_{m}^{k} \} = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$. Then $j \rightarrow^{o}_{\ell \rightarrow q_j}$ for all $q \in Q_j$. Furthermore, we define for all $k > j$, $Q_k = \{ q_1^{k}, \ldots, q_{m}^{k} \} = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$. Furthermore, we define for all $k > j$, $Q_k = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$.

If $x \in V(r_j)$, then the lemma trivially holds.

Otherwise, let $p_{i} \not\in Pos(C)$ such that $C[p_{i}] = \emptyset$ and $Q_{i} = \{ q \in Pos(t_i) \mid q = p_j, p_{j}' \parallel p_{j} \}$ for all $q \parallel x = \{ q_1^{i}, \ldots, q_{m}^{i} \}$. Then $j \rightarrow^{o}_{\ell \rightarrow q_j}$ for all $q \in Q_j$. Furthermore, we define for all $k > j$, $Q_k = \{ q_1^{k}, \ldots, q_{m}^{k} \} = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$. Furthermore, we define for all $k > j$, $Q_k = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$. Furthermore, we define for all $k > j$, $Q_k = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$. Furthermore, we define for all $k > j$, $Q_k = \{ q, p_j \parallel q \}$ for all $p_j \parallel q$.
that are reduced are removed, and positions below a reduction of the infinite balanced outermost reduction are modified such that they reflect the position of the redex in the right-hand side. This can be done since the TRS is orthogonal, which especially implies that a contained redex cannot overlap with the left-hand side of a rule that is applied above it, therefore it has to be below a variable position in the left-hand side.

Hence, we have for all \( k > j \) either \( p_k \parallel q \) for all \( q \in Q_{k-1} \), \( p_k = q \) for some \( q \in Q_{k-1} \), or \( p_k < q \) for some \( q \in Q_{k-1} \) (\( p_k \) cannot be below some \( q \), since otherwise it would not be outermost). In the first case, the reduction \( t_{k-1} \to \ldots \to t_0 \to t_k \) is unaffected. In the second case, where \( p_k = q \), we can remove this reduction step. In the third and final case, where \( p_k < q \), this reduction is also still possible, since \( R \) is orthogonal and reductions inside another redex cannot destroy the outer redex. Hence, we can again apply the argument and get an infinite reduction \( t' = t_0 \to t_1 \to t_2 \to \ldots \), where the positions \( p_i \) are the positions \( p_i \) after removing reduction steps as described above. This reduction is balanced, but not necessarily outermost. However, we can repeatedly apply Lemma 9 to get an infinite outermost reduction, which will defer non-outermost steps forever. To see that this reduction is balanced, assume the contrary. Then, a term \( t' \) and a position \( q \in \text{Pos}(t') \) exist such that \( t' \to q \) and this redex is never reduced or consumed, and there exists \( h > a \) such that \( p_h \leq q \) since the non-outermost reduction was balanced. Since Lemma 9 only swaps non-outermost reductions to the end, it must be the case that all \( p_h \leq q \) are non-outermost. Then however an outermost position \( p_h < p_h \) exists, hence it is not deferred forever. This gives a contradiction, since this position is reduced eventually, consuming the redex at position \( q \).

Finally, we show that \( \text{root}(t_i) \neq : \) for all \( i \geq 1 \). Assume this not to be the case, i.e., there is a minimal \( t_i \) with \( \text{root}(t_i) = : \). Then \( p_i = e \) and for \( t_i \to t' \to u : t' \) it must be the case that \( \text{root}(t') = : \). However, since this step was also contained in the original infinite balanced outermost reduction, this would contradict the requirement that \( \text{root}(t_i) \neq : \). Furthermore, since \( : \) is a constructor, also reordering the reductions into an outermost reduction cannot introduce a term with \( : \) as root symbol, since otherwise this term could be reduced to a term \( t' \) with \( \text{root}(t') = : \), which we have shown to be false. This proves the lemma.

Using the above lemmas, we can finally prove completeness of our main theorem.

**Proof of Completeness of Theorem 4.** Assume \( R_d \cup R_s \cup \{ x : \sigma \to \text{overflow} \} \) is not balanced outermost terminating, but \( (\Sigma_d, \Sigma_s, R_d, R_s) \) is productive. Then a term \( t \) exists that allows an infinite balanced outermost reduction \( t = t_0 \to t_1 \to \ldots \) and there exists a reduction \( t \to^* u : t' \). Since the symbol \( \text{overflow} \) does not occur on any left-hand side of \( R_d \cup R_s \), we conclude that for all \( i \geq 0 \), \( \text{root}(t_i) \neq : \), since otherwise the rule \( x : \sigma \to \text{overflow} \) would be applicable and no further reductions would be possible.

We can also construct an infinite balanced outermost reduction w.r.t. \( R_d \cup R_s \) from the given one by removing all applications of the rule \( x : \sigma \to \text{overflow} \), since the symbol \( \text{overflow} \) does not occur on any left-hand side of \( R_d \cup R_s \). This might leave some redexes that previously were contained in a redex w.r.t. that rule. However, these redexes can only be on positions above which never a reduction step takes place, hence we can reduce them at any time. Thus, we have an infinite balanced outermost reduction 
\[ t = t_0 \to t_1 \to t_2 \to \ldots \] w.r.t. the orthogonal TRS \( R_d \cup R_s \) where for all \( i \geq 0 \), \( \text{root}(t_i) \neq : \).

By Lemma 11 we get that an outermost reduction 
\[ t \to_{p_1} t_1 \to_{p_2} t_2 \to_{p_3} \ldots \] w.r.t. the orthogonal TRS \( R_d \cup R_s \) where for all \( j \geq 0 \), \( \text{root}(t_i) \neq : \). Case distinction on the relation of \( p_j \) and \( q_1 \) is performed. If \( p_j = q_1 \) then we get from Lemma 12 an infinite balanced outermost reduction 
\[ t_0 \to_{p_1} t_1 \to_{p_2} t_2 \to_{p_3} \ldots \] Otherwise, if \( p_j < q_1 \), Lemma 13 gives us an infinite balanced outermost reduction 
\[ t_0 \to_{p_1} t_1 \to_{p_2} t_2 \to_{p_3} \ldots \]
In both cases, we furthermore have that \( \text{root}(t_i^0) \neq \) for all \( i > 0 \). Hence, by induction on \( n \) we get an infinite balanced outermost reduction \( u : t_u \xrightarrow{\alpha} t_1 \xrightarrow{\alpha} \ldots \) in which no term has as root symbol ".", which yields the desired contradiction and therefore completes the proof.

\[ \square \]

5 Using Outermost Termination Tools

As stated in the introduction, balancedness is obtained for free in case there are no rewrite rules for the data, i.e., \( R_d = \emptyset \), and there are no rules in \( R_s \) that have more than one argument of stream type \( s \). In this section we prove that claim, which allows us to apply automatic tools for proving outermost termination to show productivity of stream specifications.

**Proposition 14.** Let \( (\Sigma_d, \Sigma_s, R_d, R_s) \) be a stream specification with \( R_d = \emptyset \) and the type of all \( f \in \Sigma_s \) is of the form \( d^n \times s^m \rightarrow s \) for some \( n \in \mathbb{N} \), \( m \in \{0, 1\} \).

Then every infinite outermost reduction \( t_0 \xrightarrow{\alpha} t_1 \xrightarrow{\alpha} t_2 \xrightarrow{\alpha} \ldots \) is balanced.

**Proof.** We perform structural induction to show that for any reduction step \( t \xrightarrow{\alpha} t' \), we have that \( p \leq p' \) for all positions \( p' \in \text{Pos}(t) \) with \( t \rightarrow t' \).

If \( t = c \in \Sigma_s \), then by requirement of stream specifications we have that \( t \rightarrow c \), hence \( p = p' \). Since \( c \leq p' \) for all \( p' \in \text{Pos}(t) \), we have proven this case.

Otherwise, if \( t = f(u_1, \ldots, u_n) \) (i.e., there is no argument of stream type), then we again conclude that \( t \rightarrow c \). This is due to \( R_d = \emptyset \) and the requirements of stream specifications, note that no data operations are allowed with arguments of stream type. So we have also proven this case.

In the final case to consider, we have \( t = f(u_1, \ldots, u_n, t') \). If \( t \rightarrow c \), then again we must have that \( p = p' \) and hence have proven the case. Therefore, assume that \( p > p' \). Since \( u_1, \ldots, u_n \in D = \text{NF}(R_d) \), because \( R_d = \emptyset \), it must be the case that for all \( p' \in \text{Pos}(t) \) with \( t \rightarrow t' \), \( n + 1 \leq p' \), hence this especially holds for \( p \) as well. Therefore, we get from the induction hypothesis that for the reduction step \( t' \xrightarrow{\alpha} t'', p \leq p' \) for all positions \( p' \in \text{Pos}(t') \) with \( t' \rightarrow t'' \). Because \( t \xrightarrow{\alpha} t' \), \( p = (n + 1)(p \wedge n + 1) \), and for all \( p' \in \text{Pos}(t) \) with \( t \rightarrow t' \) we have \( p' = (n + 1)p' \), it also holds that \( p \leq p' \), proving this final case and therefore the proposition.

\[ \square \]

The specification of the Thue Morse sequence given in the introduction shows the necessity of requiring at most one argument to be of stream type. It was already observed that the infinite reduction

\[ \text{tail}(\text{morse}) \rightarrow \text{tail}(0 : \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse}))) \rightarrow \text{zip}(\text{inv}(\text{morse}), \text{tail}(\text{morse})) \rightarrow \ldots , \]

continued by repeatedly reducing the redex \( \text{tail}(\text{morse}) \), is outermost but not balanced. To show that also the requirement \( R_d = \emptyset \) is needed, we again give an example that allows to construct an infinite outermost reduction that is not balanced. Consider the stream specification

\[
\begin{align*}
t \text{ail}(x : \sigma) &= \sigma \\
c &= 0 : f(\text{not}(1), \text{tail}(c)) \\
f(0, \sigma) &= 1 : f(0, \sigma) \\
f(1, \sigma) &= 0 : f(1, \sigma)
\end{align*}
\]

together with the rules \( R_d = \{ \text{not}(0) \rightarrow 1, \text{not}(1) \rightarrow 0 \} \). This stream specification is productive, as can be checked with the productivity tool of [2]. However, there also exists an infinite outermost reduction, namely

\[ \text{tail}(c) \rightarrow \text{tail}(0 : f(\text{not}(1), \text{tail}(c))) \rightarrow f(\text{not}(1), \text{tail}(c)) \rightarrow \ldots , \]
which is continued by repeatedly reducing the redex tail(c). This redex is outermost, since both rules having the symbol f as root require either 0 or 1 as first argument. To apply one of these rules, the outermost redex not(1) would have to be reduced first, which shows that the above infinite outermost reduction is not balanced.

To also present an example that does satisfy the requirements of Proposition 14, we give an alternative definition of the Thue Morse stream presented in the introduction:

\[
morse = \begin{cases} 0 : c \\ c = 1 : f(c) \\ f(0 : \sigma) = 0 : 1 : f(\sigma) \\ f(1 : \sigma) = 1 : 0 : f(\sigma) \end{cases}
\]

This example does not fit our format of stream specifications, however unfolding it leads to a stream specification that still satisfies the requirements of Proposition 14. After adding the rule \( x : \sigma \rightarrow \text{overflow} \), we have to show outermost termination of the following TRS:

\[
morse \rightarrow 0 : c \\
c \rightarrow 1 : f(c) \\
f(x : \sigma) \rightarrow g(x, \sigma) \\
g(0, \sigma) \rightarrow 0 : 1 : f(\sigma) \\
g(1, \sigma) \rightarrow 1 : 0 : f(\sigma) \\
x : \sigma \rightarrow \text{overflow}
\]

Outermost termination of the above TRS can for instance be proven using the transformation of [5] and AProVE [4] as a termination prover, or using the approach presented in [3]. This allows to conclude that the above stream specification is productive.

The next example is interesting, since it is not friendly nesting, a condition required by [2] to be applicable. Essentially, a stream specification is friendly nesting if the right-hand sides of every nested symbol start with `\:``, which is clearly not the case for the second rule below.

\[
c = 1 : c \\
f(x : \sigma) = g(x, \sigma) \\
g(0, \sigma) = 1 : f(\sigma) \\
g(1, \sigma) = 0 : f(f(\sigma)))
\]

As it can be checked, the above example fits into the stream specification format considered in this paper and it satisfies the requirements of Proposition 14. After adding the rule \( x : \sigma \rightarrow \text{overflow} \), outermost termination can be proved automatically using the above techniques, which allows to conclude productivity of the example.

6 Conclusions

We have shown that productivity of a stream specification \((\Sigma_d, \Sigma_s, R_d, R_s)\) is equivalent to showing outermost balanced termination of \(R_d \cup R_s \cup \{x : \sigma \rightarrow \text{overflow}\}\). To the best of our knowledge, this is the first approach capable of proving productivity of stream specifications that are not data-obliviously productive. It turns out that soundness of this technique for proving productivity coincides with the easier direction of our equivalence: outermost termination of the extended TRS implies productivity.
Our format of stream specifications is more restrictive than the format of [2]. However, this is not an essential restriction as any stream specification in the latter format can be transformed into our format by introducing new rules, as illustrated in [9] and at the end of the introduction of this paper.

It seems that productivity has some relationship with top termination of the stream specification. However, these notions are not equivalent. For instance, consider the stream specification

\[
\begin{align*}
    c &= f(c) \\
    f(x: \sigma) &= c
\end{align*}
\]

One easily shows that this system is top terminating, but \(c\) is not productive. We do not see how proving top termination can help for proving productivity.

When restricting to stream specifications with \(R_d = \emptyset\) and where every left-hand side of \(R_s\) contains at most one argument of type \(s\), then balancedness is obtained for free and techniques for proving outermost termination can be used to show productivity. An immediate topic for future work is hence to devise techniques for proving balanced outermost termination, which would allow to show productivity of arbitrary stream specifications.

References

[1] F. Baader & T. Nipkow (1998): Term Rewriting and All That. Cambridge University Press, Cambridge, UK.

[2] J. Endrullis, C. Grabmayer & D. Hendriks (2008): Data-oblivious stream productivity. In: Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08), Lecture Notes in Computer Science 5330. Springer-Verlag, pp. 79–96. Available at http://dx.doi.org/10.1007/978-3-540-89439-1_6. Webinterface tool: http://infinity.few.vu.nl/productivity/.

[3] J. Endrullis & D. Hendriks (2009): From Outermost to Context-Sensitive Rewriting. In: Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09), Lecture Notes in Computer Science 5595. Springer-Verlag, pp. 305–319. Available at http://dx.doi.org/10.1007/978-3-642-02348-4_22.

[4] J. Giesl, P. Schneider-Kamp & R. Thiemann (2006): AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In: Proceedings of the 3rd International Joint Conference on Automatic Reasoning (IJCAR'06), Lecture Notes in Computer Science 4130. Springer-Verlag, pp. 281–286. Available at http://dx.doi.org/10.1007/11814771_24. Downloadable from http://aprove.informatik.rwth-aachen.de.

[5] M. Raffelsieper & H. Zantema (2009): A transformational approach to prove outermost termination automatically. In: Proceedings of the 8th International Workshop in Reduction Strategies in Rewriting and Programming (WRS’08), Electronic Notes in Theoretical Computer Science 237. Elsevier Science Publishers B. V (North-Holland), pp. 3–21. Available at http://dx.doi.org/10.1016/j.entcs.2009.08.032.

[6] B. A. Sijsma (1989): On the Productivity of Recursive List Definitions. ACM Transactions on Programming Languages and Systems 11(4), pp. 633–649. Available at http://dx.doi.org/10.1145/695558.69563.

[7] R. Thiemann (2009): From outermost termination to innermost termination. In: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM’09), Lecture Notes in Computer Science 5404. Springer-Verlag, pp. 533–545. Available at http://dx.doi.org/10.1007/978-3-540-95891-8_48.

[8] H. Zantema (2009): 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), Lecture Notes in Computer Science 5728. Springer-Verlag, pp. 449–456. Available at http://dx.doi.org/10.1007/978-3-642-03741-2_32.
[9] H. Zantema (2009): Well-definedness of Streams by Termination. In: Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09). Lecture Notes in Computer Science 5595. Springer-Verlag, pp. 164–178. Available at http://dx.doi.org/10.1007/978-3-642-02348-4_12.