Simple Stream Specifications with Undecidable Productivity*

Bartek Klin[1] and Beata Nachyła[2]

[1] University of Warsaw
[2] Institute of Computer Science, Polish Academy of Sciences

A stream specification (see [3] for a survey) is a term rewriting system over a signature that includes a unary operation $a : \_$, called a constructor, for each a from some alphabet A. A typical specification, for $A = \{\mathtt{0}, \mathtt{1}\}$, looks like:

$$\mathtt{z} \longrightarrow \mathtt{0} : \mathtt{z} \qquad \mathtt{o} \longrightarrow \mathtt{1} : \mathtt{o} \qquad \mathtt{zip}(x : \sigma, \tau) \longrightarrow x : \mathtt{zip}(\tau, \sigma) \tag{1}$$

A specification is *productive* (for a designated starting term) if it can be rewritten to a term of the form $a_1 : a_2 : \cdots : a_n : t$, for an arbitrarily large number n. Specification (1) is productive for $\mathtt{zip}(\mathtt{z}, \mathtt{o})$. On the other hand, the specification

$$\mathtt{c} \longrightarrow \mathtt{0} : \mathtt{f}(\mathtt{c}) \qquad \mathtt{f}(x : y : \sigma) \longrightarrow x : \mathtt{f}(\sigma) \tag{2}$$

is not productive for $\mathtt{c}$.

Much effort (see references in [3, Sec. 7]) has been put into finding conditions on stream specifications that would ensure that they are productive, or at least that their productivity is decidable. For example, the format of stream equations from [7, Def. 4.3], which guarantees productivity, excludes the rightmost rule in (2), as it forbids (among other things) nested constructors in sources of rewrite rules. That format, translated to SOS specifications, was called *stream GSOS* in [4]. Another format studied there, *stream coGSOS*, when formulated in terms of stream specifications, forbids (among other things) nesting of non-constructor operations in targets of rewrite rules, so it excludes the leftmost rule in (2).

Both rules in (2) fit in the *pure stream format* of [1]. That format does not ensure productivity, but it does ensure its decidability. It requires (among other things) that rules are *data-oblivious*: if all constructors $a : \_$ were identified as a single constructor $\bullet : \_$, every operation could have at most one rewriting rule.

On the other hand, for specifications in the *lazy stream format* of [2] productivity is undecidable. These specifications are data-oblivious (indeed, they only allow a single constructor), but they contain head and tail operations specified by $\mathtt{head}(x : \sigma) \longrightarrow x$ and $\mathtt{tail}(x : \sigma) \longrightarrow \sigma$, disallowed in all other formats mentioned above. Undecidability is proved by a reduction from the halting problem of programs in Conway's Fractran language.

Here we show another class of simple stream specifications with undecidable productivity. Undecidability follows by a reduction from the halting problem of queue machines. We have used essentially the same reduction to prove undecidability of whether a given SOS specification induces a distributive law of a monad over a comonad [6], or whether it has a supported or stable model [5].

* This work was supported by the Polish National Science Centre (NCN) grant 2012/07/E/ST6/03026.

A queue machine with states Q and alphabet A is similar to a pushdown automaton, but it maintains a queue rather than a stack of symbols. Its configuration is a state $q \in Q$ together with a word (queue) $w \in A^*$. We consider machines that, in each step, depending on their state and on symbols at the front of the queue, remove zero, one or two elements from the front, and add exactly one element at the end of the queue. Such a machine never makes the queue empty, and it halts when it attempts to remove two symbols from the queue with only one symbol in it. For a precise definition, see [5]. It is undecidable whether a given machine halts from its initial configuration built of a state q_1 and a single symbol a_1 in the queue.

A queue machine $\mathcal{M}$ gives rise to a stream specification $S_{\mathcal{M}}$ over a signature built of a single constant $\mathtt{S}$, a unary operation $\mathtt{q}$ for each $q \in Q$ and a constructor $\mathtt{a} : \_$ for each $a \in A$, with rules:

$$\mathtt{S} \longrightarrow \mathtt{a}_1 : \mathtt{q}_1(\mathtt{S}) \ (\mathbf{S}) \qquad \mathtt{q}(\mathtt{a} : y) \longrightarrow \mathtt{c} : \mathtt{q}'(y) \ (\mathbf{R1})$$
$$\mathtt{q}(x) \longrightarrow \mathtt{c} : \mathtt{q}'(x) \quad (\mathbf{R0}) \qquad \mathtt{q}(\mathtt{a} : \mathtt{b} : z) \longrightarrow \mathtt{c} : \mathtt{q}'(z) \ (\mathbf{R2})$$

Rule **R0** is included only if, in state q, $\mathcal{M}$ does not remove any symbol from the queue, adds c to it and moves to state q'. Rule **R1** is added if, in state q with a at the front of the queue, $\mathcal{M}$ removes a, adds c to the queue and moves to state q'. Rule **R2** is added if, in state q with ab at the front of the queue, $\mathcal{M}$ removes a and b, adds c to the queue and moves to state q'.

Theorem 1. $\mathcal{M}$ halts from the initial configuration iff $S_{\mathcal{M}}$ is not productive for $\mathtt{S}$.

Specifications $S_{\mathcal{M}}$ are neither in the format of [7] (rule **R2** is not), nor in stream coGSOS (rule **S** is not), nor in the format of [1] (rules **R1**-**R2** are not data-oblivious, because a machine choses its next state depending on symbols in the queue). As a source of undecidability, they are incomparable with [2].

References

1. Jörg Endrullis, Clemens Grabmayer, and Dimitri Hendriks. Data-oblivious stream productivity. In *Procs. of LPAR'08*, volume 5300 of *LNCS*, 2008.
2. Jörg Endrullis, Clemens Grabmayer, and Dimitri Hendriks. Complexity of Fractran and productivity. In *Procs. CADE-22*, volume 5663 of *LNCS*, pages 371–387, 2009.
3. Jörg Endrullis, Dimitri Hendriks, and Jan Willem Klop. Streams are forever. *Bulletin of the EATCS*, 109:70–106, 2013.
4. B. Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412(38):5043–5069, 2011.
5. Bartek Klin and Beata Nachyła. Some undecidable properties of SOS specifications. To appear. Available from `http://www.mimuw.edu.pl/~klin/papers/jlamp16.pdf`.
6. Bartek Klin and Beata Nachyła. Distributive laws and decidable properties of SOS specifications. In *Procs. of EXPRESS/SOS'14*, volume 160 of *EPTCS*, 2014.
7. Clemens Kupke, Milad Niqui, and Jan Rutten. Stream differential equations: concrete formats for coinductive definitions. 2011.