Data-Oblivious Stream Productivity

Jörg Endrullis[1], Clemens Grabmayer[2], and Dimitri Hendriks[1]

[1] Vrije Universiteit Amsterdam, Department of Computer Science
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
`joerg@few.vu.nl  diem@cs.vu.nl`
[2] Universiteit Utrecht, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, The Netherlands
`clemens@phil.uu.nl`

Abstract. We are concerned with demonstrating productivity of specifications of infinite streams of data, based on orthogonal rewrite rules. In general, this property is undecidable, but for restricted formats computable sufficient conditions can be obtained. The usual analysis, also adopted here, disregards the identity of data, thus leading to approaches that we call data-oblivious. We present a method that is provably optimal among all such data-oblivious approaches. This means that in order to improve on our algorithm one has to proceed in a data-aware fashion.[3]

1 Introduction

For programming with infinite structures, productivity is what termination is for programming with finite structures. Productivity captures the intuitive notion of unlimited progress, of 'working' programs producing defined values indefinitely. In functional languages, usage of infinite structures is common practice. For the correctness of programs dealing with such structures one must guarantee that every finite part of the infinite structure can be evaluated, that is, the specification of the infinite structure must be productive.

We investigate this notion for stream specifications, formalized as orthogonal term rewriting systems. Common to all previous approaches for recognizing productivity is a quantitative analysis that abstracts away from the concrete values of stream elements. We formalize this by a notion of 'data-oblivious' rewriting, and introduce the concept of data-oblivious productivity. Data-oblivious

Fig. 1: Map of stream specifications

(non-)productivity implies (non-)productivity, but neither of the converse implications holds. Fig. 1 shows a Venn diagram of stream specifications, highlighting the subset of 'data-obliviously recognizable' specifications where (non-)
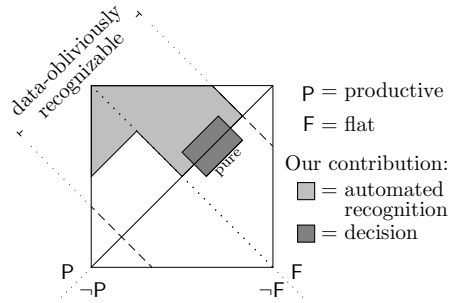
productivity can be recognized by a data-oblivious analysis.

We identify two syntactical classes of stream specifications: 'flat' and 'pure' specifications, see the description below. For the first we devise a decision algorithm for data-oblivious (d-o) productivity. This gives rise to a computable, d-o optimal, criterion for productivity: every flat stream specification that can be established to be productive by whatever d-o argument is recognized as productive by this criterion (see Fig. 1). For the subclass of pure specifications, we establish that d-o productivity coincides with productivity, and thereby obtain a decision algorithm for productivity of this class. Additionally, we extend our criterion beyond the class of flat stream specifications, allowing for 'friendly nesting' in the specification of stream functions; here d-o optimality is not preserved.

In defining the different formats of stream specifications, we distinguish between rules for stream constants, and rules for stream functions. Only the latter are subjected to syntactic restrictions. In flat stream specifications the defining rules for the stream functions do not have nesting of stream function symbols; however, in defining rules for stream constants nesting of stream function symbols *is* allowed. This format makes use of exhaustive pattern matching on data to define stream functions, allowing for multiple defining rules for an individual stream function symbol. Since the quantitative consumption/production behaviour of a symbol f might differ among its defining rules, in a d-o analysis one has to settle for the use of lower bounds when trying to recognize productivity. If for all stream function symbols f in a flat specification $\mathcal{T}$ the defining rules for f coincide, disregarding the identity of data-elements, then $\mathcal{T}$ is called pure.

Our decision algorithm for d-o productivity determines the tight d-o lower bound on the production behaviour of every stream function, and uses these bounds to calculate the d-o production of stream constants. We briefly explain both aspects. Consider the stream specification $A \to 0 : f(A)$ together with the rules $f(0 : \sigma) \to 1 : 0 : 1 : f(\sigma)$, and $f(1 : \sigma) \to 0 : f(\sigma)$, defining the stream $0 : 1 : 0 : 1 : \ldots$ of alternating bits. The tight d-o lower bound for f is the function $id : n \mapsto n$. Further note that $suc : n \mapsto n + 1$ captures the quantitative behaviour of the function prepending a data element to a stream term. Therefore the d-o production of A can be computed as $\mathrm{lfp}(suc \circ id) = \infty$, where $\mathrm{lfp}(f)$ is the least fixed point of $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ and $\overline{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$; hence A is productive. As a comparison, only a 'data-aware' approach is able to establish productivity of $B \to 0 : g(B)$ with $g(0 : \sigma) \to 1 : 0 : g(\sigma)$, and $g(1 : \sigma) \to g(\sigma)$. The d-o lower bound of g is $n \mapsto 0$, due to the latter rule. This makes it impossible for any conceivable d-o approach to recognize productivity of B.

We obtain the following results:

 (i) For the class of flat stream specifications we give a computable, d-o optimal, sufficient condition for productivity.
 (ii) We show decidability of productivity for the class of pure stream specifications, an extension of the format in [2].
(iii) Disregarding d-o optimality, we extend (i) to the bigger class of friendly nesting stream specifications.
(iv) A tool automating (i), (ii), and (iii), which can be downloaded from, and used via a web interface at: `http://infinity.few.vu.nl/productivity`.

Related work. Previous approaches [8, 3, 9, 1] employed d-o reasoning (without using this name for it) to find sufficient criteria ensuring productivity, but did not aim at optimality. The d-o production behaviour of a stream function f is bounded from below by a 'modulus of production' $\nu_f : \mathbb{N}^k \to \mathbb{N}$ with the property that the first $\nu_f(n_1, \ldots, n_k)$ elements of $f(t_1, \ldots, t_k)$ can be computed whenever the first n_i elements of t_i are defined. Sijtsma develops an approach allowing arbitrary production moduli $\nu : \mathbb{N}^k \to \mathbb{N}$, which, while providing an adequate mathematical description, are less amenable to automation. Telford and Turner [9] employ production moduli of the form $\nu(n) = n + a$ with $a \in \mathbb{Z}$. Hughes, Pareto and Sabry [3] use $\nu(n) = \max\{c \cdot x + d \mid x \in \mathbb{N}, \ n \geq a \cdot x + b\} \cup \{0\}$ with $a, b, c, d \in \mathbb{N}$. Both classes of production moduli are strictly contained in the class of 'periodically increasing' functions which we employ in our analysis. We show that the set of d-o lower bounds of flat stream function specifications is exactly the set of periodically increasing functions. Buchholz [1] presents a type system for productivity, using unrestricted production moduli. To obtain an automatable method for a restricted class of stream specifications, he devises a syntactic criterion to ensure productivity. This criterion easily handles all the examples of [9], but fails to deal with functions that have a negative effect like odd defined by $odd(x : y : \sigma) \to y : odd(\sigma)$ with a (periodically increasing) modulus $\nu_{odd}(n) = \lfloor \frac{n}{2} \rfloor$.

Overview. In Sec. 2 we define the notion of stream specification, and the syntactic format of flat and pure specifications. In Sec. 3 we formalize the notion of d-o rewriting. In Sec. 4 we introduce a production calculus as a means to compute the production of the data-abstracted stream specifications, based on the set of periodically increasing functions. A translation of stream specifications into production terms is defined in Sec. 5. Our main results, mentioned above, are collected in Sec. 6. We conclude and discuss some future research topics in Sec. 8.

2 Stream Specifications

We introduce the notion of stream specification. An example is given in Fig. 2, a productive specification of Pascal's triangle where the rows are separated by

$$P \to 0 : s(0) : f(P)$$
$$f(s(x) : y : \sigma) \to a(s(x), y) : f(y : \sigma) \qquad \textit{stream layer}$$
$$f(0 : \sigma) \to 0 : s(0) : f(\sigma)$$

$$a(s(x), y) \to s(a(x, y))$$
$$a(0, y) \to y \qquad \textit{data layer}$$

Fig. 2: Example of a flat stream specification.

zeros. Indeed, evaluating this specification, we get:

$$\mathsf{P} \twoheadrightarrow \underline{0}:\underline{1}:\underline{0}:\underline{1}:\underline{1}:\underline{0}:\underline{1}:\underline{2}:\underline{1}:\underline{0}:\underline{1}:\underline{3}:\underline{3}:\underline{1}:\ldots\ ,$$

where, for $n \in \mathbb{N}$, $\underline{n}$ is the *numeral* for n, defined by $\underline{n} := \mathsf{s}^n(\mathsf{0})$.

Stream specifications consist of a *stream layer* (top) where stream constants and functions are specified and a *data layer* (bottom).

The hierarchical setup of stream specifications is motivated as follows. In order to abstract from the termination problem when investigating productivity, the data layer is a term rewriting system on its own, and is required to be strongly normalizing. Moreover, an isolated data layer prevents the use of stream symbols by data rules. The stream layer may use symbols of the data layer, but not vice-versa. Stream dependent data functions, like $\mathsf{head}(x : \sigma) \to x$, might cause the output of undefined data terms. Consider the following examples from [8]:

$$\mathsf{S} \to 0:\mathsf{S}(6):\mathsf{S} \qquad\qquad \mathsf{T} \to 0:\mathsf{T}(7):\mathsf{T}\ ,$$

where $\sigma(n) := \mathsf{head}(\mathsf{tail}^n(\sigma))$; here S is well-defined, whereas T is not. Note that this is not a severe restriction, since stream dependent data functions usually can be replaced by pattern matching: $\mathsf{f}(\sigma, \tau) \to (\mathsf{head}(\sigma)+\mathsf{head}(\tau)):\mathsf{f}(\mathsf{tail}(\sigma), \mathsf{tail}(\tau))$, for example, can be replaced by the better readable $\mathsf{f}(x{:}\sigma, y{:}\tau) \to (x+y):\mathsf{f}(\sigma, \tau)$.

Stream specifications are formalized as many-sorted, orthogonal, constructor term rewriting systems [10]. We distinguish between *stream terms* and *data terms*. For the sake of simplicity we consider only one sort S for stream terms and one sort D for data terms. Without any complication, our results extend to stream specifications with multiple sorts for data terms and for stream terms.

Let U be a finite set of *sorts*. A *U-sorted set* A is a family of sets $\{A_u\}_{u \in U}$; for $V \subseteq U$ we define $A_V := \bigcup_{v \in V} A_v$. A *U-sorted signature* Σ is a U-sorted set of function symbols f, each equipped with an arity $ar(f) = \langle u_1 \cdots u_n, u \rangle \in U^* \times U$ where u is the sort of f; we write $u_1 \times \ldots \times u_n \to u$ for $\langle u_1 \cdots u_n, u \rangle$. Let X be a U-sorted set of variables. The *U-sorted set of terms* $Ter(\Sigma, X)$ is inductively defined by: for all $u \in U$, $X_u \subseteq Ter(\Sigma, X)_u$, and $f(t_1, \ldots, t_n) \in Ter(\Sigma, X)_u$ if $f \in \Sigma$, $ar(f) = u_1 \times \ldots \times u_n \to u$, and $t_i \in Ter(\Sigma, X)_{u_i}$. $Ter_\infty(\Sigma, X)$ denotes the set of *(possibly) infinite terms over Σ and X* (see [10]). Usually we keep the set of variables implicit and write $Ter(\Sigma)$ and $Ter_\infty(\Sigma)$. A *U-sorted term rewriting system (TRS)* is a pair $\langle \Sigma, R \rangle$ consisting of a U-sorted signature Σ and a U-sorted set R of *rules* that satisfy well-sortedness, for all $u \in U$: $R_u \subseteq Ter(\Sigma, X)_u \times Ter(\Sigma, X)_u$, as well as the standard TRS requirements.

Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a U-sorted TRS. For a term $t \in Ter(\Sigma)_u$ where $u \in U$ we denote the root symbol of t by $root(t)$. We say that two occurrences of symbols in a term are *nested* if the position [10, p.29] of one is a prefix of the position of the other. We define $\mathcal{D}(\Sigma) := \{root(l) \mid l \to r \in R\}$, the set of *defined symbols*, and $\mathcal{C}(\Sigma) := \Sigma \setminus \mathcal{D}(\Sigma)$, the set of *constructor symbols*. Then $\mathcal{T}$ is called a *constructor TRS* if for every rewrite rule $\rho \in R$, the left-hand side is of the form $f(t_1, \ldots, t_n)$ with $t_i \in Ter(\mathcal{C}(\Sigma))$; then ρ is called a *defining rule for f*. We call $\mathcal{T}$ *exhaustive* for $f \in \Sigma$ if every term $f(t_1, \ldots, t_n)$ with closed constructor terms t_i is a redex.

A *stream TRS* is a finite $\{S, D\}$-sorted, orthogonal, constructor TRS $\langle \Sigma, R \rangle$ such that ':' $\in \Sigma_S$, the *stream constructor symbol*, with arity $D \times S \to S$ is the single constructor symbol in Σ_S. Elements of Σ_D and Σ_S are called the *data symbols* and the *stream symbols*, respectively. We let $\Sigma_s := \Sigma_S \setminus \{':'\}$, and, for all $\mathsf{f} \in \Sigma_s$, we assume w.l.o.g. that the stream arguments are in front: $ar(\mathsf{f}) \in S^{ar_s(\mathsf{f})} \times D^{ar_d(\mathsf{f})} \to S$, where $ar_s(\mathsf{f})$ and $ar_d(\mathsf{f}) \in \mathbb{N}$ are called the *stream arity* and the *data arity* of f, respectively. By Σ_{scon} we denote the set of symbols in Σ_s with stream arity 0, called the *stream constant symbols*, and $\Sigma_{sfun} := \Sigma_s \setminus \Sigma_{scon}$ the set of symbols in Σ_s with stream arity unequal to 0, called the *stream function symbols*. By R_{scon} we mean the defining rules for the symbols in Σ_{scon}.

We repeat that the restriction to a single data sort D is solely for keeping the presentation simple; all of the definitions and results in the sequel generalize to multiple data and stream sorts. For stream TRSs we assume non-emptyness of data sorts, that is, for every data sort there exists a finite, closed, contructor term of this sort. In case there is only one data sort D, then the requirement boils down to the existence of a nullary constructor symbol of this sort.

We come to the definition of stream specifications.

Definition 2.1. A *stream specification* $\mathcal{T}$ is a stream TRS $\mathcal{T} = \langle \Sigma, R \rangle$ such that the following conditions hold:

(i) There is a designated symbol $\mathsf{M}_0 \in \Sigma_{scon}$ with $ar_d(\mathsf{M}_0) = 0$, the *root of* $\mathcal{T}$.
(ii) $\langle \Sigma_D, R_D \rangle$ is a terminating, D-sorted TRS; R_D is called the *data-layer* of $\mathcal{T}$.
(iii) $\mathcal{T}$ is exhaustive (for all defined symbols in $\Sigma = \Sigma_S \uplus \Sigma_D$).

In the sequel we restrict to stream specifications in which all stream constants in Σ_{scon} are reachable from the root: $\mathsf{M} \in \Sigma_{scon}$ is *reachable* if there is a term s such that $\mathsf{M}_0 \to^* s$ and M occurs in s. Note that reachability of stream constants is decidable, and that unreachable symbols may be neglected for investigating (non-)productivity.

Note that Def. 2.1 indeed imposes a hierarchical setup; in particular, stream dependent data functions are excluded by item (ii). Exhaustivity for Σ_D together with strong normalization of R_D guarantees that closed data terms rewrite to constructor normal forms, a property known as sufficient completeness [4].

We are interested in productivity of recursive stream specifications that make use of a library of 'manageable' stream functions. By this we mean a class of stream functions defined by a syntactic format with the property that their d-o lower bounds are computable and contained in a set of production moduli that is effectively closed under composition, pointwise infimum and where least fixed points can be computed. As such a format we define the class of flat stream specifications (Def. 2.2) for which d-o lower bounds are precisely the set of 'periodically increasing' functions (see Sec. 4). Thus only the stream function rules are subject to syntactic restrictions. No condition other than well-sortedness is imposed on the defining rules of stream constant symbols.

In the sequel let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. We define the relation $\rightsquigarrow$ on rules in R_S: for all $\rho_1, \rho_2 \in R_S$, $\rho_1 \rightsquigarrow \rho_2$ (ρ_1 *depends on* ρ_2) holds if and only if ρ_2 is the defining rule of a stream function symbol on the right-hand

side of ρ_1. Furthermore, for a binary relation $\to\ \subseteq A \times A$ on a set A we define $(a \to) := \{b \in A \mid a \to b\}$ for all $a \in A$, and we denote by $\to^+$ and $\to^*$ the *transitive closure* and the *reflexive and transitive closure* of $\to$, respectively.

Definition 2.2. A rule $\rho \in R_S$ is called *nesting* if its right-hand side contains nested occurrences of stream symbols from Σ_s. We use R_{nest} to denote the subset of nesting rules of R and define $R_{\neg nest} := R_S \setminus R_{nest}$, the set of *non-nesting rules*.

A rule $\rho \in R_S$ is called *flat* if all rules in $(\rho \leadsto^*)$ are non-nesting. A symbol $f \in \Sigma_s$ is called *flat* if all defining rules of f are flat; the set of flat symbols is denoted Σ_{flat}. A stream specification $\mathcal{T}$ is called *flat* if $\Sigma_s \subseteq \Sigma_{flat} \cup \Sigma_{scon}$, that is, all symbols in Σ_s are either flat or stream constant symbols.

The specification given in Fig. 2 is an example of a flat specification. A second example is given in Fig. 3. This is a productive specification that defines the

$$
\begin{aligned}
\mathsf{Q} &\to \mathsf{a} : \mathsf{Q}' \\
\mathsf{Q}' &\to \mathsf{b} : \mathsf{c} : \mathsf{f}(\mathsf{Q}') \\
\mathsf{f}(\mathsf{a} : \sigma) &\to \mathsf{a} : \mathsf{b} : \mathsf{c} : \mathsf{f}(\sigma) \qquad \text{\textit{stream layer}} \\
\mathsf{f}(\mathsf{b} : \sigma) &\to \mathsf{a} : \mathsf{c} : \mathsf{f}(\sigma) \\
\mathsf{f}(\mathsf{c} : \sigma) &\to \mathsf{b} : \mathsf{f}(\sigma)
\end{aligned}
$$

data layer

Fig. 3: Example of a flat stream specification.

ternary Thue–Morse sequence, a square-free word over $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$; see, e.g., [7]. Indeed, evaluating this specification, we get: $\mathsf{Q} \twoheadrightarrow \mathsf{a}:\mathsf{b}:\mathsf{c}:\mathsf{a}:\mathsf{c}:\mathsf{b}:\mathsf{a}:\mathsf{b}:\mathsf{c}:\mathsf{b}:\mathsf{a}:\mathsf{c}:\dots$.

As the basis of d-o rewriting (see Def. 3.3) we define the data abstraction of terms as the results of replacing all data-subterms by the symbol $\bullet$.

Definition 2.3. Let $(\!|\Sigma|\!) := \{\bullet\} \uplus \Sigma_S$. For stream terms $s \in Ter(\Sigma)_S$, the *data abstraction* $(\!|s|\!) \in Ter((\!|\Sigma|\!))_S$ is defined by:

$$
\begin{aligned}
(\!|\sigma|\!) &= \sigma\,, \\
(\!|u : s|\!) &= \bullet : (\!|s|\!)\,, \\
(\!|\mathsf{f}(s_1, \ldots, s_n, u_1, \ldots, u_m)|\!) &= \mathsf{f}((\!|s_1|\!), \ldots, (\!|s_n|\!), \bullet, \ldots, \bullet)\,.
\end{aligned}
$$

Based on this definition of data abstracted terms, we define the class of pure stream specifications, an extension of the equally named class in [2].

Definition 2.4. A stream specification $\mathcal{T}$ is called *pure* if it is flat and if for every symbol $f \in \Sigma_s$ the data abstractions $(\!|\ell|\!) \to (\!|r|\!)$ of the defining rules $\ell \to r$ of f coincide (modulo renaming of variables).

$$
\begin{aligned}
\mathsf{Q} &\to \mathsf{diff}(\mathsf{M}) \\
\mathsf{M} &\to 0 : \mathsf{zip}(\mathsf{inv}(\mathsf{M}), \mathsf{tail}(\mathsf{M})) \\
\mathsf{zip}(x : \sigma, \tau) &\to x : \mathsf{zip}(\tau, \sigma) \\
\mathsf{inv}(x : \sigma) &\to \mathsf{i}(x) : \mathsf{inv}(\sigma) \\
\mathsf{tail}(x : \sigma) &\to \sigma \\
\mathsf{diff}(x : y : \sigma) &\to \mathsf{x}(x, y) : \mathsf{diff}(y : \sigma)
\end{aligned}
\qquad \textit{stream layer}
$$

$$
\begin{aligned}
\mathsf{i}(0) \to 1 & \qquad \mathsf{i}(1) \to 0 \\
\mathsf{x}(0,0) \to \mathsf{b} & \qquad \mathsf{x}(0,1) \to \mathsf{a} \\
\mathsf{x}(1,0) \to \mathsf{c} & \qquad \mathsf{x}(1,1) \to \mathsf{b}
\end{aligned}
\qquad \textit{data layer}
$$

Fig. 4: Example of a pure stream specification.

Fig. 4 shows an alternative specification of the ternary Thue–Morse sequence, this time constructed from the binary Thue–Morse sequence specified by M. This specification belongs to the subclass of pure specifications, which is easily inferred by the shape of the stream layer: for each symbol in $\Sigma_{sfun} = \{\mathsf{zip}, \mathsf{inv}, \mathsf{tail}, \mathsf{diff}\}$ there is precisely one defining rule.

Another example of a pure stream specification is given in Fig. 5. Both defin-

$$
\begin{aligned}
\mathsf{M} &\to 0 : \mathsf{M}' \\
\mathsf{M}' &\to 1 : \mathsf{h}(\mathsf{M}') \\
\mathsf{h}(0 : \sigma) &\to 0 : 1 : \mathsf{h}(\sigma) \\
\mathsf{h}(1 : \sigma) &\to 1 : 0 : \mathsf{h}(\sigma)
\end{aligned}
\qquad \textit{stream layer}
$$

$$
\textit{data layer}
$$

Fig. 5: Example of a pure stream specification.

ing rules for h consume one, and produce two stream elements, that is, their data abstractions coincide.

Def. 2.4 generalizes the specifications called 'pure' in [2] in four ways concerning the defining rules of stream functions: First, the requirement of right-linearity of stream variables is dropped, allowing for rules like $\mathsf{f}(\sigma) \to \mathsf{g}(\sigma, \sigma)$. Second, 'additional supply' to the stream arguments is allowed. For instance, in the defining rule for the function diff in Fig. 4, the variable y is 'supplied' to the recursive call of diff. Third, the use of non-productive stream functions is allowed now, relaxing an earlier requirement on stream function symbols to be 'weakly guarded'. Finally, defining rules for stream function symbols may use a restricted form of pattern matching as long as, for every stream function f, the

d-o consumption/production behaviour (see Sec. 3) of all defining rules for f is the same.

Next we define friendly nesting stream specifications, an extension of the class of flat stream specifications.

Definition 2.5. A rule $\rho \in R_S$ is called *friendly* if for all rules $\gamma \in (\rho \rightsquigarrow^*)$ we have: (1) γ consumes in each argument at most one stream element, and (2) it produces at least one. The set of *friendly nesting rules* R_{fnest} is the largest extension of the set of friendly rules by non-nesting rules from R_S that is closed under $\rightsquigarrow$. A symbol $f \in \Sigma_s$ is *friendly nesting* if all defining rules of f are friendly nesting. A stream specification $\mathcal{T}$ is called *friendly nesting* if $\Sigma_s \subseteq \Sigma_{fnest} \cup \Sigma_{scon}$, that is, all symbols in Σ_s are either friendly nesting or stream constant symbols.

An example of a friendly nesting stream specification is given in Fig. 6. The root

$$\begin{array}{rl}
\mathsf{nats} \rightarrow 0 : \times(\mathsf{ones}, \mathsf{ones}) & \\
\mathsf{ones} \rightarrow \mathsf{s}(0) : \mathsf{ones} & \\
\times(x : \sigma, y : \tau) \rightarrow \mathsf{m}(x, y) : \mathsf{add}(\mathsf{times}(\tau, x), \times(\sigma, y : \tau)) & \textit{stream layer} \\
\mathsf{times}(x : \sigma, y) \rightarrow \mathsf{m}(x, y) : \mathsf{times}(\sigma, y) & \\
\mathsf{add}(x : \sigma, y : \tau) \rightarrow \mathsf{a}(x, y) : \mathsf{add}(\sigma, \tau) &
\end{array}$$

$$\begin{array}{rl}
\mathsf{a}(0, y) \rightarrow y & \\
\mathsf{a}(\mathsf{s}(x), y) \rightarrow \mathsf{s}(\mathsf{a}(x, y)) & \\
\mathsf{m}(0, y) \rightarrow 0 & \textit{data layer} \\
\mathsf{m}(\mathsf{s}(x), y) \rightarrow \mathsf{a}(y, \mathsf{m}(x, y)) &
\end{array}$$

Fig. 6: Example of a friendly nesting stream specification.

of this specification, the constant nats, evaluates to the stream $\underline{0} : \underline{1} : \underline{2} : \ldots$. The stream layer specifies a parameterized stream function times, which multiplies every element of a stream with the parameter, and the binary stream functions add for componentwise addition, and $\times$, the *convolution product* of streams (see [6]), mathematically defined as an operation $\langle \sigma, \tau \rangle \mapsto \sigma \times \tau$:

$$(\sigma \times \tau)(i) = \sum_{j=0}^{i} \sigma(j) \cdot \tau(i - j) \qquad \text{(for all } i \in \mathbb{N}).$$

The rewrite rule for $\times$ is nesting, because its right-hand has nested stream function symbols, both times and $\times$ are nested within add. Because of the presence of a nesting rule in the stream layer, which is not a defining rule of a stream constant symbol, this stream specification is not flat. However, the defining rules for add, times, and $\times$ are friendly. For instance for the rule for $\times$ we check: in both arguments it consumes (at most) one stream element (x and y), it produces

(at least) one stream element $(\mathsf{m}(x,y))$, and the defining rules of the stream function symbols in the right-hand side (add, times, and $\times$) are again friendly. Thus the function stream layer consists of one friendly nesting rule, two flat (and friendly) rules, and one defining rule for a stream constant. Therefore this stream specification is friendly nesting.

Definition 2.6. Let $\mathcal{A} = \langle\, Ter(\Sigma)_S, \to \,\rangle$ be an abstract reduction system (ARS) on the set of terms over a stream TRS signature Σ. The *production function* $\Pi_{\mathcal{A}} : Ter(\Sigma)_S \to \overline{\mathbb{N}}$ of $\mathcal{A}$ is defined for all $s \in Ter(\Sigma)_S$ by:

$$\Pi_{\mathcal{A}}(s) := \sup \{\, n \in \mathbb{N} \mid s \to^*_{\mathcal{A}} u_1 : \ldots : u_n : t \,\}\,.$$

We call $\mathcal{A}$ *productive for a stream term s* if $\Pi_{\mathcal{A}}(s) = \infty$. A stream specification $\mathcal{T}$ is called *productive* if $\mathcal{T}$ is productive for its root M_0.

The following proposition justifies this definition of productivity by stating an easy consequence: the root of a productive stream specification can be evaluated continually in such a way that a uniquely determined stream in constructor normal form is obtained as the limit. This follows easily from the fact that a stream specification is an orthogonal TRS, and hence has a confluent rewrite relation that enjoys the property UN^∞ (uniqueness of infinite normal form) [5].

Proposition 2.7. *A stream specification is productive if and only if its root has an infinite constructor term of the form $u_1 : u_2 : u_3 : \ldots$ as its unique infinite normal form.*

3   Data-Oblivious Analysis

We formalize the notion of d-o rewriting and introduce the concept of d-o productivity. The idea is a quantitative reasoning where all knowledge about the concrete values of data elements during an evaluation sequence is ignored. For example, consider the following stream specification:

$$\mathsf{M} \to \mathsf{f}(0:1:\mathsf{M}) \quad (1)\ \mathsf{f}(0:x:\sigma) \to 0:1:\mathsf{f}(\sigma) \quad (2)\ \mathsf{f}(1:x:\sigma) \to x:\mathsf{f}(\sigma)$$

The specification of M is productive: $\mathsf{M} \to^2 \mathsf{0{:}1{:}f(M)} \to^3 \mathsf{0{:}1{:}0{:}1{:}f(f(M))} \to^* \ldots$. During the rewrite sequence (2) is never applied. Disregarding the identity of data, however, (2) becomes applicable and allows for the rewrite sequence:

$$\mathsf{M} \to \mathsf{f}(\bullet : \bullet : \mathsf{M}) \to^{(2)} \bullet : \mathsf{f}(\mathsf{M}) \to^* \bullet : \mathsf{f}(\bullet : \mathsf{f}(\bullet : \mathsf{f}(\ldots)))\,,$$

producing only one element. Hence M is not d-o productive.

D-o term rewriting can be thought of as a two-player game between a *rewrite player* $\mathcal{R}$ which performs the usual term rewriting, and an *opponent* $\mathcal{G}$ which before every rewrite step is allowed to arbitrarily exchange data elements for (sort-respecting) data terms in constructor normal form. The opponent can either handicap or support the rewrite player. Respectively, the d-o lower (upper)

bound on the production of a stream term s is the infimum (supremum) of the production of s with respect to all possible strategies for the opponent $\mathcal{G}$. Fig. 7 depicts d-o rewriting of the above stream specification M; by exchanging data elements, the opponent $\mathcal{G}$ enforces the application of (2).

Remark 3.1. At first glance it might appear natural to model the opponent using a function $\mathcal{G} : Ter(\Sigma_D) \to Ter(\Sigma_D)$ from data terms to data terms. However, such a per-element deterministic exchange strategy preserves equality of data elements. Then the following specification of W:

$$W \to g(Z, Z) \qquad\qquad Z \to 0 : Z$$
$$g(0 : \sigma, 0 : \tau) \to 0 : g(\sigma, \tau) \qquad\qquad g(0 : \sigma, 1 : \tau) \to g(\sigma, \tau)$$
$$g(1 : \sigma, 1 : \tau) \to 0 : g(\sigma, \tau) \qquad\qquad g(1 : \sigma, 0 : \tau) \to g(\sigma, \tau)$$

would be productive for every such $\mathcal{G}$, which is clearly not what one would expect of a d-o analysis.

The opponent can be modelled by an operation on stream terms: $Ter(\Sigma)_S \to Ter(\Sigma)_S$. However, it can be shown that it is sufficient to quantify over strategies for $\mathcal{G}$ for which $\mathcal{G}(s)$ is invariant under exchange of data elements in s for all terms s. Therefore we first abstract from the data elements in favour of symbols $\bullet$ and then define the opponent $\mathcal{G}$ on the abstracted terms, $\mathcal{G} : Ter((\!|\Sigma|\!))_S \to Ter(\Sigma)_S$.

Definition 3.2. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. A *data-exchange function on* $\mathcal{T}$ is a function $\mathcal{G} : Ter((\!|\Sigma|\!))_S \to Ter(\Sigma)_S$ such that $\forall r \in Ter((\!|\Sigma|\!))_S$ it holds: $(\!|\mathcal{G}(r)|\!) = r$, and $\mathcal{G}(r)$ is in data-constructor normal form.

Definition 3.3. We define the ARS $\mathcal{A}_{\mathcal{T},\mathcal{G}} \subseteq Ter((\!|\Sigma|\!))_S \times Ter((\!|\Sigma|\!))_S$ for every data-exchange function $\mathcal{G}$, as follows:

$$\mathcal{A}_{\mathcal{T},\mathcal{G}} := \{ s \to (\!|t|\!) \mid s \in Ter((\!|\Sigma|\!)), \ t \in Ter(\Sigma) \text{ with } \mathcal{G}(s) \to_{\mathcal{T}} t \} \ .$$

The *d-o production range* $\overline{do}_{\mathcal{T}}(s)$ of a data abstracted stream term $s \in Ter((\!|\Sigma|\!))_S$ is defined as follows:

$$\overline{do}_{\mathcal{T}}(s) := \{ \Pi_{\mathcal{A}_{\mathcal{T},\mathcal{G}}}(s) \mid \mathcal{G} \text{ a data-exchange function on } \mathcal{T} \} \ .$$

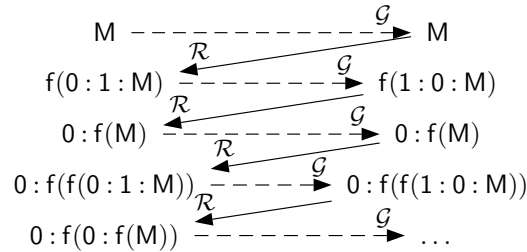Fig. 7: Data-oblivious rewriting

The *d-o lower* and *upper bound on the production of s* are defined by

$$\underline{do}_{\mathcal{T}}(s) := \inf(\overline{\underline{do}}_{\mathcal{T}}(s)), \qquad \text{and} \qquad \overline{do}_{\mathcal{T}}(s) := \sup(\overline{\underline{do}}_{\mathcal{T}}(s)),$$

respectively. These notions carry over to stream terms $s \in Ter(\Sigma)_S$ by taking their data abstraction $(\!|s|\!)$.

A stream specification $\mathcal{T}$ is *d-o productive* (*d-o non-productive*) if $\underline{do}_{\mathcal{T}}(\mathsf{M}_0) = \infty$ (if $\overline{do}_{\mathcal{T}}(\mathsf{M}_0) < \infty$) holds.

Proposition 3.4. *For $\mathcal{T} = \langle \Sigma, R \rangle$ a stream specification and $s \in Ter(\Sigma)_S$:*

$$\underline{do}_{\mathcal{T}}(s) \leq \Pi_{\mathcal{T}}(s) \leq \overline{do}_{\mathcal{T}}(s).$$

Hence d-o productivity implies productivity and d-o non-productivity implies non-productivity.

We define lower and upper bounds on the d-o consumption/production behaviour of stream functions. These bounds are used to reason about d-o (non-) productivity of stream constants, see Sec. 6.

Definition 3.5. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification, $\mathsf{g} \in \Sigma_s$, $k = ar_s(\mathsf{g})$, and $\ell = ar_d(\mathsf{g})$. The *d-o production range* $\overline{\underline{do}}_{\mathcal{T}}(\mathsf{g}) : \mathbb{N}^k \to \mathcal{P}(\overline{\mathbb{N}})$ *of* g is:

$$\overline{\underline{do}}_{\mathcal{T}}(\mathsf{g})(n_1, \ldots, n_k) := \overline{\underline{do}}_{\mathcal{T}}(\,\mathsf{g}((\bullet^{n_1} : \sigma), \ldots, (\bullet^{n_k} : \sigma), \underbrace{\bullet, \ldots, \bullet}_{\ell\text{-times}})\,),$$

where $\bullet^m : \sigma := \overbrace{\bullet : \ldots : \bullet}^{m \text{ times}} : \sigma$. The *d-o lower and upper bounds on the production of* g are defined by $\underline{do}_{\mathcal{T}}(\mathsf{g}) := \inf(\overline{\underline{do}}_{\mathcal{T}}(\mathsf{g}))$ and $\overline{do}_{\mathcal{T}}(\mathsf{g}) := \sup(\overline{\underline{do}}_{\mathcal{T}}(\mathsf{g}))$, respectively.

Even simple stream function specifications can exhibit a complex d-o behaviour. For example, consider the flat function specification:

$$\mathsf{f}(\sigma) \to \mathsf{g}(\sigma, \sigma)$$
$$\mathsf{g}(0 : y : \sigma, x : \tau) \to 0 : 0 : \mathsf{g}(\sigma, \tau)$$
$$\mathsf{g}(1 : \sigma, x_1 : x_2 : x_3 : x_4 : \tau) \to 0 : 0 : 0 : 0 : 0 : \mathsf{g}(\sigma, \tau)$$

Fig. 8 (left) shows a (small) selection of the possible function-call traces for f. In particular, it depicts the traces that contribute to the d-o lower bound $\underline{do}_{\mathcal{T}}(\mathsf{f})$. The lower bound $\underline{do}_{\mathcal{T}}(\mathsf{f})$, shown on the right, is a superposition of multiple traces of f. In general $\underline{do}_{\mathcal{T}}(\mathsf{f})$ can even be a superposition of infinitely many traces.

First Observations on Data-Oblivious Rewriting

For a stream function symbol g, we define its optimal production modulus ν_{g}, the data-aware, quantitative lower bound on the production of g, and compare it to its d-o lower bound $\underline{do}_{\mathcal{T}}(\mathsf{g})$.
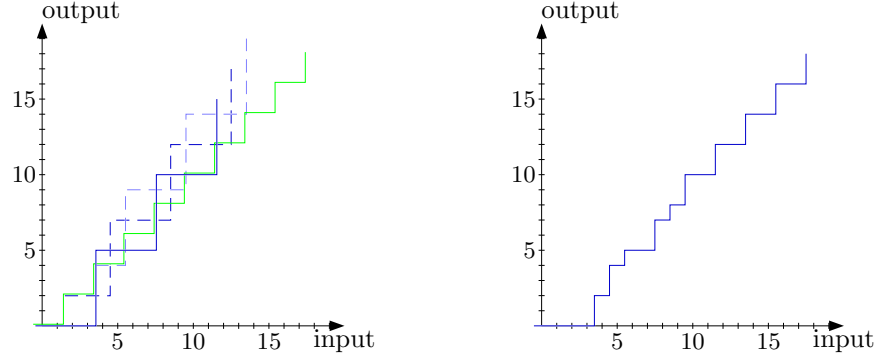
Fig. 8: Traces

Definition 3.6. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. We define the set S_n of *n-defined* stream terms, i.e. finite stream terms with a constructor-prefix of length $n \in \mathbb{N}$:

$$S_n := \{u_1 : \ldots : u_n : \sigma \mid u_1, \ldots, u_n \in \mathit{Ter}(\mathcal{C}(\Sigma_D), \varnothing)\} \,,$$

Moreover, let $\mathsf{g} \in \Sigma_{sfun}$ with $k = ar_s(\mathsf{g})$ and $\ell = ar_d(\mathsf{g})$, and define, for all $\boldsymbol{n} = n_1, \ldots, n_k$, the set $G_{\boldsymbol{n}}$ of applications of g to n_i-defined arguments:

$$G_{\boldsymbol{n}} := \{\, \mathsf{g}(\boldsymbol{s}, \boldsymbol{v}) \mid \forall i.\, s_i \in S_{n_i},\ \forall j.\, v_j \in \mathit{Ter}(\mathcal{C}(\Sigma_D), \varnothing) \,\} \,,$$

where $\boldsymbol{s} = s_1, \ldots, s_k$ and $\boldsymbol{v} = v_1, \ldots, v_\ell$. Then, the *optimal production modulus* $\nu_{\mathsf{g}} : \overline{\mathbb{N}}^k \to \overline{\mathbb{N}}$ *of* g is defined by:

$$\nu_{\mathsf{g}}(\boldsymbol{n}) := \inf \{\varPi_{\mathcal{T}}(t) \mid t \in G_{\boldsymbol{n}}\} \,.$$

To illustrate the difference between the optimal production modulus ν_{h} and the d-o lower bound $\underline{do}_{\mathcal{T}}(\mathsf{h})$, consider the following stream function specification:

$$\mathsf{h}(0 : x : \sigma) \to x : x : \mathsf{h}(0 : \sigma) \qquad\qquad (\rho_{\mathsf{h}_0})$$
$$\mathsf{h}(1 : x : \sigma) \to x : \mathsf{h}(0 : \sigma) \qquad\qquad (\rho_{\mathsf{h}_1})$$

with $\Sigma_D = \{0, 1\}$. Then $\nu_{\mathsf{h}}(n) = 2n \mathbin{\dot-} 3$ is the optimal production modulus of the stream function h. To obtain this bound one has to take into account that the data element 0 is supplied to the recursive call and conclude that (ρ_{h_1}) is only applicable in the first step of a rewrite sequence $\mathsf{h}(u_1 : \ldots : u_n : \sigma) \to \ldots$. However, the d-o lower bound is $\underline{do}_{\mathcal{T}}(\mathsf{h})(n) = n \mathbin{\dot-} 1$, derived from rule (ρ_{h_1}).

The following lemmas state an observation about the role of the opponent and the rewrite player. Basically, the opponent $\mathcal{G}$ can select the rule which is applicable for each position in the term; the rewrite player $\mathcal{R}$ can choose which position to rewrite. We use subscripts for pebbles $\bullet$, for example $\bullet_w$, to introduce 'names' for referring to these pebbles.

Definition 3.7 (Instantiation with respect to a rule ρ). For $s \in Ter(\langle\!|\Sigma|\!\rangle)_S$ with $s \equiv f(s_1, \ldots, s_{ar_s(f)}, \bullet_1, \ldots, \bullet_{ar_d(f)})$ and $\rho \in R$ a defining rule of f, we define a data-exchange function $\mathcal{G}_{s,\rho}$ as follows. Note that ρ is of the form:

$$\rho : f(\boldsymbol{u}_1 : \sigma_1, \ldots, \boldsymbol{u}_n : \sigma_n, v_1, \ldots, v_{ar_d(f)}) \to r \in R \,.$$

For $i = 1, \ldots, ar_s(f)$ let $n_i \in \mathbb{N}$ be maximal such that $s_i \equiv \bullet_{i,1} : \ldots \bullet_{i,n_i} : s_i'$.

Let $\mathcal{G}_{s,\rho}$ for all $1 \leq i \leq ar_d(f)$ instantiate the pebbles $\bullet_i$ with closed instances of the data patters v_i, and for all $1 \leq i \leq ar_s(f)$, $1 \leq j \leq \min(n_i, |\boldsymbol{u}_i|)$ instantiate the pebble $\bullet_{i,j}$ with closed instances of the data pattern $\boldsymbol{u_{ij}}$, respectively.

Lemma 3.8. *Let $s \in Ter(\langle\!|\Sigma|\!\rangle)_S$, $s \equiv f(s_1, \ldots, s_{ar_s(f)}, \bullet_1, \ldots, \bullet_{ar_d(f)})$ and $\rho \in R$ a defining rule of f; we use the notation from Def. 3.7. Then $\mathcal{G}_{s,\rho}(s)$ is a redex if and only if for all $1 \leq i \leq ar_s(f)$ we have: $\boldsymbol{u}_i \leq n_i$, that is, there is 'enough supply for the application of ρ'. Furthermore if $\mathcal{G}_{s,\rho}(s)$ is a redex, then it is a redex with respect to ρ (and no other rule from R by orthogonality).*

Proof. A consequence of the fact that $\mathcal{T}$ is an orthogonal constructor TRS. □

Lemma 3.9. *Let $\mathcal{G}$ be a data-exchange function, $t \in Ter(\langle\!|\Sigma|\!\rangle)$ a term and define $\mathcal{P} = \{p \mid p \in Pos(t),\ root(t|_p) \in \Sigma\}$. For every $\varsigma : \mathcal{P} \to R$ such that $\varsigma(p)$ is a defining rule for $root(t|_p)$ for all $p \in \mathcal{P}$, there is a data-exchange function $\mathcal{G}^{t,\varsigma}$ such that for all $p \in \mathcal{P}$ if the term $\mathcal{G}^{t,\varsigma}(t)|_p$ is a redex, then it is an $\varsigma(p)$-redex, and $\mathcal{G}^{t,\varsigma}(s) \equiv \mathcal{G}(s)$ for all $s \not\equiv t$.*

Proof. For all $p \in \mathcal{P}$ and $\varsigma(p) \equiv \ell \to r$ we alter $\mathcal{G}$ to obtain a data-guess function $\mathcal{G}^{t,\varsigma}$ as follows. If $q \in Pos(\ell)$ such that $\ell|_q$ is a data term and none of the $t(q \cdot p')$ with p' a non empty prefix of p is a symbol from Σ_{sfun}, then instantiate the data term at position $q \cdot p$ in t with an instance of the data pattern $\ell|_q$. Then if $\mathcal{G}^{t,\varsigma}(t)$ is a redex, then by orthogonality of $\mathcal{T}$ it can only be a $\ell \to r$-redex. □

History Aware Data-Exchange

Above we have modelled the opponent using history free data-exchange strategies and the rewrite player was omniscient, that is, she always chooses the best possible rewrite sequence, which produces the maximum possible number of elements.

Now we investigate the robustness of our definition. We strengthen the opponent, allowing for history aware data-exchange strategies, and we weaken the rewrite player, dropping omniscience, assuming only an outermost-fair rewrite strategy. However it turns out that these changes do not affect the d-o production range $\overline{do}_{\mathcal{T}}(s)$, in this way providing evidence for the robustness of our first definition.

Definition 3.10. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. A *history in $\mathcal{T}$* is a finite list in $(Ter(\Sigma)_S \times R \times \mathbb{N}^* \times Ter(\Sigma)_S)^*$ of the form:

$$\langle s_0, \ell_0 \to r_0, p_0, t_0 \rangle \langle s_1, \ell_1 \to r_1, p_1, t_1 \rangle \ldots \langle s_n, \ell_n \to r_n, p_n, t_n \rangle$$

such that $\forall 0 \leq i \leq n$: $s_i \to t_i$ by an application of rule $\ell_i \to r_i$ at position p_i. We use $H_{\mathcal{T}}$ to denote the set of all histories of $\mathcal{T}$.

A function $\mathcal{G} : H_{\mathcal{T}} \to (\mathit{Ter}(\lvert\Sigma\rvert))_S \to \mathit{Ter}(\Sigma)_S)$ from histories in $\mathcal{T}$ to data-exchange functions on $\mathcal{T}$ is called a *history-aware data-exchange function on $\mathcal{T}$*. For such a function we write $\mathcal{G}_h$ as shorthand for $\mathcal{G}(h)$ for all $h \in H_{\mathcal{T}}$.

Definition 3.11. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. Let $\mathcal{G}$ be a history-aware data-exchange function. We define the ARS $\mathcal{H}_{\mathcal{T},\mathcal{G}} \subseteq (\mathit{Ter}(\lvert\Sigma\rvert))_S \times H_{\mathcal{T}}) \times (\mathit{Ter}(\lvert\Sigma\rvert))_S \times H_{\mathcal{T}})$ as follows:

$$\mathcal{H}_{\mathcal{T},\mathcal{G}} := \{ \langle \langle\lvert s\rvert\rangle, h \rangle \to \langle \langle\lvert t\rvert\rangle, h\langle s, \ell \to r, p, t \rangle \rangle \mid$$
$$s, t \in \mathit{Ter}(\Sigma),\ h \in H_{\mathcal{T}},\ \text{with } \mathcal{G}_h(\lvert s\rvert) = s \text{ and } s \to_{\mathcal{T}} t \} .$$

For $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ we define the *production function $\Pi_{\mathcal{B}} : \mathit{Ter}(\Sigma)_S \to \overline{\mathbb{N}}$* by:

$$\Pi_{\mathcal{B}}(s) := \sup \{ n \in \mathbb{N} \mid \langle s, \epsilon \rangle \to^*_{\mathcal{B}} \langle u_1 : \ldots : u_n : t, h \rangle \} ,$$

that is, the production of s starting with empty history ϵ.

In an ARS $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ we allow to write $s_0 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} s_1 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} s_2 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \ldots$ when we will actually mean a rewrite sequence in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ of the form $\langle s_0, h_0 \rangle \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \langle s_1, h_1 \rangle \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \langle s_2, h_2 \rangle \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \ldots$ with some histories $h_0, h_1, h_2, \ldots \in H_{\mathcal{T}}$.

The notion of *positions of rewrite steps and redexes* extends to the ARS $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ in the obvious way, since the steps in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ are projections of steps in $\mathcal{T}$. Note that the opponent can change the data elements in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ at any time, therefore the rules might change with respect to which a certain position is a redex. For this reason we adapt the notion of outermost-fairness for our purposes. Intiutively, a rewrite sequence is outermost-fair if every position which is always eventually an outermost redex position, is rewritten infinitely often. Formally, we define a rewrite sequence $\langle s_1, h_1 \rangle \to \langle s_2, h_2 \rangle \to \ldots$ in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ to be *outermost-fair* if it is finite, or it is infinite and whenever p is an outermost redex position for infinitely many s_n, $n \in \mathbb{N}$, then there exist infinitely many rewrite steps $s_m \to s_{m+1}$, $m \in \mathbb{N}$, at position p. Moreover, a strategy $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ is *outermost-fair* if every rewrite sequence with respect to this strategy is outermost-fair.

Proposition 3.12. *For every stream specification $\mathcal{T} = \langle \Sigma, R \rangle$ there exists a computable, deterministic om-fair strategy on $\mathcal{H}_{\mathcal{T},\mathcal{G}}$.*

Definition 3.13. The *history aware d-o production range $\overline{\underline{do}}^{\mathcal{H}}_{\mathcal{T}}(s)$* of $s \in \mathit{Ter}(\lvert\Sigma\rvert))_S$ is defined as follows:

$$\overline{\underline{do}}^{\mathcal{H}}_{\mathcal{T}}(s) := \{ \Pi_{\mathcal{B}}(s) \mid \mathcal{G} \text{ a history-aware data-exchange function on } \mathcal{T},$$
$$\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}} \text{ an outermost-fair strategy} \} .$$

For $s \in \mathit{Ter}(\Sigma)_S$ define $\overline{\underline{do}}^{\mathcal{H}}_{\mathcal{T}}(s) := \overline{\underline{do}}^{\mathcal{H}}_{\mathcal{T}}(\lvert s\rvert)$.

Note that also the strategy of the rewrite player $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ is history aware since the elements of the ARS $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ have history annotations.

Lemma 3.14. *Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. Let $\mathsf{f} \in \Sigma_{sfun}$ with $ar_s(\mathsf{f}) = n$ and $ar(\mathsf{f}) = n'$, and let $s_1, \ldots, s_n \in Ter(\Sigma)_S$, $u_1, \ldots, u_{n'} \in Ter(\Sigma)_D$. Then it holds:*

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(s_1, \ldots, s_n, u_1, \ldots, u_{n'})) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma, u_1, \ldots, u_{n'})), \quad (1)$$

where $m_i := \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s_i)$ for all $i \in \{1, \ldots, n\}$.

Proof. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. Let $\mathsf{f} \in \Sigma_{sfun}$ with $ar_s(\mathsf{f}) = n$, and $s_1, \ldots, s_n \in Ter(\Sigma)_S$. We assume that the data arity of f is zero; in the presence of data arguments the proof proceeds analogously. Furthermore we let $m_i := \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s_i)$ for $i \in \{1, \ldots, n\}$. We show (1) by demonstrating the inequalities "$\leq$" and "$\geq$" in this equation.

$\geq$: Suppose that $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(s_1, \ldots, s_n)) = \Pi_{\mathcal{B}}(\langle\!\langle \mathsf{f}(s_1, \ldots, s_n) \rangle\!\rangle) = N \in \mathbb{N}$ for some om-fair strategy $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ and some history-aware data-exchange function $\mathcal{G}$ on $\mathcal{T}$. (In case that $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(s_1, \ldots, s_n)) = \infty$, nothing remains to be shown.) Then there exists an om-fair rewrite sequence

$$\xi : \langle\!\langle \mathsf{f}(s_1, \ldots, s_n) \rangle\!\rangle \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_1 \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_2 \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \cdots$$
$$\cdots \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_l \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_{l+1} \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \cdots$$

in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ with $\lim_{l \to \infty} \#_:(u_l) = N$. Due to the definition of the numbers m_i, ah history-aware data-exchange function $\mathcal{G}'$ from $H_{\mathcal{T}}$ to data-exchange functions can be defined that enables a rewrite sequence

$$\xi' : \mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma) \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} u_1' \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} u_2' \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} \cdots$$
$$\cdots \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} u_l' \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} u_{l+1}' \rightarrow_{\overline{\overline{\mathcal{H}}}_{\mathcal{T},\mathcal{G}'}} \cdots$$

with the properties that $\#_:(u_l) = \#_:(u_l')$ holds for all $l \in \mathbb{N}$, and that ξ' is the projection of ξ to a rewrite sequence with source $\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)$. More precisely, ξ' arises as follows: Steps $u_l \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_{l+1}$ in ξ that contract redexes in descendants of the outermost symbol f in the source of ξ give rise to steps $u_l' \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} u_{l+1}'$ that contract redexes at the same position and apply the same rule. This is possible because, due to the definition of the numbers m_i, always enough pebble-supply is guaranteed to carry out steps in ξ' corresponding to such steps in ξ. Steps $u_l \rightarrow_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_{l+1}$ that contract redexes in descendants of one of the subterms s_1, ..., s_n in the source of ξ project to empty steps $u_l' = u_{l+1}'$ in ξ'. (On histories $h \in H_{\mathcal{T}}$ and terms $r \in Ter(\langle\!\langle \Sigma \rangle\!\rangle)_S$ occurring in ξ, $\mathcal{G}'$ is defined in such a way as to make this projection possible. On histories h and terms r that do not occur in ξ', $\mathcal{G}'$ is defined arbitrarily with the only restriction $\langle\!\langle \mathcal{G}_h(r) \rangle\!\rangle$ ensuring that $\mathcal{G}'$ behaves as a history-aware data-exchange function on these terms.) By its construction, ξ' is again om-fair. Now let $\mathcal{B}'$ be the extension of the sub-ARS of $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ induced by ξ' to a deterministic om-fair strategy for $\mathcal{H}_{\mathcal{T},\mathcal{G}'}$. (Choose an arbitrary deterministic om-fair strategy $\tilde{\mathcal{B}}$ on $\mathcal{H}_{\mathcal{T},\mathcal{G}}$, which is

possible by Proposition 3.12. On term-history pairs that do not occur in ξ', define $\mathcal{B}'$ according to $\tilde{\mathcal{B}}$.) Then ξ' is also a rewrite sequence in $\mathcal{B}'$ that witnesses $\Pi_{\mathcal{B}}(\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)) = \lim_{l \to \infty} \#_:(u_l') = \lim_{l \to \infty} \#_:(u_l) = N = \Pi_{\mathcal{B}}((\![\mathsf{f}(s_1, \ldots, s_n)]\!))$. Now it follows:

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)) \leq \Pi_{\mathcal{B}'}(\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma))$$
$$= \Pi_{\mathcal{B}}((\![\mathsf{f}(s_1, \ldots, s_n)]\!)) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(s_1, \ldots, s_n)) \,,$$

establishing "$\geq$" in (1).

$\leq$: Suppose that $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(\bullet^{m_1}:\sigma, \ldots, \bullet^{m_n}:\sigma)) = \Pi_{\mathcal{B}}(\mathsf{f}(\bullet^{m_1}:\sigma, \ldots, \bullet^{m_n}:\sigma)) = N$ for some $N \in \mathbb{N}$, and an om-fair strategy $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ and a history-aware data-exchange function $\mathcal{G}$ on $\mathcal{T}$. (In case that $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)) = \infty$, nothing remains to be shown.) Then there exists an om-fair rewrite sequence

$$\xi : \ \mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)$$
$$\to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_1 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_2 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \cdots \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_l \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} u_{l+1} \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \cdots$$

in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ with $\lim_{l \to \infty} \#_:(u_l) = N$. Let $l_0 \in \mathbb{N}$ be minimal such that $\#_:(u_{l_0}) = N$. Furthermore, let, for all $i \in \{1, \ldots, n\}$, $m_i' \leq m_i$ be minimal such that each of the topmost m_i' symbols "$\bullet$" of the subterm $\bullet^{m_i} : \sigma$ of the source $\mathsf{f}(\bullet^{m_1} : \sigma, \ldots, \bullet^{m_n} : \sigma)$ of ξ is part of a redex pattern at some step among the first l_0 steps of ξ.
Since by assumption, we have $m_i := \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s_i)$ for $i \in \{1, \ldots, n\}$, there exist, for each $i \in \{1, \ldots, n\}$, history-aware data-guess functions $\mathcal{G}_i$, and om-fair strategies $\mathcal{B}_i \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}_i}$ such that $\Pi_{\mathcal{B}_i}((\![s_i]\!)) = m_i$; let all $\mathcal{B}_i$ and $\mathcal{G}_i$ be chosen as such. Then there exist, for each i, om-fair rewrite sequences of the form:

$$\xi_i : \ (\![s_i]\!) \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}_i}} u_{i,1} \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}_i}} \cdots \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}_i}} u_{i,l_i} = \bullet^{m_i'} : \tilde{u}_{i,l_i} \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}_i}} \cdots$$

in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ with $\lim_{l \to \infty} \#_:(u_{i,l}) = m_i$.
Now it is possible to combine the history-aware data-exchange functions $\mathcal{G}$, $\mathcal{G}_1, \ldots, \mathcal{G}_n$ into a function $\mathcal{G}'$ that makes it possible to combine the rewrite sequences ξ and $\xi_1, \ldots, \xi_n$ into a rewrite sequence of the form:

$$\xi' : (\![\mathsf{f}(s_1, \ldots, s_n)]\!)$$
$$\to^*_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} \ \mathsf{f}(\bullet^{m_1'} : u_{1,l_1}, \ldots, \bullet^{m_n'} : u_{n,l_n})$$

$\qquad$ (carry out the first l_i steps in ξ_i in the context $\mathsf{f}(\square_1, \ldots, \square_n)$)

$$\to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} u_1' \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} u_2' \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} \cdots \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} u_{l_0}' = \bullet^N : \tilde{u}_{l_0}'$$

$\qquad$ (parallel to the first l_0 steps of ξ)

$$\to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} w_1 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} w_2 \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}'}} \cdots$$

$\qquad$ (fair interleaving of the rests of the ξ_i put in context
$\qquad$ and of steps parallel to the rest of ξ')

By its construction, ξ' is again om-fair and it holds: $\lim_{l \to \infty} w_l = N$. Now let $\mathcal{B}'$ be the extension of the sub-ARS of $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ induced by ξ' to a deterministic

om-fair strategy for $\mathcal{H}_{\mathcal{T},\mathcal{G}'}$. (Again choose an arbitrary deterministic om-fair strategy $\tilde{\mathcal{B}}$ on $\mathcal{H}_{\mathcal{T},\mathcal{G}}$, which is possible by Proposition 3.12. On term-history pairs that do not occur in ξ', define $\mathcal{B}'$ according to $\tilde{\mathcal{B}}$.) Then ξ' is also a rewrite sequence in $\mathcal{B}'$ that witnesses $\Pi_{\mathcal{B}}(\langle\!| f(s_1,\ldots,s_n)|\!\rangle) = \lim_{l\to\infty} \#_:(w_l) = N = \Pi_{\mathcal{B}}(f(\bullet^{m_1}:\sigma,\ldots,\bullet^{m_n}:\sigma))$. Now it follows:

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(f(s_1,\ldots,s_n)) \le \Pi_{\mathcal{B}}(\langle\!| f(s_1,\ldots,s_n)|\!\rangle)$$
$$= \Pi_{\mathcal{B}'}(f(\bullet^{m_1}:\sigma,\ldots,\bullet^{m_n}:\sigma)) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(f(\bullet^{m_1}:\sigma,\ldots,\bullet^{m_n}:\sigma)) \,,$$

establishing "$\le$" in (1). $\square$

Lemma 3.15. *Let* $s \in Ter(\langle\!|\Sigma|\!\rangle)_S$ *be a stream term and* $\phi : Var_S \to Ter(\langle\!|\Sigma|\!\rangle)_S$ *a substitution of stream variables. Then*

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(s^\phi) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s^\varphi)$$

where $\varphi : Var_S \to Ter(\langle\!|\Sigma|\!\rangle)_S$ *is defined by* $\varphi(\tau) = \bullet^{m_\tau}:\sigma$, *and* $m_\tau := \underline{do}_{\mathcal{T}}^{\mathcal{H}}(\phi(\tau))$.

Proof. Induction using Lem. 3.14. $\square$

Corollary 3.16. *Let* $s \in Ter(\langle\!|\Sigma|\!\rangle)_S$ *and* $\phi,\varphi : Var_S \to Ter(\langle\!|\Sigma|\!\rangle)_S$ *substitutions of stream variables such that* $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(\phi(\tau)) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(\varphi(\tau))$ *for all* $\tau \in Var_S$. *Then*

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(s^\phi) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s^\varphi) \,.$$

Proof. Two applications of Lem. 3.15. $\square$

We define a history free data-exchange function $\underline{\mathcal{G}}_{\mathcal{T}}$ such that no reduction with respect to $\underline{\mathcal{G}}_{\mathcal{T}}$ produces more than $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$ elements.

Definition 3.17. Let a stream specification $\mathcal{T} = \langle \Sigma, R \rangle$ be given, and let $>$ be a well-founded order on R. We define the *lower bound data-exchange function* $\underline{\mathcal{G}}_{\mathcal{T}} : Ter(\langle\!|\Sigma|\!\rangle)_S \to Ter(\Sigma)_S$ as follows.

Let $s \in Ter(\langle\!|\Sigma|\!\rangle)_S$ and $f(s_1,\ldots,s_{ar_s(f)},\bullet_1,\ldots,\bullet_{ar_d(f)})$ a subterm occurrence of s. For $i = 1,\ldots, ar_s(f)$ let $n_i \in \mathbb{N}$ be maximal such that $s_i \equiv \bullet_{i,1}:\ldots\bullet_{i,n_i} : s_i'$. We define $m_i := \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s_i)$ for $1 \le i \le n$, then we have by Lem. 3.14:

$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(t := f(s_1,\ldots,s_{ar_s(f)},\bullet_1,\ldots,\bullet_{ar_d(f)})) =$$
$$\underline{do}_{\mathcal{T}}^{\mathcal{H}}(t' := f(\bullet^{m_1}:\sigma,\ldots,\bullet^{m_{ar_s(f)}}:\sigma,\bullet_1,\ldots,\bullet_{ar_d(f)})) \,.$$

We choose a defining rule ρ of f as follows. In case there exists a rule

$$\gamma : f(\boldsymbol{u}_1:\sigma_1,\ldots,\boldsymbol{u}_n:\sigma_n,v_1,\ldots,v_{ar_d(f)}) \to t \in R$$

such that $m_i < |\boldsymbol{u}_i|$ for some $i \in \{1,\ldots,n\}$, then let $\rho := \gamma$. In case there are multiple possible choices for γ, we pick the minimal γ with respect to $>$.

Otherwise in exists a history-aware data-exchange function $\mathcal{G}$, and an outermost-fair rewrite sequence $\sigma : t' \to_{\mathcal{H}_{\mathcal{T},\mathcal{G}}} \ldots$ in $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ producing only $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(t')$ elements. From exhaustivity of $\mathcal{T}$ we get $\mathcal{G}(t')$ is not a normal form, since all rules have

enough supply. Moreover, by orthogonality exactly one defining rule γ of f is applicable, let $\rho := \gamma$. Again, in case there are multiple possible choices for γ, due to freedom in the choice of the data-exchange function $\mathcal{G}$, we take the minimal γ with respect to $>$.

We define $\underline{\mathcal{G}}_{\mathcal{T}}$ to instantiate:

- for $1 \leq i \leq ar_d(\mathsf{f})$ the occurrences of pebbles $\bullet_i$ in s, and
- for $1 \leq i \leq ar_s(\mathsf{f})$, $1 \leq j \leq n_i$ the occurrences of pebbles $\bullet_{i,j}$ in s

with respect to the rule ρ (Def. 3.7).

Rewrite steps with respect to the lower bound data-exchange function $\underline{\mathcal{G}}_{\mathcal{T}}$ do not change the d-o lower bound of the production of a term.

Lemma 3.18. *For all $s, s \in Ter(\langle\!\langle\Sigma\rangle\!\rangle)_S$: $s \to_{\mathcal{A}_{\mathcal{T},\underline{\mathcal{G}}_{\mathcal{T}}}} t$ implies $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(s) = \underline{do}_{S}^{\mathcal{H}}(t)$.*

Proof. We use the notions introduced in Def. 3.17. Note that the lower bound data-exchange function $\underline{\mathcal{G}}_{\mathcal{T}}$ is defined in a way that the pebbles are instantiated independent of the context above the stream function symbol. Therefore it is sufficient to consider rewrite steps $s \to_{\mathcal{A}_{\mathcal{T},\underline{\mathcal{G}}_{\mathcal{T}}}} t$ at the root, closure under contexts follows from Cor. 3.16. $\square$

Corollary 3.19. *For all $s \in Ter(\langle\!\langle\Sigma\rangle\!\rangle)_S$ we have: $\Pi_{\mathcal{A}_{\mathcal{T},\underline{\mathcal{G}}_{\mathcal{T}}}}(s) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$.*

Proof. Direct consequence of Lem. 3.18. $\square$

Hence, as a consequence of Cor. 3.19, the lower bound of the history-free d-o production conincides with the lower bound of the history-aware d-o production.

Lemma 3.20. *For all $s \in Ter(\Sigma)_S$ we have: $\underline{do}_{\mathcal{T}}(s) = \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$.*

Proof. The direction $\underline{do}_{\mathcal{T}}(s) \geq \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$ is trivial, and $\underline{do}_{\mathcal{T}}(s) \leq \underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$ follows from Cor. 3.19, that is, using the history-free data-exchange function $\underline{\mathcal{G}}_{\mathcal{T}}$ no reduction produces more than $\underline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$ elements. $\square$

Lemma 3.21. *For all $s \in Ter(\Sigma)_S$ we have: $\overline{do}_{\mathcal{T}}(s) = \overline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$.*

Proof. $\subseteq$ Let $n \in \overline{do}_{\mathcal{T}}(s)$. Then there exists a data-exchange function $\mathcal{G}$ such that $\Pi_{\mathcal{A}_{\mathcal{T},\mathcal{G}}}(s) = n$, and a rewrite sequence $\sigma : s_0 \to_{\mathcal{A}_{\mathcal{T},\mathcal{G}}} \dots \to_{\mathcal{A}_{\mathcal{T},\mathcal{G}}} s_m$ with $s_m \equiv u_1 : \dots : u_n : t$. We define for all $h \in \mathcal{H}_{\mathcal{T}}$ a data-exchange function $\mathcal{G}_h$ by setting $\mathcal{G}_h(s) := \mathcal{G}(s)$ for all $s \in Ter(\langle\!\langle\Sigma\rangle\!\rangle)_S$. Moreover, we define the strategy $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ to execute σ and continue outermost-fair afterwards. Since s_m does not produce more than n elements, every maximal rewrite sequence with respect to $\mathcal{B}$ will produce exactly n elements. Hence $n \in \overline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$.

$\supseteq$ Let $n \in \overline{do}_{\mathcal{T}}^{\mathcal{H}}(s)$. Then there exist data-exchange functions $\mathcal{G}_h$ for every $h \in \mathcal{H}_{\mathcal{T}}$, and an outermost-fair strategy $\mathcal{B} \subseteq \mathcal{H}_{\mathcal{T},\mathcal{G}}$ such that $\Pi_{\mathcal{B}}(s) = n$. There exists a (possibly infinite) maximal, outermost-fair rewrite sequence

$$\sigma : \langle s, \epsilon \rangle \equiv \langle s_0, h_0 \rangle \to_{\mathcal{B}} \dots \to_{\mathcal{B}} \langle s_m, h_m \rangle \to_{\mathcal{B}} \dots$$

such that $s_m \equiv u_1 : \ldots : u_n : t$. Since the elements of $\mathcal{H}_{\mathcal{T},\mathcal{G}}$ are annotated with their history, we do not encounter repetitions during the rewrite sequence. Hence, w.l.o.g. we can assume that $\mathcal{B}$ is a deterministic strategy, admitting only the maximal rewrite sequence σ. Disregarding the history annotations there might exist repetitions, that is, $i < j \in \mathbb{N}$ with $s_i \equiv s_j$. Nevertheless, from σ we can construct a history free data-exchange function $\mathcal{G}$ as follows. For all $s \in Ter((\!|\Sigma|\!))_S$ let $S := \{i \in \mathbb{N} \mid s_i \equiv s\}$ and define:

$$\mathcal{G}'(s) := \begin{cases} \mathcal{G}_{h_j}(s_j) & \text{if } j := \sup S < \infty \\ \mathcal{G}_{h_k}(s_k) & \text{if } \sup S = \infty, \ k := \inf S \\ \text{arbitrary} & \text{if } \neg\exists i \in \mathbb{N}. \ s_i \equiv s \end{cases}$$

$\square$

4   The Production Calculus

As a means to compute the d-o production behaviour of stream specifications, we introduce a 'production calculus' with periodically increasing functions as its central ingredient. This calculus is the term equivalent of the calculus of pebbleflow nets that was introduced in [2], see Sec. 4.3.

4.1   Periodically Increasing Functions

We use $\overline{\mathbb{N}} := \mathbb{N} \uplus \{\infty\}$, the *extended natural numbers*, with the usual $\leq$, $+$, and we define $\infty - \infty := 0$. An infinite sequence $\sigma \in X^\omega$ is *eventually periodic* if $\sigma = \alpha\beta\beta\beta \ldots$ for some $\alpha \in X^*$ and $\beta \in X^+$. A function $f : \mathbb{N} \to \overline{\mathbb{N}}$ is *eventually periodic* if the sequence $\langle f(0), f(1), f(2), \ldots \rangle$ is eventually periodic.

Definition 4.1. A function $g : \mathbb{N} \to \overline{\mathbb{N}}$ is called *periodically increasing* if it is increasing, i.e. $\forall n.\ g(n) \leq g(n+1)$, and the *derivative of g*, $n \mapsto g(n+1) - g(n)$, is eventually periodic. A function $h : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ is called *periodically increasing* if its restriction to $\mathbb{N}$ is periodically increasing and if $h(\infty) = \lim_{n\to\infty} h(n)$. Finally, a k-ary function $i : (\overline{\mathbb{N}})^k \to \overline{\mathbb{N}}$ is called *periodically increasing* if $i(n_1, ..., n_k) = \min(i_1(n_1), \ldots, i_k(n_k))$ for some unary periodically increasing functions $i_1, \ldots, i_k$.

Periodically increasing (p-i) functions can be denoted by their value at 0 followed by a representation of their derivative. For example, $03\overline{12}$ denotes the p-i function $f : \mathbb{N} \to \mathbb{N}$ with values $0, 3, 4, 6, 7, 9, \ldots$. However, we use a finer and more flexible notation over the alphabet $\{+, -\}$ that will be useful for computing several operations on p-i functions. For instance, we represent f as above by the 'io-sequence'

$$-{+}{+}{+}\overline{-{+}-{+}{+}} = -{+}{+}{+} -{+}-{+}{+} -{+}-{+}{+} -{+}-{+}{+} \ldots,$$

in turn abbreviated by the 'io-term' $\langle -{+}{+}{+}, -{+}-{+}{+} \rangle$.

Definition 4.2. Let $\pm := \{+, -\}$. An *io-sequence* is a finite ($\in \pm^*$) or infinite sequence ($\in \pm^\omega$) over $\pm$, the set of io-sequences is denoted by $\pm^\infty := \pm^* \cup \pm^\omega$. We let $\bot$ denote the empty io-sequence, and use it as an explicit end marker. An *io-term* is a pair $\langle \alpha, \beta \rangle$ of finite io-sequences $\alpha, \beta \in \pm^*$ with $\beta \neq \bot$. The set of io-terms is denoted by $\mathcal{I}$, and we use ι, κ to range over io-terms. For an io-sequence $\sigma \in \pm^\omega$ and an io-term $\iota = \langle \alpha, \beta \rangle \in \mathcal{I}$ we say that ι *denotes* σ if $\sigma = \alpha\overline{\beta}$ where $\overline{\beta}$ stands for the infinite sequence $\beta\beta\beta\ldots$. A sequence $\sigma \in \pm^\infty$ is *rational* if it is finite or if it is denoted by an io-term. The set of rational io-sequences is denoted by $\pm_{\mathrm{R}}^\infty$. An infinite sequence $\sigma \in \pm^\omega$ is *productive* if it contains infinitely many $+$'s:

$$\sigma \in \pm^\omega \text{ is } productive \iff \forall n. \exists m \geq n. \, \sigma(m) = + \, .$$

We let $\pm_{\mathrm{P}}^\omega$ denote the set of productive sequences and define $\pm_{\mathrm{P}}^\infty := \pm^* \cup \pm_{\mathrm{P}}^\omega$.

The reason for having both io-terms and io-sequences is that, depending on the purpose at hand, one or the other is more convenient to work with. Operations are often easier to understand on io-sequences than on io-terms, whereas we need finite representations to be able to compute these operations.

Definition 4.3. We define $[\![\sigma]\!] : \mathbb{N} \to \overline{\mathbb{N}}$, the *interpretation* of $\sigma \in \pm^\infty$, by:

$$[\![\bot]\!](n) = 0 \tag{2}$$
$$[\![+\sigma]\!](n) = 1 + [\![\sigma]\!](n) \tag{3}$$
$$[\![-\sigma]\!](0) = 0 \tag{4}$$
$$[\![-\sigma]\!](n+1) = [\![\sigma]\!](n) \tag{5}$$

for all $n \in \mathbb{N}$, and extend it to $\overline{\mathbb{N}} \to \overline{\mathbb{N}}$ by defining $[\![\sigma]\!](\infty) = \lim_{n\to\infty} [\![\sigma]\!](n)$. We say that $[\![\sigma]\!]$ *interprets* σ, and, conversely, that σ *represents* $[\![\sigma]\!]$. We overload notation and define $[\![\_]\!] : \mathcal{I} \to (\overline{\mathbb{N}} \to \overline{\mathbb{N}})$ by $[\![\langle \alpha, \beta \rangle]\!] := [\![\alpha\overline{\beta}]\!]$.

It is easy to verify that, for every $\sigma \in \pm^\infty$, the function $[\![\sigma]\!]$ is increasing, and that, for every $\iota \in \mathcal{I}$, the function $[\![\iota]\!]$ is periodically increasing. Furthermore, every increasing function is represented by an io-sequence, and every p-i function is denoted by an io-term.

Subsequently, for an eventually constant function $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, we write $\underline{\underline{f}}$ for the shortest finite io-sequence representing f (trailing $-$'s can be removed). For an always eventually strictly increasing function $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, i.e. $\forall n. \exists m > n. \, f(m) > f(n)$, we write $\underline{f}$ for the unique io-sequence representing f; note that then $\underline{f} \in \pm_{\mathrm{P}}^\omega$ follows. For a periodically increasing function $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, we write $\underline{f}$ for the io-term $\iota = \langle \alpha, \beta \rangle$ such that $[\![\iota]\!] = f$ and $|\alpha| + |\beta|$ is minimal; uniqueness follows from the following lemma:

Lemma 4.4. *For all p-i functions $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, the term $\underline{f} \in \mathcal{I}$ is unique.*

Proof. Consider the 'compression' TRS consisting of the rules:

$$\langle \alpha x, \beta x \rangle \to \langle \alpha, x\beta \rangle \qquad\qquad (x \in \pm) \qquad\qquad (6)$$

$$\langle \alpha, \beta^p \rangle \to \langle \alpha, \beta \rangle \qquad\qquad (n > 1) \qquad\qquad (7)$$

Clearly, $\iota \to \kappa$ implies $|\iota| > |\kappa|$ where, for $\iota = \langle \alpha, \beta \rangle$, $|\iota| := |\alpha| + |\beta|$. Hence, compression is terminating.

There are two critical pairs due to an overlap of rule (7) with itself, and in rules (6) and (7): $\langle \langle \alpha, \beta^n \rangle, \langle \alpha, \beta^m \rangle \rangle$ originating from a term of the form $\langle \alpha, \beta^{n \cdot m} \rangle$, and $\langle \langle \alpha, (x\beta)^n \rangle, \langle \alpha x, \beta x \rangle \rangle$ originating from a term of the form $\langle \alpha x, (\beta x)^n \rangle$, respectively. Both pairs are easily joinable in one step: the first to $\langle \alpha, \beta \rangle$, and the second to $\langle \alpha, x\beta \rangle$. Hence, the system is locally confluent. Therefore, by Newman's Lemma, it is also confluent, and normal forms are unique.

Finally, assume that two io-terms $\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle \in \mathcal{I}$ each of minimal length represent the same p-i function f. It is easy to see that then $\alpha_1 \overline{\beta_1} = \alpha_2 \overline{\beta_2}$. It follows that there exist $\langle \alpha, \beta \rangle$ such that $\langle \alpha, \beta \rangle \to^* \langle \alpha_1, \beta_1 \rangle$ and $\langle \alpha, \beta \rangle \to^* \langle \alpha_2, \beta_2 \rangle$. By confluence and termination of compression it follows that they have the same normal form $\langle \alpha_0, \beta_0 \rangle$. Since compression reduces the length of io-terms it must follow that $\langle \alpha_1, \beta_1 \rangle = \langle \alpha_0, \beta_0 \rangle = \langle \alpha_2, \beta_2 \rangle$. □

Note that we have that $[\![\underline{\underline{f}}]\!] = f$ for all increasing functions f, and $[\![\underline{f}]\!] = f$ for all p-i functions f. As an example one can check that the interpretation of the aforementioned io-term $\underline{f} = \langle -+++, -+-++ \rangle$ indeed has the sequence $0, 3, 4, 6, 7, 9, \ldots$ as its graph.

Remark 4.5. Note that Def. 4.3 is well-defined due to the productivity requirement on infinite io-sequences. This can be seen as follows: starting on $[\![\sigma]\!](n)$, after finitely many, say $m \in \mathbb{N}$, $-$'s, we either arrive at rule (3) (if $m \leq n$) or at rule (4) (if $m > n$). Had we not required infinite sequences $\sigma \in \pm^\omega$ to contain infinitely many $+$'s, then, e.g., the computation of $[\![--- \ldots]\!](\infty)$ would consist of applications of rule (5) only, and hence would not lead to an infinite normal form.

Proposition 4.6. *Unary periodically increasing functions are closed under composition and pointwise infimum.*

We want to constructively define the operations of composition, pointwise infimum, and least fixed point calculation of increasing functions (p-i functions) on their io-sequence (io-term) representations. We first define these operations on io-sequences by means of coinductive clauses. The operations on io-terms are more involved, because they require 'loop checking' on top. We will proceed by showing that the operations of composition and pointwise infimum on io-sequences preserve rationality. This will then give rise to the needed algorithms for computing the operations on io-terms.

Definition 4.7. The operation *composition* $\circ : \pm_P^\infty \times \pm_P^\infty \to \pm_P^\infty$, $\langle \sigma, \tau \rangle \mapsto \sigma \circ \tau$ *of (finite or productive) io-sequences* is defined coinductively as follows:

$$\bot \circ \tau = \bot \tag{8}$$

$$+\sigma \circ \tau = +(\sigma \circ \tau) \tag{9}$$

$$-\sigma \circ \bot = \bot \tag{10}$$

$$-\sigma \circ +\tau = \sigma \circ \tau \tag{11}$$

$$-\sigma \circ -\tau = -(-\sigma \circ \tau) \tag{12}$$

An argument similar to Rem. 4.5 concerning well-definedness applies for Def. 4.7.

Remark 4.8. Note that the defining rules (8)–(12) are orthogonal and exhaust all cases. As it stands, it is not immediately clear that this definition is well-defined (productive!), the problematic case being (11). Rule (11) is to be thought of as an internal communication between components σ and τ, as a silent step in the black box $\sigma \circ \tau$. How to guarantee that always, after finitely many internal steps, either the process ends or there will be an external step, in the form of output or a requirement for input? The recursive call in (11) is not guarded. However, well-definedness of composition can be justified as follows:
Consider arbitrary sequences $\sigma, \tau \in \pm_P^\infty$. By definition of $\pm_P^\infty$, there exists $m \in \mathbb{N}$ such that $\sigma = -^m +\sigma'$ for some $\sigma' \in \pm_P^\infty$ or $\sigma = -^m \bot$; likewise there exists $n \in \mathbb{N}$ such that $\tau = -^n +\tau'$ for some $\tau' \in \pm_P^\infty$, or $\tau = -^n \bot$. Rules (11) and (12) are decreasing with respect to the lexicographic order on (m, n). After finitely many applications of (11) and (12), one of the rules (8)–(10) must be applied. Hence composition is well-defined and the sequence produced is an element of $\pm_P^\infty$, i.e. either it is a finite sequence or it contains an infinite number of $+$'s.

Lemma 4.9. *Composition of io-sequences is associative.*

Proof. Let $R = \{\langle \sigma \circ (\tau \circ \upsilon), (\sigma \circ \tau) \circ \upsilon \rangle \mid \sigma, \tau, \upsilon \in \pm_P^\infty\}$. To show that R is a bisimulation, we prove that, for all $\sigma, \tau, \upsilon, \phi_1, \phi_2 \in \pm_P^\infty$, if $\phi_1 = \sigma \circ (\tau \circ \upsilon)$ and $\phi_2 = (\sigma \circ \tau) \circ \upsilon$, then either $\phi_1 = \bot = \phi_2$ or $\mathsf{head}(\phi_1) = \mathsf{head}(\phi_2)$ and $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in R$, by induction on the number $n \in \mathbb{N}$ of leading $-$'s of σ^4, and a sub-induction on the number $m \in \mathbb{N}$ of leading $-$'s of τ.

If $n = 0$ and $\sigma = \bot$, then $\phi_1 = \bot = \phi_2$. If $\sigma = +\sigma'$, we have $\phi_1 = +(\sigma' \circ (\tau \circ \upsilon))$, and $\phi_2 = +((\sigma' \circ \tau) \circ \upsilon)$, and so $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in R$.

If $n > 0$, then $\sigma = -\sigma'$, and we proceed by sub-induction on m. If $m = 0$ and $\tau = \bot$, then $\phi_1 = \bot = \phi_2$. If $\tau = +\tau'$, we compute $\phi_1 = \sigma' \circ (\tau' \circ \upsilon)$, and $\phi_2 = (\sigma' \circ \tau') \circ \upsilon$, and conclude by IH.

If $m > 0$, then $\tau = -\tau'$, and we proceed by case distinction on υ. If $\upsilon = \bot$, then $\phi_1 = \bot = \phi_2$. If $\upsilon = +\upsilon'$, we compute $\phi_1 = \sigma \circ (\tau' \circ \upsilon')$, and $\phi_2 = (\sigma \circ \tau') \circ \upsilon'$, and conclude by sub-IH. Finally, if $\upsilon = -\upsilon'$, then $\phi_1 = -(\sigma \circ (\tau \circ \upsilon'))$, and $\phi_2 = -((\sigma \circ \tau) \circ \upsilon')$. Clearly $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in R$. □

---

[4] By definition of $\pm_P^\infty$ the number of leading $-$'s of any sequence in $\pm_P^\infty$ is finite: for all $\sigma \in \pm_P^\infty$ there exists $n \in \mathbb{N}$ such that either $\sigma = -^n \bot$ or $\sigma = -^n +\sigma'$.

Remark 4.10. If we allow to use extended natural numbers at places where an io-sequence is expected, using a coercion $n \mapsto +^n$, with $+^0 = \bot$, one observes that the interpretation of an io-sequence (Def. 4.3) is just a special case of composition:

$$[\![\sigma]\!](n) = \sigma \circ n \,.$$

Proposition 4.11. *For all increasing functions* $f, g : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$: $[\![\underline{f \circ g}]\!] = [\![\underline{f}]\!] \circ [\![\underline{g}]\!]$.

Proof. Immediate from Rem. 4.10, and Lem. 4.9.   □

Next, we show that composition of io-sequences preserves rationality.

Lemma 4.12. *If* $\sigma, \tau \in \pm_{\mathrm{R}}^{\infty}$, *then* $\sigma \circ \tau \in \pm_{\mathrm{R}}^{\infty}$.

Proof. Let $\sigma, \tau \in \pm_{\mathrm{R}}^{\infty}$ and set $\sigma_0 := \sigma$ and $\tau_0 := \tau$. In the rewrite sequence starting with $\sigma_0 \circ \tau_0$ each of the steps is either of the form:

$$\sigma_n \circ \tau_n \to C_n[\sigma_{n+1} \circ \tau_{n+1}] \,, \tag{13}$$

where $\sigma_{n+1} \in \{\sigma_n, \mathsf{tail}(\sigma_n)\}$, $\tau_{n+1} \in \{\tau_n, \mathsf{tail}(\tau_n)\}$ and $C_n \in \{\Box, + : \Box, - : \Box\}$, or the rewrite sequence ends with a step of the form:

$$\sigma_n \circ \tau_n \to \bot \,.$$

In the latter case, $\sigma \circ \tau$ results in a finite and hence rational io-sequence.

Otherwise the rewrite sequence is infinite, consisting of steps of form (13) only. For $\theta \in \pm^*$ we define the context C_θ inductively: $C_\theta \equiv \Box$, for $\theta \equiv \bot$; and $C_{+\theta} \equiv +C_\theta$ as well as $C_{-\theta} \equiv -C_\theta$, for all $\theta \in \pm^*$. Because the sequences σ, τ are rational, the sets $\{\sigma_n \mid n \in \mathbb{N}\}$ and $\{\tau_n \mid n \in \mathbb{N}\}$ are finite. Then, the pigeonhole principle implies the existence of $i, j \in \mathbb{N}$ such that $i < j$, $\sigma_i = \sigma_j$ and $\tau_i = \tau_j$. Now let $\alpha, \gamma \in \pm^*$ with $\gamma \neq \bot$ such that $C_\alpha \equiv C_0[\ldots C_{i-1}[C_i] \ldots]$, $C_\gamma \equiv C_{i+1}[\ldots C_{j-1}[C_j] \ldots]$, and:

$$\sigma_0 \circ \tau_0 \to^* C_\alpha[\sigma_i \circ \tau_i] \,, \text{ and}$$
$$\sigma_i \circ \tau_i \to^* C_\gamma[\sigma_j \circ \tau_j] = C_\gamma[\sigma_i \circ \tau_i] \,.$$

Then we find an eventually 'spiralling' rewrite sequence:

$$\begin{aligned} \sigma_0 \circ \tau_0 &\to^* C_\alpha[\sigma_i \circ \tau_i] = \alpha(\sigma_i \circ \tau_i) \\ &\to^* C_\alpha[C_\gamma[\sigma_i \circ \tau_i]] = \alpha\gamma(\sigma_i \circ \tau_i) \\ &\to^* C_\alpha[C_\gamma[C_\gamma[\sigma_i \circ \tau_i]]] = \alpha\gamma\gamma(\sigma_i \circ \tau_i) \,, \end{aligned}$$

and therefore $\sigma_0 \circ \tau_0 \twoheadrightarrow \alpha\overline{\gamma}$. This shows that $\sigma \circ \tau$ is rational.   □

Next, we define the operation of pointwise infimum of increasing functions on io-sequences.

Definition 4.13. The operation *pointwise infimum* $\wedge : \pm^\infty \times \pm^\infty \rightarrow \pm^\infty$, $\langle \sigma, \tau \rangle \mapsto \sigma \wedge \tau$ *of io-sequences* is defined coinductively by the following equations:

$$\bot \wedge \tau = \bot \qquad\qquad\qquad \mathsf{del}(\bot) = \bot$$
$$\sigma \wedge \bot = \bot \qquad\qquad\qquad \mathsf{del}(+\sigma) = +\mathsf{del}(\sigma)$$
$$+\sigma \wedge +\tau = +(\sigma \wedge \tau) \qquad\qquad \mathsf{del}(-\sigma) = \sigma$$
$$-\sigma \wedge \tau = -(\sigma \wedge \mathsf{del}(\tau)) \quad (\tau \neq \bot)$$
$$\sigma \wedge -\tau = -(\mathsf{del}(\sigma) \wedge \tau) \quad (\sigma \neq \bot)$$

where $\mathsf{del}(\sigma)$ removes the first requirement of $\sigma \in \pm^\infty$ (if any).

We let $\circ$ bind stronger than $\wedge$.

Requirement removal distributes over pointwise infimum:

Lemma 4.14. *For all $\sigma, \tau \in \pm_P^\infty$, it holds that:* $\mathsf{del}(\sigma \wedge \tau) = \mathsf{del}(\sigma) \wedge \mathsf{del}(\tau)$.

Proof. Check that $\{\langle \mathsf{del}(\sigma \wedge \tau), \mathsf{del}(\sigma) \wedge \mathsf{del}(\tau) \rangle \mid \sigma, \tau \in \pm^\infty\} \cup \{\langle \sigma, \sigma \rangle \mid \sigma \in \pm^\infty\}$ is a bisimulation. $\qquad\square$

Lemma 4.15. *Infimum is idempotent, commutative, and associative.*

Proof. By coinduction, with the use of Lem. 4.14 in case of associativity. $\qquad\square$

In a composition requirements come from the second component:

Lemma 4.16. *For all $\sigma, \tau \in \pm_P^\infty$, it holds that:* $\mathsf{del}(\sigma \circ \tau) = \sigma \circ \mathsf{del}(\tau)$.

Composition distributes both left and right over pointwise infimum:

Lemma 4.17. *For all $\sigma, \tau, \upsilon \in \pm_P^\infty$, it holds that:* $\sigma \circ (\tau \wedge \upsilon) = \sigma \circ \tau \wedge \sigma \circ \upsilon$.

Proof. By coinduction; let $L = \{\langle \sigma \circ (\tau \wedge \upsilon), \sigma \circ \tau \wedge \sigma \circ \upsilon \rangle \mid \sigma, \tau, \upsilon \in \pm_P^\infty\}$. To show that L is a bisimulation, we prove that, for all $\sigma, \tau, \upsilon, \phi_1, \phi_2 \in \pm_P^\infty$, if $\phi_1 = \sigma \circ (\tau \wedge \upsilon)$ and $\phi_2 = \sigma \circ \tau \wedge \sigma \circ \upsilon$, then either $\phi_1 = \bot = \phi_2$, or $\mathsf{head}(\phi_1) = \mathsf{head}(\phi_2)$ and $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in L$, by induction on the number $n \in \mathbb{N}$ of leading $-$'s of σ.

If $n = 0$ and $\sigma = \bot$, then $\phi_1 = \bot = \phi_2$.

If $n = 0$ and $\sigma = +\sigma'$, then: $\phi_1 = +(\sigma' \circ (\tau \wedge \upsilon))$, and $\phi_2 = +(\sigma' \circ \tau \wedge \sigma' \circ \upsilon)$, and so $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in L$.

If $n > 0$ and $\sigma = -\sigma'$, we proceed by case analysis of τ and υ.

If one of τ, υ is empty, then $\phi_1 = \bot = \phi_2$.

If $\tau = +\tau'$ and $\upsilon = +\upsilon'$, then $\phi_1 = \sigma' \circ (\tau' \wedge \upsilon')$ and $\phi_2 = \sigma \circ \tau \wedge \sigma \circ \upsilon$, and we conclude by the induction hypothesis.

If $\tau = +\tau'$ and $\upsilon = -\upsilon'$, then $\phi_1 = -(\sigma \circ (\mathsf{del}(\tau) \wedge \upsilon'))$ and $\phi_2 = -(\mathsf{del}(\sigma \circ \tau) \wedge \sigma \circ \upsilon') = -(\sigma \circ \mathsf{del}(\tau) \wedge \sigma \circ \upsilon')$ by Lem. 4.16. Thus $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in L$.

The case $\tau = -\tau'$, $\upsilon = +\upsilon'$ is proved similarly.

Finally, if $\tau = -\tau'$ and $\upsilon = -\upsilon'$, we compute $\phi_1 = -(\sigma \circ (\tau' \wedge \upsilon'))$ and $\phi_2 = -(\sigma \circ \tau' \wedge \sigma \circ \upsilon')$, and conclude $\langle \mathsf{tail}(\phi_1), \mathsf{tail}(\phi_2) \rangle \in L$. $\qquad\square$

Lemma 4.18. *For all $\sigma, \tau, \upsilon \in \pm_P^\infty$, it holds that:* $(\tau \wedge \upsilon) \circ \sigma = \tau \circ \sigma \wedge \upsilon \circ \sigma$.

Proof. Analogous to the proof of Lem. 4.17.   □

The operation of pointwise infimum of (periodically) increasing functions is defined on their io-sequence (io-term) representations.

Proposition 4.19. *For all increasing functions $f, g : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$:* $[\![\underline{f \wedge g}]\!] = [\![\underline{f}]\!] \wedge [\![\underline{g}]\!]$.

Proof. Immediate from Rem. 4.10 and Lem. 4.18.   □

We give a coinductive definition of the calculation of the least fixed point of an io-sequence. The operation del was defined in Def. 4.13.

Definition 4.20. *The operation* $\mathsf{fix} : \pm^\infty \to \overline{\mathbb{N}}$ *computing the least fixed point of a sequence $\sigma \in \pm^\infty$, is defined, for all $\sigma \in \pm^\infty$, by:*

$$\mathsf{fix}(\bot) = 0$$
$$\mathsf{fix}(+\sigma) = 1 + \mathsf{fix}(\mathsf{del}(\sigma))$$
$$\mathsf{fix}(-\sigma) = 0$$

Removal of a requirement and feeding an input have equal effect:

Lemma 4.21. *For all $\sigma, \tau \in \pm_P^\infty$, it holds that:* $\mathsf{del}(\sigma) \circ \tau = \sigma \circ (+\tau)$.

Proof. We show that $R = \{\langle \mathsf{del}(\sigma) \circ \tau, \sigma \circ (+\tau) \rangle\} \cup \{\langle \sigma, \sigma \rangle\}$ is a bisimulation, by case analysis of σ. For $\sigma = \bot$, and $\sigma = -\sigma'$, $\langle \mathsf{del}(\sigma) \circ \tau, \sigma \circ (+\tau) \rangle \in R$ follows by reflexivity. If $\sigma = +\sigma'$, then $\mathsf{del}(\sigma) \circ \tau = +(\mathsf{del}(\sigma') \circ \tau)$, and $\sigma \circ (+\tau) = +(\sigma' \circ (+\tau))$. Hence, $\langle \mathsf{tail}(\mathsf{del}(\sigma) \circ \tau), \mathsf{tail}(\sigma \circ (+\tau)) \rangle \in R$.   □

The following proposition states that $\mathsf{fix}(\sigma)$ is a fixed point of $[\![\sigma]\!]$:

Proposition 4.22. *For all $\sigma \in \pm_P^\infty$, it holds that:* $[\![\sigma]\!](\mathsf{fix}(\sigma)) = \mathsf{fix}(\sigma)$.

Proof. We prove $\mathsf{fix}(\sigma) = \sigma \circ \mathsf{fix}(\sigma)$ by case analysis and coinduction.

If $\sigma = \bot$, then $\mathsf{fix}(\sigma) = \bot = \sigma \circ \mathsf{fix}(\sigma)$.

If $\sigma = -\sigma'$, then $\mathsf{fix}(\sigma) = \bot$, and $\sigma \circ \mathsf{fix}(\sigma) = (-\sigma') \circ \bot = \bot$.

If $\sigma = +\sigma'$, then $\mathsf{fix}(\sigma) = +\mathsf{fix}(\mathsf{del}(\sigma'))$ and $\sigma \circ \mathsf{fix}(\sigma) = +(\sigma' \circ (+\mathsf{fix}(\mathsf{del}(\sigma'))))$, and we have to prove that $\mathsf{fix}(\mathsf{del}(\sigma')) = \sigma' \circ (+\mathsf{fix}(\mathsf{del}(\sigma')))$ which follows by an instance of Lem. 4.21: $\sigma' \circ (+\mathsf{fix}(\mathsf{del}(\sigma'))) = \mathsf{del}(\sigma') \circ \mathsf{fix}(\mathsf{del}(\sigma'))$.   □

Lemma 4.23. *For all $\sigma \in \pm^\infty$, it holds that:* $\mathrm{lfp}([\![\sigma]\!]) = \mathsf{fix}(\sigma)$.

Proof.

4.2   Computing Production

We introduce a term syntax for the production calculus and rewrite rules for evaluating closed terms.

Definition 4.24. Let $\mathcal{X}$ be a set of recursion variables. The set of *production terms* $\mathcal{P}$ is generated by:

$$p ::= \underline{k} \mid x \mid \bullet(p) \mid \underline{f}(p) \mid \mu x.p \mid \mathsf{min}(p, p)$$

where $x \in \mathcal{X}$, for $k \in \overline{\mathbb{N}}$, the symbol $\underline{k}$ is a *numeral* (a term representation) for k, and, for a unary p-i function $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, $\underline{f} \in \mathcal{I}$, the io-term representing f. For every finite set $P = \{p_1, \ldots, p_n\} \subseteq \mathcal{P}$, we use $\mathsf{min}(p_1, \ldots, p_n)$ and $\mathsf{min}\, P$ as shorthands for the production term $\mathsf{min}(p_1, \mathsf{min}(p_2, \ldots, \mathsf{min}(p_{n-1}, p_n)))$.

The 'production' $[\![p]\!] \in \overline{\mathbb{N}}$ of a closed production term $p \in \mathcal{P}$ is defined by induction on the term structure, interpreting μ as the least fixed point operator, $\underline{f}$ as f, $\underline{k}$ as k, and min as min, as follows.

Definition 4.25. The *production* $[\![p]\!]^\alpha \in \overline{\mathbb{N}}$ of a term $p \in \mathcal{P}$ with respect to an *assignment* $\alpha : \mathcal{X} \to \overline{\mathbb{N}}$ is defined inductively by:

$$[\![\underline{k}]\!]^\alpha = k \qquad\qquad [\![\iota(p)]\!]^\alpha = [\![\iota]\!]([\![p]\!]^\alpha)$$
$$[\![x]\!]^\alpha = \alpha(x) \qquad\qquad [\![\mu x.p]\!]^\alpha = \mathrm{lfp}(\lambda n.[\![p]\!]^{\alpha[x \mapsto n]})$$
$$[\![\bullet(p)]\!]^\alpha = 1 + [\![p]\!]^\alpha \qquad [\![\mathsf{min}(p_1, p_2)]\!]^\alpha = \min([\![p_1]\!]^\alpha, [\![p_2]\!]^\alpha)$$

where $\alpha[x \mapsto n]$ denotes an *update* of α, defined by $\alpha[x \mapsto n](y) = n$ if $y = x$, and $\alpha[x \mapsto n](y) = \alpha(y)$ otherwise.

Finally, we let $[\![p]\!] := [\![p]\!]^{\alpha_0}$ with α_0 defined by $\alpha_0(x) = 0$ for all $x \in \mathcal{X}$.

As becomes clear from Def. 4.25, we could have done without the clause $\bullet(p)$ in the BNF grammar for production terms in Def. 4.24, as $\bullet$ can be abbreviated to $+\overline{-}+$, an io-term that denotes the successor function. However, we take it up as a primitive constructor here in order to match with pebbleflow nets, see Sec. 4.3.

For faithfully modelling the d-o lower bounds of stream functions with stream arity r, we employ r-ary p-i functions, which we represent by 'r-ary gates'.

Definition 4.26. An *r-ary gate* $\mathsf{gate}_k(\sigma_1, \ldots, \sigma_r)$ is defined as a production term context of the form:

$$\mathsf{gate}_k(\sigma_1, \ldots, \sigma_r) := \mathsf{min}(\underline{k}, \sigma_1(\square_1), \ldots, \sigma_r(\square_r)) \,,$$

where $k \in \overline{\mathbb{N}}$ and $\sigma_1, \ldots, \sigma_r \in \pm\infty$. We use γ as a syntactic variable for gates. The *interpretation* of a gate $\gamma = \mathsf{gate}_k(\sigma_1, \ldots, \sigma_r)$ is defined by:

$$[\![\gamma]\!](n_1, \ldots, n_r) := \min(k, [\![\sigma_1]\!](n_1), \ldots, [\![\sigma_r]\!](n_r)) \,.$$

In case $k = \infty$, we simplify $\mathsf{gate}_k(\sigma_1, \ldots, \sigma_r)$ to

$$\mathsf{gate}(\sigma_1, \ldots, \sigma_r) := \mathsf{min}(\sigma_1(\square_1), \ldots, \sigma_r(\square_r)) \,.$$

It is possible to choose unique gate representations $\underline{f}$ of p-i functions f that are efficiently computable from other gate representations, see Section 4.3.

Owing to the restriction to (term representations of) periodically increasing functions in Def. 4.25 it is possible to calculate the production $[\![p]\!]$ of terms $p \in \mathcal{P}$. For that purpose, we define a rewrite system which reduces any closed term to a numeral $\underline{k}$. This system makes use of the computable operations $\circ$ and fix on io-terms mentioned above.

Definition 4.27. The *rewrite relation* $\rightarrow_{\mathsf{R}}$ *on production terms* is defined as the compatible closure of the following rules:

$$\bullet(p) \rightarrow +\overline{-+}(p) \tag{R1}$$

$$\iota_1(\iota_2(p)) \rightarrow \iota_1 \circ \iota_2(p) \tag{R2}$$

$$\iota(\mathsf{min}(p_1, p_2)) \rightarrow \mathsf{min}(\iota(p_1), \iota(p_2)) \tag{R3}$$

$$\mu x.\mathsf{min}(p_1, p_2) \rightarrow \mathsf{min}(\mu x.p_1, \mu x.p_2) \tag{R4}$$

$$\mu x.p \rightarrow p \qquad \text{if } x \notin \mathsf{FV}(p) \tag{R5}$$

$$\mu x.\iota(x) \rightarrow \underline{\mathsf{fix}(\iota)} \tag{R6}$$

$$\mathsf{min}(\underline{k_1}, \underline{k_2}) \rightarrow \underline{\mathsf{min}(k_1, k_2)} \tag{R7}$$

$$\iota(\underline{k}) \rightarrow \underline{[\![\iota]\!](k)} \tag{R8}$$

$$\mu x.x \rightarrow \underline{0} \tag{R9}$$

The following two lemmas establish the usefulness of the rewrite relation $\rightarrow_{\mathsf{R}}$. In order to compute the production $[\![p]\!]$ of a production term p it suffices to obtain a $\rightarrow_{\mathsf{R}}$-normal form of p.

Lemma 4.28. *The rewrite relation* $\rightarrow_{\mathsf{R}}$ *is production preserving:*

$$p \rightarrow_{\mathsf{R}} p' \implies [\![p]\!] = [\![p']\!] .$$

Proof. It suffices to prove: $C[\ell^\sigma] \rightarrow_{\mathsf{R}} C[r^\sigma] \implies \forall \alpha. [\![C[\ell^\sigma]]\!]^\alpha = [\![C[r^\sigma]]\!]^\alpha$, where $\ell \rightarrow r$ is a rule of the TRS given in Def. 4.27, and C a unary context over $\mathcal{P}$. We proceed by induction on C. For the base case, $C = [\,]$, we give the essential proof steps only: For rule (R1), observe that $[\![\overline{-+}]\!]$ is the identity function on $\overline{\mathbb{N}}$. For rule (R2), we apply Prop. 4.11. For rule (R3) the desired equality follows from C_1 on page 31. For rule (R4) we conclude by C_2 ibid. For rule (R6) we use Lem. 4.23. For the remaining rules the statement trivially holds. For the induction step, the statement easily follows from the induction hypotheses. $\square$

Lemma 4.29. *The rewrite relation* $\rightarrow_{\mathsf{R}}$ *is terminating and confluent, and every closed* $p \in \mathcal{P}$ *has a numeral* $\underline{k}$ *as its unique* $\rightarrow_{\mathsf{R}}$-*normal form.*

Proof. To see that $\rightarrow_{\mathsf{R}}$ is terminating, let $\mathrm{w} : \mathcal{P} \rightarrow \mathbb{N}$ be defined by:

$$\mathrm{w}(x) = 1 \quad \mathrm{w}(\bullet(p)) = 2 \cdot \mathrm{w}(p) + 1 \qquad \mathrm{w}(\mu x.p) = 2 \cdot \mathrm{w}(p)$$

$$\mathrm{w}(\underline{k}) = 1 \quad \mathrm{w}(\sigma(p)) = 2 \cdot \mathrm{w}(p) \qquad \mathrm{w}(\mathsf{min}(p_1, p_2)) = \mathrm{w}(p_1) + \mathrm{w}(p_2) + 1 ,$$

and observe that $p \to_R q$ implies $w(p) > w(q)$.

Some of the rules of $\to_R$ overlap; e.g. rule (R2) with itself. For each of the five critical pairs we can find a common reduct (the critical pair $\langle \sigma \circ (\tau \circ \upsilon), (\sigma \circ \tau) \circ \upsilon \rangle$ due to an (R2)/(R2)-overlap can be joined by Lem. 4.9), and hence $\to_R$ is locally confluent, by the Critical Pairs Lemma (cf. [10]). By Newman's Lemma, we obtain that $\to_R$ is confluent. Thus normal forms are unique.

To show that every closed net normalises to a source, let p be an arbitrary normal form. Note that the set of free variables of a net is closed under $\to_R$, and hence p is a closed net. Clearly, p does not contain pebbles, otherwise (R1) would be applicable. To see that p contains no subterms of the form $\mu x.q$, suppose it does and consider the innermost such subterm, viz. q contains no μ. If $q \equiv \underline{k}$ or $q \equiv x$, then (R5), resp. (R9) is applicable. If $q \equiv \sigma(q')$, we further distinguish four cases: if $q' \equiv \underline{k}$ or $q' \equiv x$, then (R8) resp. (R6) is applicable; if the root symbol of q' is one of box, min, then q constitutes a redex w.r.t. (R2), (R3), respectively. If $q \equiv \min(q_1, q_2)$, we have a redex w.r.t. (R4). Thus, there are no subterms $\mu x.q$ in p, and therefore, because p is closed, also no variables x. To see that p has no subterms of the form $\sigma(q)$, suppose it does and consider the innermost such subterm. Then, if $q \equiv \underline{k}$ or $q \equiv \min(q_1, q_2)$ then (R8) resp. (R3) is applicable; other cases have been excluded above. Finally, p does not contain subterms of the form $\min(p_1, p_2)$. For if it does, consider the innermost occurrence and note that, since the other cases have been excluded already, p_1 and p_2 have to be sources, and so we have a redex w.r.t. (R7). We conclude that $p \equiv \underline{k}$ for some $k \in \overline{\mathbb{N}}$.   □

4.3   Pebbleflow Nets

Production terms can be visualized by 'pebbleflow nets' introduced in [2], and serve as a means to model the 'data-oblivious' consumption/production behaviour of stream specifications. The idea is to abstract from the actual stream elements (data) in a stream term in favour of occurrences of the symbol •, which we call 'pebble'. Thus, a stream term $u : s$ is translated to $[u : s] = \bullet([s])$, see Section 5.

We give an operational description of pebbleflow nets, and define the production of a net as the number of pebbles a net is able to produce at its output port. Then we prove that this definition of production coincides with Def. 4.25.

Pebbleflow nets are networks built of pebble processing units (fans, boxes, meets, sources) connected by wires. We use the term syntax given in Def. 4.24 for nets and the rules governing the flow of pebbles through a net, and then give an operational meaning of the units a net is built of.

Definition 4.30. The *pebbleflow rewrite relation* $\to_{\mathsf{P}}$ is defined as the compatible closure of the union of the following rules:

$$\mathsf{min}(\bullet(p_1), \bullet(p_2)) \to \bullet(\mathsf{min}(p_1, p_2)) \tag{P1}$$

$$\mu x.\bullet(p(x)) \to \bullet(\mu x.p(\bullet(x))) \tag{P2}$$

$$+\sigma(p) \to \bullet(\sigma(p)) \tag{P3}$$

$$-\sigma(\bullet(p)) \to \sigma(p) \tag{P4}$$

$$\underline{1+k} \to \bullet(\underline{k}) \tag{P5}$$

Wires are unidirectional FIFO communication channels. They are idealised in the sense that there is no upper bound on the number of pebbles they can store; arbitrarily long queues are allowed. Wires have no counterpart on the term level; in this sense they are akin to the edges of a term tree. Wires connect *boxes*, *meets*, *fans*, and *sources*, that we describe next.

A *meet* is waiting for a pebble at each of its input ports and only then produces one pebble at its output port, see Fig. 9. Put differently, the number of pebbles a meet produces equals the minimum of the numbers of pebbles available at each of its input ports. Meets enable explicit branching; they are used to model stream functions of arity > 1, as will be explained below. A meet with an arbitrary number $n \geq 1$ of input ports is implemented by using a single wire in case $n = 1$, and if $n = k + 1$ with $k \geq 1$, by connecting the output port of a 'k-ary meet' to one of the input ports of a (binary) meet.
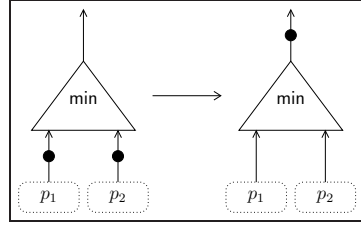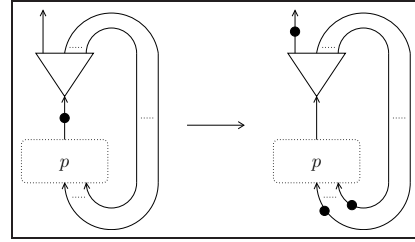

Fig. 9: Rule (P1).

Fig. 10: Rule (P2).

The behaviour of a *fan* is dual to that of a meet: a pebble at its input port is duplicated along its output ports. A fan can be seen as an explicit sharing device, and thus enables the construction of cyclic nets. More specifically, we use fans only to implement feedback when drawing nets; there is no explicit term representation for the fan in our term calculus. In Fig. 10 a pebble is sent over the output wire of the net and at the same time is fed back to the 'recursion wire(s)'. Turning a cyclic net into a term (tree) means to introduce a notion of binding; certain nodes need to be labelled by a name (μx) so that a wire pointing

to that node is replaced by a name (x) *referring* to the labelled node. In rule (P2) feedback is accomplished by substituting $\bullet(x)$ for all free occurrences x of p.

A *source* has an output port only, contains a number $k \in \overline{\mathbb{N}}$ of pebbles, and can fire if $k > 0$, see Fig. 13. The rewrite relation $\to_{\mathsf{R}}$ given in Def. 4.27 collapses any closed net (without input ports that is) into a source.

A *box* consumes pebbles at its input port and produces pebbles at its output port, controlled by an io-sequence $\sigma \in \pm^\infty$ associated with the box. For example, consider the unary stream function dup, defined as follows, and its corresponding io-sequence:

$$\mathsf{dup}(x : \sigma) = x : x : \mathsf{dup}(\sigma) \qquad \overline{-++}$$

which is to be thought of as: *for* dup *to produce two outputs, it first has to consume one input, and this process repeats indefinitely.* Intuitively, the symbol $-$ represents a requirement for an input pebble, and $+$ represents a ready state for an output pebble. Pebbleflow through boxes is visualised in Figs. 11 and 12.
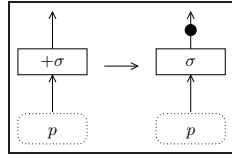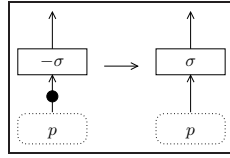

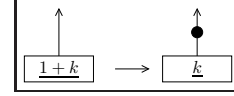Fig. 11: Rule (P3).   Fig. 12: Rule (P4).   Fig. 13: Rule (P5).

The data-oblivious production behaviour of stream functions f with a stream arity $ar_{\mathsf{s}}(\mathsf{f}) = r$ are modelled by r-ary gates (defined in Def. 4.26) that express the contribution of each individual stream argument to the total production of f, see Fig. 14. The precise translation of stream functions into gates is given in Sec. 5, in particular in Def. 5.9.
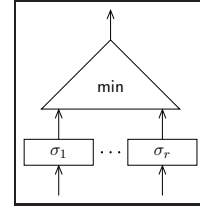
Fig. 14: $\mathsf{gate}(\sigma_1, \ldots, \sigma_r)$

Lemma 4.31. *The pebbleflow rewrite relation $\to_{\mathsf{P}}$ is confluent.*

Proof. The rules of $\to_{\mathsf{P}}$ can be viewed as a higher-order rewriting system (HRS) that is orthogonal. Applying Thm. 11.6.9 in [10] then establishes the lemma. □

Definition 4.32. The *production function* $\Pi : \mathcal{P} \to \overline{\mathbb{N}}$ *of nets* is defined by:

$$\Pi(p) := \sup \left\{ n \in \mathbb{N} \mid p \twoheadrightarrow_{\mathsf{P}} \bullet^n(p') \right\},$$

for all $p \in \mathcal{P}$. $\Pi(p)$ is called *the production of p*. Moreover, for a net p and an assignment $\alpha \in \mathcal{X} \to \overline{\mathbb{N}}$, let $\Pi(p, \alpha) := \Pi(p^\alpha)$ where p^α denotes the net obtained by replacing each free variable x of p with $\bullet^{\alpha(x)}(x)$.

Note that for closed nets p (where $\mathsf{FV}(p) = \varnothing$), $p^\alpha = p$ and therefore $[\![p]\!]^\alpha = [\![p]\!]$, for all assignments α.

An important property used in the following lemma is that functions of the form $\lambda n \in \overline{\mathbb{N}}.\ \Pi(p, \alpha[x \mapsto n])$ are monotonic functions over $\overline{\mathbb{N}}$. Every monotonic function $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ in the complete chain $\overline{\mathbb{N}}$ has a least fixed point $\mathrm{lfp}(f)$ which can be computed by $\mathrm{lfp}(f) = \lim_{n \to \infty} f^n(0)$. In what follows we employ, for monotonic $f, g : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$, two basic properties:

$$\forall n, m.\ \big(f(\min(n, m)) = \min(f(n), f(m))\big) \qquad (C_1)$$

$$\mathrm{lfp}(\lambda n.\ \min(f(n), g(n))) = \min(\mathrm{lfp}(f), \mathrm{lfp}(g)) \qquad (C_2)$$

Lemma 4.33. *For all nets $q \in \mathcal{P}$ and all assignments α, we have that $\Pi(\mu x.q, \alpha)$ is the least fixed point of $\lambda n \in \overline{\mathbb{N}}.\ \Pi(q, \alpha[x \mapsto n])$.*

Proof. Let $\alpha : \mathcal{X} \to \overline{\mathbb{N}}$ be an arbitrary assignment and $q_0 := q^{\alpha[x \mapsto 0]}$. Observe that $\Pi(\mu x.q, \alpha) = \Pi(\mu x.q_0)$ and consider a rewrite sequence of the form

$$\mu x.q_0 \to^* \ldots \to^* \bullet^{n_i}(\mu x.q_i) \to^* \bullet^{n_i}(\mu x.\bullet^{\ell_i}(q_i')) \to^* \bullet^{n_i + \ell_i}(\mu x.q_{i+1}) \to^* \ldots$$

where $\ell_i = \Pi(q_i)$, $n_0 = 0$, $n_{i+1} = n_i + \ell_i$, and $q_{i+1} := q_i'(\bullet^{\ell_i}(x))$. Note that $\lim_{m \to \infty} n_m = \Pi(\mu x.q_0)$; '$\leq$' follows from $\forall m.\ \mu x.q_0 \to^* \bullet^{n_m}(\mu x.q_m)$, and '$\geq$' since if $\lim_{m \to \infty} n_m < \infty$ then $\exists m \in \mathbb{N}$ such that $\ell_m := \Pi(q_m) = 0$ and therefore $\Pi(\mu x.q_0) = \Pi(\bullet^{n_m}(\mu x.q_m)) = n_m$ by confluence.

Let $f_i = \lambda n.[\![q_i(\bullet^n(x))]\!]$, and $f_i' = \lambda n.[\![q_i'(\bullet^n(x))]\!]$. We prove

$$\forall k \in \mathbb{N}.\ f_0(n_m + k) = n_m + f_m(k) \qquad (*)$$

by induction over m. The base case $m = 0$ is trivial, we consider the induction step. We have $q_m \to^* \bullet^{\ell_m}(q_m')$ and by substituting $\bullet^k(x)$ for x we get

$$\forall k \in \mathbb{N}.\ f_m(k) = \ell_m + f_m'(k) \qquad (**)$$

Moreover, since $f_{m+1}(k) = f_m'(\ell_m + k)$, we get $n_{m+1} + f_{m+1}(k) = n_{m+1} + f_m'(\ell_m + k) = n_m + \ell_m + f_m'(\ell_m + k) \overset{(**)}{=} n_m + f_m(\ell_m + k) \overset{(*)}{=} f_0(n_m + \ell_m + k) = f_0(n_{m+1} + k)$.

Let $f := f_0$. We proceed with showing $\forall m.\ f^m(0) = n_m$ by induction over $m \in \mathbb{N}$. For the base case $m = 0$ we have $f^0(0) = 0$ and $n_0 = 0$, and for the induction step we get $f^{m+1}(0) = f(f^m(0)) \overset{\mathrm{IH}}{=} f(n_m) \overset{(*)}{=} n_m + f_m(0) = n_m + \ell_m = n_{m+1}$.

Hence $\mathrm{lfp}(f) = \lim_{m \to \infty} f^m(0) = \lim_{m \to \infty} n_m = \Pi(\mu x.q_0) = \Pi(\mu x.q, \alpha)$. $\qquad \square$

Lemma 4.34. *For $p \in \mathcal{P}$, $\sigma \in \pm_\mathrm{P}^\infty$, $\alpha : \mathcal{X} \to \overline{\mathbb{N}}$: $\Pi(\sigma(p), \alpha) = [\![\sigma]\!](\Pi(p, \alpha))$.*

Proof. We show that the relation $R \subseteq \overline{\mathbb{N}} \times \overline{\mathbb{N}}$ defined as follows is a bisimulation:

$$R := \big\{\ \langle \Pi(\sigma(p), \alpha), [\![\sigma]\!](\Pi(p, \alpha)) \rangle \mid \sigma \in \pm_\mathrm{P}^\infty,\ p \in \mathcal{P},\ \alpha : \mathcal{X} \to \overline{\mathbb{N}}\ \big\},$$

that is, we prove that, for all $k_1, k_2 \in \overline{\mathbb{N}}$, $\sigma \in \pm_{\mathsf{P}}^{\infty}$, $p \in \mathcal{P}$, and $\alpha : \mathcal{X} \to \overline{\mathbb{N}}$, if $k_1 = \Pi(\sigma(p), \alpha)$ and $k_2 = [\![\sigma]\!](\Pi(p, \alpha))$, then either $k_1 = k_2 = 0$ or $k_1 = 1 + k_1'$, $k_2 = 1 + k_2'$ and $\langle k_1', k_2' \rangle \in R$.

Let $k_1, k_2 \in \overline{\mathbb{N}}$, $\sigma \in \pm_{\mathsf{P}}^{\infty}$, $p \in \mathcal{P}$, and $\alpha : \mathcal{X} \to \overline{\mathbb{N}}$, be such that $k_1 = \Pi(\sigma(p), \alpha)$ and $k_2 = [\![\sigma]\!](\Pi(p, \alpha))$. By definition of $\pm^{\infty}$, we have that $\sigma \equiv -^n + \tau$ for some $n \in \mathbb{N}$ and $\tau \in \pm^{\infty}$. We proceed by induction on n. If $n = 0$, then $k_1 = 1 + k_1'$ with $k_1' = \Pi(\tau(p), \alpha)$ and $k_2 = 1 + k_2'$ with $k_2' = [\![\tau]\!](\Pi(p, \alpha))$, and $\langle k_1', k_2' \rangle \in R$. If $n = n' + 1$, we distinguish cases: If $\Pi(p, \alpha) = 0$, then $k_1 = k_2 = 0$. If $\Pi(p, \alpha) = 1 + m$, then $p \twoheadrightarrow_{\mathsf{P}} \bullet(q)$ for some $q \in \mathcal{P}$ with $\Pi(q, \alpha) = m$. Thus we get $k_1 = \Pi(-^{n'} + \tau(q), \alpha)$ and $k_2 = [\![-^{n'} + \tau]\!](\Pi(q, \alpha))$, and $\langle k_1, k_2 \rangle \in R$ by induction hypothesis. $\qquad\square$

Next we show that production of a term $p \in \mathcal{P}$ (Def. 4.25) coincides with the maximal number of pebbles produced by the net p via the rewrite relation $\to_{\mathsf{P}}$ (Def. 4.32).

Lemma 4.35. *For all nets p and assignments α, it holds $\Pi(p, \alpha) = [\![p]\!]^{\alpha}$.*

Proof. The statement of the lemma can be proved by a straightforward induction on the number of μ-bindings of a net p, with a subinduction on the size of p. In the cases $p \equiv \sigma(p')$ and $p \equiv \mu x.p'$ Lem. 4.34 and Lem. 4.33 are applied, respectively. $\qquad\square$

Subsequently, we will use the interpretation of terms in $\mathcal{P}$ and the production of pebbleflow nets, $[\![\_]\!]$ and Π, interchangeably.

5   Translation into Production Terms

In this section we define a translation from stream constants in flat or friendly nesting specifications to production terms. In particular, the root M_0 of a specification $\mathcal{T}$ is mapped by the translation to a production term $[\mathsf{M}_0]$ with the property that if $\mathcal{T}$ is flat (friendly nesting), then the d-o lower bound on the production of M_0 in $\mathcal{T}$ equals (is bounded from below by) the production of $[\mathsf{M}_0]$.

5.1   Translation of Flat and Friendly Nesting Symbols

As a first step of the translation, we describe how for a flat (or friendly nesting) stream function symbol f in a stream specification $\mathcal{T}$ a periodically increasing function $\langle \mathsf{f} \rangle$ can be calculated that is (that bounds from below) the d-o lower bound on the production of f in $\mathcal{T}$.

Let us again consider the rules (i) $\mathsf{f}(\mathsf{s}(x) : y : \sigma) \to \mathsf{a}(\mathsf{s}(x), y) : \mathsf{f}(y : \sigma)$, and (ii) $\mathsf{f}(0 : \sigma) \to 0 : \mathsf{s}(0) : \mathsf{f}(\sigma)$ from Fig. 2. We model the d-o lower bound on the production of f by a function from $\overline{\mathbb{N}}$ to $\overline{\mathbb{N}}$ defined as the unique solution for X_{f} of the following system of equations. We disregard what the concrete stream elements are, and therefore we take the infimum over all possible traces:

$$X_{\mathsf{f}}(n) = \inf \left\{ X_{\mathsf{f},(i)}(n), X_{\mathsf{f},(ii)}(n) \right\}$$

where the solutions for $X_{f,(i)}$ and $X_{f,(ii)}$ are the d-o lower bounds of f assuming that the first rule applied in the rewrite sequence is (i) or (ii), respectively. The rule (i) consumes two elements, produces one element and feeds one element back to the recursive call. For rule (ii) these numbers are 1, 2, 0 respectively. Therefore we get:

$$X_{f,(i)}(n) = \text{let } n' := n - 2, \text{ if } n' < 0 \text{ then } 0 \text{ else } 1 + X_f(1 + n'),$$
$$X_{f,(ii)}(n) = \text{let } n' := n - 1, \text{ if } n' < 0 \text{ then } 0 \text{ else } 2 + X_f(n').$$

The unique solution for X_f is $n \mapsto n \dot{-} 1$, represented by the io-term $-\overline{-+}$.

In general, functions may have multiple arguments, which during rewriting may get permuted, deleted or duplicated. The idea is to trace single arguments, and to take the infimum over traces in case an argument is duplicated.

In the definition of the translation of stream functions, we need to distinguish the cases according to whether a symbol is weakly guarded or not: On Σ_s we define $\leadsto := \{\langle root(\ell), root(r)\rangle \mid \ell \to r \in R_S, \text{ ':'} \neq root(r) \in \Sigma_s\}$. the *dependency relation* between symbols in Σ_s. We say that a symbol $f \in \Sigma_s$ is *weakly guarded* if f is strongly normalising with respect to $\leadsto$ and *unguarded*, otherwise. Note that since Σ_s is finite it is (easily) decidable whether a symbol $f \in \Sigma_s$ is weakly guarded or unguarded.

The translation of a stream function symbol is defined as the unique solution of a (usually infinite) system of defining equations where the unknowns are functions. More precisely, for each symbol $f \in \Sigma_{fnest} \supseteq \Sigma_{sfun}$ of a flat or friendly nesting stream specification, this system has a p-i function $\langle f\rangle$ as the solution for X_f, which is unique among the continuous functions. Later we will see (Prop. 5.14 and Lem. 5.15) that the translation $\langle f\rangle$ of a flat (friendly nesting) stream function symbol f coincides with (is a lower bound for) the d-o lower bound $\underline{do}_{\mathcal{T}}(f)$ of f.

Definition 5.1. Let $\mathcal{T} = \langle \Sigma, R\rangle$ be a stream specification. For each flat or friendly nesting symbol $f \in \Sigma_{fnest} \supseteq \Sigma_{flat}$ with arities $k = ar_s(f)$ and $\ell = ar_d(f)$ we define $\langle f\rangle : \overline{\mathbb{N}}^k \to \overline{\mathbb{N}}$, called the *(p-i function) translation of f in $\mathcal{T}$*, as the unique solution s_{X_f} for X_f of the following system of defining equations, where the solution of an equation of the form $X(n_1, \ldots, n_k) = \ldots$ is a function $s_X : \overline{\mathbb{N}}^k \to \overline{\mathbb{N}}$ (it is to be understood that $s_X \in \overline{\mathbb{N}}$ if $k = 0$): For all $n_1, \ldots, n_k \in \overline{\mathbb{N}}$, $i \in \{\star, 1, \ldots, k\}$, and $n \in \mathbb{N}$:

$$X_f(n_1, \ldots, n_k) = \min(X_{f,\star}, X_{f,1}(n_1), \ldots, X_{f,k}(n_k)),$$

$$X_{f,\star} = \begin{cases} \inf \left\{X_{f,\star,\rho} \mid \rho \text{ a defining rule of } f\right\} & \text{if } f \text{ is weakly guarded,} \\ 0 & \text{if } f \text{ is unguarded,} \end{cases}$$

$$X_{f,i}(n) = \begin{cases} \inf \left\{X_{f,i,\rho}(n) \mid \rho \text{ a defining rule of } f\right\} & \text{if } f \text{ is weakly guarded,} \\ 0 & \text{if } f \text{ is unguarded.} \end{cases}$$

We write $\boldsymbol{u}_i : \sigma_i$ for $u_{i,1} : \ldots : u_{i,p} : \sigma_i$, and $|\boldsymbol{u}_i|$ for p. For specifying $X_{f,i,\rho}$ we distinguish the possible forms the rule ρ can have. If ρ is nesting, then $X_{f,\star,\rho} = \infty$,

and $X_{f,i,\rho}(n) = n$ for all $n \in \overline{\mathbb{N}}$. Otherwise, ρ is non-nesting and of the form:

$$f((\boldsymbol{u}_1 : \sigma_1), \ldots, (\boldsymbol{u}_k : \sigma_k), v_1, \ldots, v_\ell) \to w_1 : \ldots : w_m : s \,,$$

where either (a) $s \equiv \sigma_j$, or (b) $s \equiv g((\boldsymbol{u'}_1 : \sigma_{\phi(1)}), \ldots, (\boldsymbol{u'}_{k'} : \sigma_{\phi(k')}), v'_1, \ldots, v'_{\ell'})$
with $k' = ar_s(g)$, $\ell' = ar_d(g)$, and $\phi : \{1, \ldots, k'\} \to \{1, \ldots, k\}$. Let:

$$X_{f,\star,\rho} = \begin{cases} \infty & \text{case (a)} \\ m + X_{g,\star} & \text{case (b)} \end{cases}$$

$$X_{f,i,\rho}(n) = \text{let } n' := n - |\boldsymbol{u}_i|, \text{ if } n' < 0 \text{ then } 0 \text{ else}$$

$$m + \begin{cases} n' & \text{case (a), } i = j \\ \infty & \text{case (a), } i \neq j \\ \inf \left\{ X_{g,j}(n' + |\boldsymbol{u'}_j|) \mid j \in \phi^{-1}(i) \right\} & \text{case (b)} \end{cases}$$

where we agree $\inf \varnothing = \infty$.

We mention a useful intuition for understanding the fabric of the formal definition above of the translation $\langle f \rangle$ of a stream function f, the solution s_{X_f} for X_f of the system of equations in Def. 5.1. For each $i \in \{1, \ldots, k\}$ where $k = ar_s(f)$ the solution $s_{X_{f,i}}$ for $X_{f,i}$ (a unary p-i function) in this system describes to what extent consumption from the i-th component of f 'delays' the overal production. Since in a d-o rewrite sequence from $f(\bullet^{n_1} : \sigma_1, \ldots, \bullet^{n_k} : \sigma_k)$ it may happen that all stream variables $\sigma_1, \ldots, \sigma_k$ are erased at some point, and that the sequence subsequently continues rewriting stream constants, monitoring the delays caused by individual arguments is not sufficient alone to define the production function of f. This is the reason for the use, in the definition of $\langle f \rangle$, of the solution $s_{X_{f,\star}}$ for the variable $X_{f,\star}$, which defines a 'glass ceiling' for the not adequate 'overall delay' function $\min(s_{X_{f,1}(n_1)}, \ldots, s_{X_{f,k}(n_k)})$, taking account of situations in which in d-o rewrite sequences from terms $f(\bullet^{n_1} : \sigma_1, \ldots, \bullet^{n_k} : \sigma_k)$ all input components have been erased.

Concerning non-nesting rules on which defining rules for friendly nesting symbols depend via $\rightsquigarrow$, this translation uses the fact that their production is bounded below by 'min'. These bounds are not necessarily optimal, but can be used to show productivity of examples like $X \to 0 : f(X)$ with $f(x : \sigma) \to x : f(f(\sigma))$.

Def. 5.1 can be used to define, for every flat or friendly nesting symbol $f \in \Sigma_{sfun}$ in a stream specification $\mathcal{T}$, a 'gate translation' of f: by defining this translation by choosing a gate that represents the p-i function $\langle f \rangle$.

However, it is desirable to define this translation also in an equivalent way that lends itself better for computation, and where the result directly is a production term (gate) representation of a p-i function: by specifying the io-terms in a gate translation as denotations of rational io-sequences that are the solutions of 'io-sequence specifications'. In particular, the gate translation of a symbol $f \in \Sigma_{sfun}$ will be defined in terms of the solutions, for each argument place of a stream function symbol f, of a finite io-sequence specification that is extracted from an infinite one which precisely determines the d-o lower bound of f in that argument place.

Definition 5.2. Let $\mathcal{X}$ be a set of variables. The set of *io-sequence specification expressions over* $\mathcal{X}$ is defined by the following grammar:

$$E ::= \bot \mid X \mid -E \mid +E \mid E \wedge E$$

where $X \in \mathcal{X}$. Guardedness is defined by induction: An io-sequence specification expression is called *guarded* if it is of one of the forms $\bot$, $-E_0$, or $+E_0$, or if it is of the form $E_1 \wedge E_2$ for guarded E_1 and E_2.

Suppose that $\mathcal{X} = \{X_\alpha \mid \alpha \in A\}$ for some (countable) set A. Then by an *io-sequence specification over (the set of recursion variables)* $\mathcal{X}$ we mean a family $\{X_\alpha = E_\alpha\}_{\alpha \in A}$ of *recursion equations*, where, for all $\alpha \in A$, E_α is an io-sequence specification expression over $\mathcal{X}$. Let $\mathcal{E}$ be an io-sequence specification. If $\mathcal{E}$ consists of finitely many recursion equations, then it is called *finite*. $\mathcal{E}$ is called *guarded* (*weakly guarded*) if the right-hand side of every recursion equation in $\mathcal{E}$ is guarded (or respectively, can be rewritten, using equational logic and the equations of $\mathcal{E}$, to a guarded io-sequence specification expression).

Let $\mathcal{E} = \{X_\alpha = E_\alpha\}_{\alpha \in A}$ an io-sequence specification. Furthermore let $\alpha_0 \in A$ and $\tau \in \pm^\omega$. We say that τ is *a solution of* $\mathcal{E}$ *for* X_{α_0} if there exist io-sequences $\{\sigma_\alpha\}_{\alpha \in A}$ such that $\sigma_{\alpha_0} = \tau$, and all of the recursion equations in $\mathcal{E}$ are true statements (under the interpretation of $\wedge$ as defined in Def. 4.13), when, for all $\alpha \in A$, σ_α is substituted for X_α, respectively.

It turns out that weakly guarded io-sequence specifications have unique solutions, and that the solutions of finite, weakly guarded io-sequence specifications are rational io-sequences.

Lemma 5.3. *For every weakly guarded io-sequence specification* $\mathcal{E}$ *and recursion variable* X *of* $\mathcal{E}$*, there exists a unique solution* $\sigma \in \pm^\infty$ *of* $\mathcal{E}$ *for* X*. Moreover, if* $\mathcal{E}$ *is finite then the solution* σ *of* $\mathcal{E}$ *for* X *is a rational io-sequence, and an io-term that denotes* σ *can be computed on the input of* $\mathcal{E}$ *and* X*.*

Let $\mathcal{T}$ be a stream specification, and $\mathsf{f} \in \Sigma_{sfun}$. By exhaustiveness of $\mathcal{T}$ for f, there is at least one defining rule for f in $\mathcal{T}$. Since $\mathcal{T}$ is a constructor stream TRS, it follows that every defining rule ρ for f is of the form:

$$\mathsf{f}(p_1, \ldots, p_{ar_s(\mathsf{f})}, q_1, \ldots, q_{ar_d(\mathsf{f})}) \rightarrow u_1 : \ldots : u_m : s \qquad (\rho)$$

with $p_1, \ldots, p_{ar_s(\mathsf{f})} \in Ter(\mathcal{C}(\Sigma))_S$, $q_1, \ldots, q_{ar_d(\mathsf{f})} \in Ter(\mathcal{C}(\Sigma))_D$, $u_1, \ldots, u_m \in Ter(\Sigma)_D$ and $s \in Ter(\Sigma)_S$ where $root(s) \neq$ ':'. If ρ is non-nesting then either $s \equiv \sigma_j$ or $s \equiv \mathsf{g}(\boldsymbol{w}_1 : \sigma_{\phi(1)}, \ldots, \boldsymbol{w}_{ar_s(\mathsf{g})} : \sigma_{\phi(ar_s(\mathsf{g}))}, u'_1, \ldots, u'_{ar_d(\mathsf{g})})$ where $\boldsymbol{w}_i : \sigma_i$ is shorthand for $w_{i,1} : \ldots : w_{i,m_i} : \sigma_i$, and $\phi : \{1, \ldots, ar_s(\mathsf{g})\} \rightarrow \{1, \ldots, ar_s(\mathsf{f})\}$ is a function that describes how the stream arguments are permuted and replicated.

We now define, for every given stream specification $\mathcal{T}$, an infinite io-sequence specification $\mathcal{E}_{\mathcal{T}}$ that will be instrumental for defining gate translations of the stream function symbols in $\mathcal{T}$.

Definition 5.4. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification. Based on the set:

$$A := \{\langle \epsilon, - \rangle, \langle \epsilon, + \rangle, \langle \epsilon, -+ \rangle\}$$
$$\cup \left\{ \begin{matrix} \langle \mathsf{f}, \star \rangle, \langle \mathsf{f}, \star, \rho \rangle, \\ \langle \mathsf{f}, i, q \rangle, \langle \mathsf{f}, i, q, \rho \rangle \end{matrix} \; \middle| \; \begin{matrix} \mathsf{f} \in \Sigma_{fnest}, \, q \in \mathbb{N}, \, \rho \text{ defining rule for } \mathsf{f}, \\ 1 \leq i \leq ar_s(\mathsf{f}) \end{matrix} \right\}$$

of tuples we define the infinite io-sequence specification $\mathcal{E}_\mathcal{T} = \{X_\alpha = E_\alpha\}_{\alpha \in A}$ by listing the equations of $\mathcal{E}_\mathcal{T}$. We start with the equations

$$X_{\langle \epsilon, - \rangle} = \bot \,, \qquad X_{\langle \epsilon, + \rangle} = +X_{\langle \epsilon, + \rangle} \,, \qquad X_{\langle \epsilon, -+ \rangle} = -+X_{\langle \epsilon, -+ \rangle} \,.$$

Then we let, for all friendly nesting (or flat) $\mathsf{f} \in \Sigma_{sfun}$ with arities $k = ar_s(\mathsf{f})$ and $\ell = ar_d(\mathsf{f})$:

$$X_{\langle \mathsf{f}, \star \rangle} = \begin{cases} \min\{X_{\langle \mathsf{f}, \star, \rho \rangle} \mid \rho \text{ a defining rule of } \mathsf{f}\} & \text{if } \mathsf{f} \text{ is weakly guarded,} \\ X_{\langle \epsilon, - \rangle} & \text{if } \mathsf{f} \text{ is unguarded,} \end{cases}$$

and for all $1 \leq i \leq ar_s(\mathsf{f})$, and $q \in \mathbb{N}$:

$$X_{\langle \mathsf{f}, i, q \rangle} = \begin{cases} \bigwedge\{X_{\langle \mathsf{f}, i, q, \rho \rangle} \mid \rho \text{ a defining rule of } \mathsf{f}\} & \text{if } \mathsf{f} \text{ is weakly guarded,} \\ X_{\langle \epsilon, - \rangle} & \text{if } \mathsf{f} \text{ is unguarded.} \end{cases}$$

For specifying $X_{\langle \mathsf{f}, \star, \rho \rangle}$ and $X_{\langle \mathsf{f}, i, q, \rho \rangle}$ we distinguish the possible forms the rule ρ can have. In doing so, we abbreviate terms $u_{i,1} : \ldots : u_{i,p} : \sigma_i$ with σ_i a variable of sort stream by $\boldsymbol{u}_i : \sigma_i$ and let $|\boldsymbol{u}_i| := p$. If ρ is nesting, then we let

$$X_{\langle \mathsf{f}, \star, \rho \rangle} = X_{\langle \epsilon, + \rangle} \,, \qquad\qquad X_{\langle \mathsf{f}, i, q, \rho \rangle} = X_{\langle \epsilon, -+ \rangle} \,.$$

Otherwise, ρ is non-nesting and of the form:

$$\mathsf{f}((\boldsymbol{u}_1 : \sigma_1), \ldots, (\boldsymbol{u}_k : \sigma_k), v_1, \ldots, v_\ell) \to w_1 : \ldots : w_m : s \,,$$

where either (a) $s \equiv \sigma_j$, or (b) $s \equiv \mathsf{g}((\boldsymbol{u'}_1 : \sigma_{\phi(1)}), \ldots, (\boldsymbol{u'}_{k'} : \sigma_{\phi(k')}), v'_1, \ldots, v'_{\ell'})$ with $k' = ar_s(\mathsf{g})$, $\ell' = ar_d(\mathsf{g})$, and $\phi : \{1, \ldots, k'\} \to \{1, \ldots, k\}$. Let:

$$X_{\langle \mathsf{f}, \star, \rho \rangle} = \begin{cases} X_{\langle \epsilon, + \rangle} & \text{case (a)} \\ +^m X_{\langle \mathsf{g}, \star \rangle} & \text{case (b)} \end{cases}$$

$$X_{\langle \mathsf{f}, i, q, \rho \rangle} = \text{let } p := |\boldsymbol{u}_i| \dot{-} q, \, q' := q \dot{-} |\boldsymbol{u}_i| \text{ in}$$
$$-^p +^m \begin{cases} +^{q'} X_{\langle \epsilon, -+ \rangle} & \text{case (a), } i = j \\ X_{\langle \epsilon, + \rangle} & \text{case (a), } i \neq j \\ \bigwedge\{X_{\langle \mathsf{g}, j, q' + |\boldsymbol{u'}_j| \rangle} \mid j \in \phi^{-1}(i)\} & \text{case (b)} \end{cases}$$

where we agree $\bigwedge \varnothing := X_{\langle \epsilon, + \rangle}$.

We formally state an easy observation about the system $\mathcal{E}_\mathcal{T}$ defined in Def. 5.9, and an immediate consequence due to Lem. 5.3.

Proposition 5.5. *Let $\mathcal{T}$ be a stream specification. The io-sequence specification $\mathcal{E}_{\mathcal{T}}$ (defined in Def. 5.9) is weakly guarded. As a consequence, $\mathcal{E}_{\mathcal{T}}$ has a unique solution in $\pm^{\infty}$ for every variable X in $\mathcal{E}_{\mathcal{T}}$.*

Another easy observation is that the solutions for the variables $X_{\langle \mathsf{f}, \star \rangle}$ in a system $\mathcal{E}_{\mathcal{T}}$ correspond very directly to numbers in $\overline{\mathbb{N}}$.

Proposition 5.6. *Let $\mathcal{T}$ be a stream specification and $\mathsf{f} \in \Sigma_{sfun}$. The unique solution of $\mathcal{E}_{\mathcal{T}}$ (defined in Def. 5.9) for $X_{\langle \mathsf{f}, \star \rangle}$ is of the form $+^{\omega}$ or $+^{n}\bot$.*

Furthermore observe that $\mathcal{E}_{\mathcal{T}}$ is infinite in case that $\Sigma_{sfun} \neq \emptyset$, and hence typically is infinite. Whereas Lem. 5.3 implies unique solvability of $\mathcal{E}_{\mathcal{T}}$ for individual variables in view of (the first statement in) Prop. 5.5, it will usually not guarantee that these unique solutions are rational io-sequences. Nevertheless it can be shown that all solutions of $\mathcal{E}_{\mathcal{T}}$ are rational io-sequences. For our purposes it will suffice to show this only for the solutions of $\mathcal{E}_{\mathcal{T}}$ for certain of its variables.

We will only be interested in the solutions of $\mathcal{E}_{\mathcal{T}}$ for variables $X_{\langle \mathsf{f}, i, 0 \rangle}$ and $X_{\langle \mathsf{f}, \star \rangle}$. It turns out that from $\mathcal{E}_{\mathcal{T}}$ a finite weakly guarded specification $\mathcal{E}'_{\mathcal{T}}$ can be extracted that, for each of the variables $X_{\langle \mathsf{f}, i, 0 \rangle}$ and $X_{\langle \mathsf{f}, \star \rangle}$, has the same solution as $\mathcal{E}_{\mathcal{T}}$, respectively. Lem. 5.7 below states that, for a stream specification $\mathcal{T}$, the finite io-sequence specification $\mathcal{E}'_{\mathcal{T}}$ can always be obtained algorithmically, which together with Lem. 5.3 implies that the unique solutions of $\mathcal{E}_{\mathcal{T}}$ for the variables $X_{\langle \mathsf{f}, i, 0 \rangle}$ and $X_{\langle \mathsf{f}, \star \rangle}$ are rational io-sequences for which representing io-terms can be computed. As a consequence these solutions, for a stream specification $\mathcal{T}$, of the recursion system $\mathcal{E}_{\mathcal{T}}$, can be viewed to represent p-i functions: the p-i functions that are represented by the io-term denoting the respective solution, a rational io-sequence.

As an example let us consider an io-sequence specification that corresponds to the defining rules for f in Fig. 2:

$$X = -++X \wedge --+Y \qquad\qquad Y = ++X \wedge -+Y \ .$$

The unique solution for X of this system is the rational io-sequence (and respectively, io-term) $--\overline{+}$, which is the translation of f (as mentioned earlier).

Lemma 5.7. *Let $\mathcal{T}$ be a stream specification. There exists an io-sequence specification $\mathcal{E}'_{\mathcal{T}} = \{X_{\alpha} = E'_{\alpha}\}_{\alpha \in A'}$ such that:*

(i) $\mathcal{E}'_{\mathcal{T}}$ is finite and weakly guarded;
(ii) $\{\langle \mathsf{f}, i, 0 \rangle, \langle \mathsf{f}, \star \rangle \mid \mathsf{f} \in \Sigma_{sfun}, i, q \in \mathbb{N}, i \in \{1, \ldots, ar_s(\mathsf{f})\}\} \subseteq A'$;
    for each $\mathsf{f} \in \Sigma_{sfun} \cap \Sigma_{fnest}$ and $i \in \{1, \ldots, ar_s(\mathsf{f})\}$, $\mathcal{E}'_{\mathcal{T}}$ has the same solution for $X_{\langle \mathsf{f}, i, 0 \rangle}$, and respectively for $X_{\langle \mathsf{f}, \star \rangle}$, as the io-sequence specifications $\mathcal{E}_{\mathcal{T}}$ (see Def. 5.4);
(iii) on the input of $\mathcal{T}$, $\mathcal{E}'_{\mathcal{T}}$ can be computed.

Proof (Sketch). An algorithm for obtaining $\mathcal{E}'_{\mathcal{T}}$ from $\mathcal{E}_{\mathcal{T}}$ can be obtained as follows. On the input of $\mathcal{E}_{\mathcal{T}}$, set $\mathcal{E} := \mathcal{E}_{\mathcal{T}}$ and repeat the following step on $\mathcal{E}$ as long as it is applicable:

(RPC) Detect and remove a reachable *non-consuming pseudo-cycle* from the io-sequence specification $\mathcal{E}$: Suppose that, for a function symbol h, for $j, k, l \in \mathbb{N}$, and for a recursion variable $X_{\langle \mathsf{h}, j, k \rangle}$ that is reachable from $X_{\langle \mathsf{f}, i, 0 \rangle}$, we have $X_{\langle \mathsf{h}, j, k \rangle} \xrightarrow{w} X_{\langle \mathsf{h}, j, l \rangle}$ (from the recursion variable $X_{\langle \mathsf{h}, j, k \rangle}$ the variable $X_{\langle \mathsf{h}, j, l \rangle}$ is reachable via a path in the specification on which the finite io-sequence w is encountered as the word formed by consecutive labels), where $k < l$ and w only contains symbols '+'. Then modify $\mathcal{E}$ by setting $X_{\langle \mathsf{h}, j, k \rangle} = X_{\langle \epsilon, + \rangle}$.

It is not difficult to show that a step (RPC) preserves weakly guardedness and the unique solution of $\mathcal{E}_{\mathcal{T}}$, and that, on the input of $\mathcal{E}_{\mathcal{T}}$, the algorithm terminates in finitely many steps, having produced an io-sequence specification $\mathcal{E}'_{\mathcal{T}}$ with $[\mathsf{f}]_i$ as the solution for $X_{\langle \mathsf{f}, i, 0 \rangle}$ and the property that only finitely many recursion variables are reachable in $\mathcal{E}'_{\mathcal{T}}$ from $X_{\langle \mathsf{f}, i, 0 \rangle}$. □

As an immediate consequence of Lem. 5.7,
and of Lem. 5.3 we obtain the following lemma.

Lemma 5.8. *Let $\mathcal{T}$ be a stream specification, and let $\mathsf{f} \in \Sigma_{sfun}$.*

(i) *For each $i \in \{1, \ldots, ar_s(\mathsf{f})\}$ there is precisely one io-sequence σ that solves $\mathcal{E}_{\mathcal{T}}$ for $X_{\langle \mathsf{f}, i, 0 \rangle}$; furthermore, σ is rational. Moreover there exists an algorithm that, on the input of $\mathcal{T}$, f, and i, computes the shortest io-term that denotes the solution of $\mathcal{E}_{\mathcal{T}}$ for $X_{\langle \mathsf{f}, i, 0 \rangle}$.*

(ii) *There is precisely one io-sequence σ that solves $\mathcal{E}_{\mathcal{T}}$ for $X_{\langle \mathsf{f}, \star \rangle}$; this io-sequence is rational. And there is an algorithm that, on the input of $\mathcal{T}$ and f, computes the shortest io-term that denotes the solution of $\mathcal{E}_{\mathcal{T}}$ for $X_{\langle \mathsf{f}, \star \rangle}$.*

Lem. 5.8 guarantees the well-definedness in the definition below of the 'gate translation' for flat or friendly nesting stream functions in a stream specification.

Definition 5.9. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream definition. For each flat or friendly nesting symbol $\mathsf{f} \in \Sigma_{sfun} \cap \Sigma_{fnest}$ with stream arity $k = ar_s(\mathsf{f})$ we define the *gate translation* $[\mathsf{f}]$ *of* f by:

$$[\mathsf{f}] := \mathsf{gate}_{[\mathsf{f}]_\star}([\mathsf{f}]_1, \ldots, [\mathsf{f}]_{ar_s(\mathsf{f})}) \, ,$$

where $[\mathsf{f}]_\star \in \overline{\mathbb{N}}$ is defined by:

$$[\mathsf{f}]_\star := \begin{cases} n & \text{the unique solution of } \mathcal{E}_{\mathcal{T}} \text{ for } X_{\langle \mathsf{f}, \star \rangle} \text{ is } +^n \bot \\ \infty & \text{the unique solution of } \mathcal{E}_{\mathcal{T}} \text{ for } X_{\langle \mathsf{f}, \star \rangle} \text{ is } +^\omega \end{cases}$$

and, for $1 \leq i \leq ar_s(\mathsf{f})$, $[\mathsf{f}]_i$ is the shortest io-term that denotes the unique solution for $X_{\langle \mathsf{f}, i, 0 \rangle}$ of the weakly guarded io-sequence specification $\mathcal{E}_{\mathcal{T}}$ in Def. 5.4.

From Lem. 5.8 we also obtain that, for every stream specification, the function which maps stream function symbols to their gate translations is computable.

Lemma 5.10. *There is an algorithm that, on the input of a stream specification $\mathcal{T}$, and a flat or friendly nesting symbol $\mathsf{f} \in \Sigma_{sfun} \cap \Sigma_{fnest}$, computes the gate translation $[\mathsf{f}]$ of f.*

Example 5.11. Consider a flat stream specification consisting of the rules:

$$\mathsf{f}(x : \sigma) \to x : \mathsf{g}(\sigma, \sigma, \sigma) \, ,$$
$$\mathsf{g}(x : y : \sigma, \tau, \upsilon) \to x : \mathsf{g}(y : \tau, y : \upsilon, y : \sigma) \, .$$

The translation of f is $[\mathsf{f}] = \mathsf{gate}([\mathsf{f}]_1)$, where $[\mathsf{f}]_1$ is the unique solution for $X_{\langle \mathsf{f}, 1, 0 \rangle}$ of the io-sequence specification $\mathcal{E}_{\mathcal{T}}$:

$$X_{\langle \mathsf{f}, 1, 0 \rangle} = -+(X_{\langle \mathsf{g}, 1, 0 \rangle} \wedge X_{\langle \mathsf{g}, 2, 0 \rangle} \wedge X_{\langle \mathsf{g}, 3, 0 \rangle})$$
$$X_{\langle \mathsf{g}, 1, 0 \rangle} = --+X_{\langle \mathsf{g}, 3, 1 \rangle} \quad X_{\langle \mathsf{g}, 1, 1 \rangle} = -+X_{\langle \mathsf{g}, 3, 1 \rangle}$$
$$X_{\langle \mathsf{g}, 1, q \rangle} = +X_{\langle \mathsf{g}, 3, q-1 \rangle} \quad (q \geq 2)$$
$$X_{\langle \mathsf{g}, 2, q \rangle} = +X_{\langle \mathsf{g}, 1, q+1 \rangle} \quad (q \in \mathbb{N})$$
$$X_{\langle \mathsf{g}, 3, q \rangle} = +X_{\langle \mathsf{g}, 2, q+1 \rangle} \quad (q \in \mathbb{N})$$

By the algorithm referred to in Lem. 5.10 this infinite specification can be turned into a finite one. The 'non-consuming pseudocycle' $X_{\langle \mathsf{g}, 3, 1 \rangle} \xrightarrow{+++} X_{\langle \mathsf{g}, 3, 2 \rangle}$ justifies the modification of $\mathcal{E}_{\mathcal{T}}$ by setting $X_{\langle \mathsf{g}, 3, 1 \rangle} = +X_{\langle \mathsf{g}, 3, 1 \rangle}$; likewise we set $X_{\langle \mathsf{g}, 3, 0 \rangle} = +X_{\langle \mathsf{g}, 3, 0 \rangle}$. Furthermore all equations not reachable from $X_{\langle \mathsf{f}, 1, 0 \rangle}$ are removed (garbage collection), and we obtain a finite specification $\mathcal{E}'_{\mathcal{T}}$, which, by Lem. 5.3, has a periodically increasing solution, with io-term-denotation $[\mathsf{f}]_1 = \langle -+--, + \rangle$. The reader may try to calculate the gate corresponding to g, it is: $[\mathsf{g}] = \mathsf{gate}(--\overline{\top}, +-\overline{\top}, \overline{+})$.

Second, consider the flat stream specification with data constructor symbols 0 and 1:

$$\mathsf{f}(0 : \sigma) \to \mathsf{g}(\sigma) \, , \quad \mathsf{f}(1 : x : \sigma) \to x : \mathsf{g}(\sigma) \, ,$$
$$\mathsf{g}(x : y : \sigma) \to x : y : \mathsf{g}(\sigma) \, ,$$

denoted $\rho_{\mathsf{f}0}$, $\rho_{\mathsf{f}1}$, and ρ_{g}, respectively. Then, $[\mathsf{f}]_1$ is the solution for $X_{\langle \mathsf{f}, 1, 0 \rangle}$ of

$$X_{\langle \mathsf{f}, 1, 0 \rangle} = X_{\langle \mathsf{f}, 1, 0, \rho_{\mathsf{f}0} \rangle} \wedge X_{\langle \mathsf{f}, 1, 0, \rho_{\mathsf{f}1} \rangle}$$
$$X_{\langle \mathsf{f}, 1, 0, \rho_{\mathsf{f}0} \rangle} = -X_{\langle \mathsf{g}, 1, 0 \rangle} \quad X_{\langle \mathsf{f}, 1, 0, \rho_{\mathsf{f}1} \rangle} = --+X_{\langle \mathsf{g}, 1, 0 \rangle}$$
$$X_{\langle \mathsf{g}, 1, 0 \rangle} = --++X_{\langle \mathsf{g}, 1, 0 \rangle} \, .$$

Example 5.12. Consider a pure stream specification with the function layer:

$$\mathsf{f}(x : \sigma) \to x : \mathsf{g}(\sigma, \sigma, \sigma) \, ,$$
$$\mathsf{g}(x : y : \sigma, \tau, \upsilon) \to x : \mathsf{g}(y : \tau, y : \upsilon, y : \sigma) \, .$$

The translation of f is $\langle f \rangle$, the unique solution for X_f of the system:

$$X_f(n) = \min(X_{f,\star}(0), X_{f,1}(n))$$
$$X_{f,1}(n) = \text{let } n' := n - 1$$
$$\qquad \text{if } n' < 0 \text{ then } 0 \text{ else } 1 + \inf\left\{X_{g,1}(n'), X_{g,2}(n'), X_{g,3}(n')\right\}$$
$$X_{f,\star}(n) = 1 + X_{g,\star}(0)$$
$$X_{g,1}(n) = \text{let } n' := n - 2, \text{ if } n' < 0 \text{ then } 0 \text{ else } 1 + X_{g,3}(1 + n')$$
$$X_{g,2}(n) = 1 + X_{g,1}(1 + n)$$
$$X_{g,3}(n) = 1 + X_{g,2}(1 + n)$$
$$X_{g,\star}(n) = 1 + X_{f,\star}(0)$$

An algorithm for solving such systems of equations is described in (the proof of) Lemma 5.7; here we solve the system directly. Note that $X_{g,3}(n) = 1 + X_{g,2}(n + 1) = 2 + X_{g,1}(n + 2) = 3 + X_{g,3}(n)$, hence $\forall n \in \mathbb{N}. X_{g,3}(n) = \infty$. Likewise we obtain $X_{g,2}(n) = \infty$ if $n \geq 1$ and 1 for $n = 0$, and $X_{g,1}(n) = \infty$ if $n \geq 2$ and 0 for $n \leq 1$. Then we get $\langle f \rangle(0) = 0$, $\langle f \rangle(1) = \langle f \rangle(2) = 1$, and $\langle f \rangle(n) = \infty$ for all $n \geq 2$, represented by the gate $\langle f \rangle = \text{gate}(-+--\overline{+})$. The gate corresponding to g is $\langle g \rangle = \text{gate}(--\overline{+}, +-\overline{+}, \overline{+})$.

Example 5.13. Consider a flat stream function specification with the following rules which use pattern matching on the data constructors 0 and 1:

$$f(0 : \sigma) \to g(\sigma) \qquad f(1 : x : \sigma) \to x : g(\sigma) \qquad g(x : y : \sigma) \to x : y : g(\sigma)$$

denoted ρ_{f_0}, ρ_{f_1}, and ρ_g, respectively. Then, $\langle f \rangle$ is the solution for $X_{f,1}$ of:

$$X_f(n) = \min(X_{f,\star}(0), X_{f,1}(n))$$
$$X_{f,1}(n) = \inf\left\{X_{f,1,\rho_{f_0}}(n), X_{f,1,\rho_{f_1}}(n)\right\}$$
$$X_{f,1,\rho_{f_0}}(n) = \text{let } n' := n - 1, \text{ if } n' < 0 \text{ then } 0 \text{ else } X_{g,1}(n')$$
$$X_{f,1,\rho_{f_1}}(n) = \text{let } n' := n - 2, \text{ if } n' < 0 \text{ then } 0 \text{ else } 1 + X_{g,1}(n')$$
$$X_{f,\star}(n) = \min(X_{g,\star}(0), 1 + X_{g,\star}(0))$$
$$X_{g,1}(n) = \text{let } n' := n - 2, \text{ if } n' < 0 \text{ then } 0 \text{ else } 2 + X_{g,1}(n')$$
$$X_{g,\star}(n) = 2 + X_{g,\star}(0) .$$

As solution we obtain an overlapping of both traces $\langle f \rangle_{1,\rho_{f_0}}$ and $\langle f \rangle_{1,\rho_{f_1}}$, that is, $\langle f \rangle_1(n) = n \div 2$ represented by the gate $\langle f \rangle = \text{gate}(--\overline{-+})$.

The following proposition explains the correspondence between Def. 5.1 and Def. 5.9.

Proposition 5.14. *Let $\mathcal{T}$ be a stream specification. For each $f \in \Sigma_{sfun}$ it holds:*

$$\langle f \rangle(n_1, \ldots, n_k) = [\![f]\!](n_1, \ldots, n_k) \qquad \textit{(for all } n_1, \ldots, n_k \in \overline{\mathbb{N}}),$$

where $k = ar_s(f)$. That is, the p-i function translation $\langle f \rangle$ of f coincides with the p-i function that is represented by the gate translation $[\![f]\!]$.

The following lemma states that the translation $\langle f \rangle$ of a flat stream function symbol f (as defined in Def. 5.1) is the d-o lower bound on the production function of f. For friendly nesting stream symbols f it states that $\langle f \rangle$ pointwisely bounds from below the d-o lower bound on the production function of f.

Lemma 5.15. *Let $\mathcal{T}$ be a stream specification, and let $f \in \Sigma_{fnest} \supseteq \Sigma_{flat}$.*

(i) If f is flat, then: $\langle f \rangle = \underline{do}_{\mathcal{T}}(f)$. Hence, $\underline{do}_{\mathcal{T}}(f)$ is periodically increasing.
(ii) If f is friendly nesting, then it holds: $\langle f \rangle \leq \underline{do}_{\mathcal{T}}(f)$ (pointwise inequality).

5.2 Translation of Stream Constants

In the second step, we now define a translation of stream constants in a flat or friendly nesting stream specification into production terms under the assumption that gate translations for the stream functions are given. Here the idea is that the recursive definition of a stream constant M is unfolded step by step; the terms thus arising are translated according to their structure using gate translations of the stream function symbols from a given family of gates; whenever a stream constant is met that has been unfolded before, the translation stops after establishing a binding to a μ-binder created earlier.

Definition 5.16. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification, and $\mathcal{F} = \{\gamma_f\}_{f \in \Sigma_{sfun}}$ a family of gates that are associated with the symbols in Σ_{sfun}.

Let $Ter(\Sigma)_S^0$ be the set of terms in $\mathcal{T}$ of sort stream that do not contain variables of sort stream. The *translation function* $[\cdot]^{\mathcal{F}} : Ter(\Sigma)_S^0 \to \mathcal{P}$ is defined by $s \mapsto [s]^{\mathcal{F}} := [s]_{\varnothing}^{\mathcal{F}}$ based on the following definition of expressions $[s]_{\alpha}^{\mathcal{F}}$, where $\alpha \subseteq \Sigma_{scon}$, by induction on the structure of $s \in Ter(\Sigma)_S^0$, using the clauses:

$$[M(\boldsymbol{u})]_{\alpha}^{\mathcal{F}} := \begin{cases} \mu M.\mathsf{min} \ \{[r]_{\alpha \cup \{M\}}^{\mathcal{F}} \mid M(\boldsymbol{v}) \to r \in R\} & \text{if } M \notin \alpha \\ M & \text{if } M \in \alpha \end{cases}$$

$$[u : s]_{\alpha}^{\mathcal{F}} := \bullet([s]_{\alpha}^{\mathcal{F}})$$

$$[f(s_1, \ldots, s_{ar_s(f)}, u_1, \ldots, u_{ar_d(f)})]_{\alpha}^{\mathcal{F}} := \gamma_f([s_1]_{\alpha}^{\mathcal{F}}, \ldots, [s_{ar_s(f)}]_{\alpha}^{\mathcal{F}})$$

where $M \in \Sigma_{scon}$, $\boldsymbol{u} = \langle u_1, \ldots, u_{ar_d(M)} \rangle$ a vector of terms in $Ter(\Sigma)_D$, $f \in \Sigma_{sfun}$, $s, s_1, \ldots s_{ar_s(f)} \in Ter(\Sigma)_S^0$, and $u, u_1, \ldots, u_{ar_d(f)} \in Ter(\Sigma)_D$. Note that the definition of $[M(\boldsymbol{u})]_{\alpha}^{\mathcal{F}}$ does not depend on the vector $\boldsymbol{u}$ of terms in $Ter(\Sigma)_D$. Therefore we define, for all $M \in \Sigma_{scon}$, the *translation of M with respect to $\mathcal{F}$* by $[M]^{\mathcal{F}} := [M(\boldsymbol{x})]_{\emptyset}^{\mathcal{F}} \in \mathcal{P}$ where $\boldsymbol{x} = \langle x_1, \ldots, x_{ar_d(M)} \rangle$ is a vector of data variables.

The following lemma is the basis of our result in Sec. 6, Thm. 6.1, concerning the decidability of d-o productivity for flat stream specifications. In particular the lemma states that if we use gates that represent d-o optimal lower bounds on the production of the stream functions, then the translation of a stream constant M yields a production term that rewrites to the d-o lower bound of the production of M.

Lemma 5.17. *Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification, and $\mathcal{F} = \{\gamma_f\}_{f \in \Sigma_{sfun}}$ be a family of gates such that, for all $f \in \Sigma_{sfun}$, the arity of γ_f equals the stream arity of f. Then the following statements hold:*

(i) Suppose that $[\![\gamma_f]\!] = \underline{do}_{\mathcal{T}}(f)$ holds for all $f \in \Sigma_{sfun}$. Then for all $M \in \Sigma_{scon}$ and vectors $\boldsymbol{u} = \langle u_1, \ldots, u_{ar_d(M)} \rangle$ of data terms $[\![[M]^{\mathcal{F}}]\!] = \underline{do}_{\mathcal{T}}(M(\boldsymbol{u}))$ holds. And consequently, $\mathcal{T}$ is d-o productive if and only if $[\![[M_0]^{\mathcal{F}}]\!] = \infty$.

(ii) Suppose that $[\![\gamma_f]\!] = \overline{do}_{\mathcal{T}}(f)$ holds for all $f \in \Sigma_{sfun}$. Then for all $M \in \Sigma_{scon}$ and vectors $\boldsymbol{u} = \langle u_1, \ldots, u_{ar_d(M)} \rangle$ of data terms $[\![[M]^{\mathcal{F}}]\!] = \overline{do}_{\mathcal{T}}(M(\boldsymbol{u}))$ holds. And consequently, $\mathcal{T}$ is d-o non-productive if and only if $[\![[M_0]^{\mathcal{F}}]\!] < \infty$.

Lem. 5.17 is an immediate consequence of the following lemma, which also is the basis of our result in Sec. 6 concerning the recognizability of productivity for flat and friendly nesting stream specifications. In particular the lemma below asserts that if we use gates that represent p-i functions which are lower bounds on the production of the stream functions, then the translation of a stream constant M yields a production term that rewrites to a number in $\overline{\mathbb{N}}$ smaller or equal to the d-o lower bound of the production of M.

Lemma 5.18. *Let $\mathcal{T}$ be a stream specification, and let $\mathcal{F} = \{\gamma_f\}_{f \in \Sigma_{sfun}}$ be a family of gates such that, for all $f \in \Sigma_{sfun}$, the arity of γ_f equals the stream arity of f. Suppose that one of the following statements holds:*

(a) $[\![\gamma_f]\!] \leq \underline{do}_{\mathcal{T}}(f)$ for all $f \in \Sigma_{sfun}$;
(b) $[\![\gamma_f]\!] \geq \underline{do}_{\mathcal{T}}(f)$ for all $f \in \Sigma_{sfun}$;
(c) $\overline{do}_{\mathcal{T}}(f) \leq [\![\gamma_f]\!]$ for all $f \in \Sigma_{sfun}$;
(d) $\overline{do}_{\mathcal{T}}(f) \geq [\![\gamma_f]\!]$ for all $f \in \Sigma_{sfun}$.

Then, for all $M \in \Sigma_{scon}$ and vectors $\boldsymbol{u} = \langle u_1, \ldots, u_{ar_d(M)} \rangle$ of data terms in $\mathcal{T}$, the corresponding one of the following statements holds:

(a) $[\![[M]^{\mathcal{F}}]\!] \leq \underline{do}_{\mathcal{T}}(M(\boldsymbol{u}))$;
(b) $[\![[M]^{\mathcal{F}}]\!] \geq \underline{do}_{\mathcal{T}}(M(\boldsymbol{u}))$;
(c) $\overline{do}_{\mathcal{T}}(M(\boldsymbol{u})) \leq [\![[M]^{\mathcal{F}}]\!]$;
(d) $\overline{do}_{\mathcal{T}}(M(\boldsymbol{u})) \geq [\![[M]^{\mathcal{F}}]\!]$.

6   Deciding Data-Oblivious Productivity

In this section we assemble our results concerning decision of d-o productivity, and automatable recognition of productivity. We define methods:

(DOP) for deciding d-o productivity of flat stream specifications,
(DP) for deciding productivity of pure stream specifications, and
(RP) for recognising productivity of friendly nesting stream specifications,

that proceed in the following steps:

(i) Take as input a (DOP) flat, (DP) pure, or (RP) friendly nesting stream speci-fication $\mathcal{T} = \langle \Sigma, R \rangle$.

(ii) Translate the stream function symbols into gates $\mathcal{F} := \{\langle \mathsf{f} \rangle\}_{\mathsf{f} \in \Sigma_{sfun}}$ (Def. 5.1).

(iii) Construct the production term $[\mathsf{M}_0]^{\mathcal{F}}$ with respect to $\mathcal{F}$ (Def. 5.16).

(iv) Compute the production k of $[\mathsf{M}_0]^{\mathcal{F}}$ using $\to_{\mathsf{R}}$ (Def. 4.27).

(v) Give the following output:

(DOP) "$\mathcal{T}$ is d-o productive" if $k = \infty$, else "$\mathcal{T}$ is not d-o productive".

(DP) "$\mathcal{T}$ is productive" if $k = \infty$, else "$\mathcal{T}$ is not productive".

(RP) "$\mathcal{T}$ is productive" if $k = \infty$, else "don't know".

Note that all of these steps are automatable (cf. our productivity tool, Sec. 8).

Our main result states that d-o productivity is decidable for flat stream spec-ifications. It rests on the fact that the algorithm DOP obtains the lower bound $\underline{do}_{\mathcal{T}}([\mathsf{M}_0])$ on the production of M_0 in $\mathcal{T}$. Since d-o productivity implies pro-ductivity (Prop. 3.4), we obtain a computable, d-o optimal, sufficient condition for productivity of flat stream specifications, which cannot be improved by any other d-o analysis. Second, since for pure stream specifications d-o productivity and productivity are the same, we get that productivity is decidable for them.

Theorem 6.1. *(i)* DOP *decides d-o productivity of flat stream specifications,* *(ii)* DP *decides productivity of pure stream specifications.*

Proof. Let k be the production of the term $[\mathsf{M}_0]^{\mathcal{F}} \in \mathcal{P}$ in step (iv) of DOP/DP.

(i) By Lem. 5.15 (i), Lem. 5.17 (i), Lem. 4.28, Lem. 4.29 we find: $k = \underline{do}_{\mathcal{T}}(\mathsf{M}_0)$.

(ii) For pure specifications we additionally note: $\Pi_{\mathcal{T}}(\mathsf{M}_0) = \underline{do}_{\mathcal{T}}(\mathsf{M}_0)$.   □

Third, we obtain a computable, sufficient condition for productivity of friendly nesting stream specifications.

Theorem 6.2. *A friendly nesting (flat) stream specification $\mathcal{T}$ is productive if the algorithm* RP *(*DOP*) recognizes $\mathcal{T}$ as productive.*

Proof. Let k be the production of the term $[\mathsf{M}_0]^{\mathcal{F}} \in \mathcal{P}$ in step (iv) of RP/DOP. By Lem. 5.15 (ii), Lem. 5.18 (a), Lem. 4.28 and Lem. 4.29: $k \leq \underline{do}_{\mathcal{T}}(\mathsf{M}_0) \leq \Pi_{\mathcal{T}}(\mathsf{M}_0)$.   □

Example 6.3. We illustrate the translation and decision of d-o productivity by means of Pascal's triangle, Fig. 2. The translation of the stream function sym-bols is $\mathcal{F} = \{\langle \mathsf{f} \rangle\}$ with $\langle \mathsf{f} \rangle = \mathsf{gate}(-\overline{-+})$, see page 32. We calculate $[\mathsf{P}]^{\mathcal{F}}$, the translation of $\overline{\mathsf{P}}$, and reduce it to normal form with respect to $\to_{\mathsf{R}}$:

$$[\mathsf{P}]^{\mathcal{F}} = \mu P.\bullet(\bullet(-\overline{-+}(P))) \twoheadrightarrow_{\mathsf{R}} \mu P.++-\overline{-+}(P) \to_{\mathsf{R}} \underline{\infty}$$

Hence $\underline{do}_{\mathcal{T}}(\mathsf{P}) = \infty$, and P is d-o productive and therefore productive.

7   Examples

Productivity of all of the following examples is recognized fully automatically by a Haskell implementation of our decision algorithm for data-oblivious productivity. The tool and a number of examples can be found at:

http://infinity.few.vu.nl/productivity.

In Subsections 7.1–7.6 below, we give possible input-representations as well as the tool-output for the examples of stream specifications in Section 2 and for an example of a stream function specification in Section 3.

7.1   Example in Fig. 2 on Page 3

For applying the automated productivity prover to the flat stream specification in Fig. 2 on page 2 of the stream of rows in Pascal's triangle, one can use the input:

```
Signature(
  P : stream(nat),
  0 : nat,
  f : stream(nat) -> stream(nat),
  a : nat -> nat -> nat,
  s : nat -> nat
)
P = 0:s(0):f(P)
f(s(x):y:sigma) = a(s(x),y):f(y:sigma)
f(0:sigma) = 0:s(0):f(sigma)

a(s(x),y) = s(a(x,y))
a(0,y) = y
```

On this input the automated productivity prover produces the following output:

```
The automated productivity prover has been applied to:

    Signature(
      -- stream symbols --
      P : stream(nat),
      f : stream(nat) -> stream(nat),

      -- data symbols --
      0 : nat,
      a : nat -> nat -> nat,
      s : nat -> nat
    )

    -- stream layer --
    P = 0:s(0):f(P)
    f(s(x):y:sigma) = a(s(x),y):f(y:sigma)
```

```
f(0:sigma) = 0:s(0):f(sigma)

-- data layer --
a(s(x),y) = s(a(x,y))
a(0,y) = y
```

Termination of the data layer has been proven automatically.

The function symbol f is flat, we can compute the precise data-oblivious lower bound:

$$[f] = \mathsf{gate}([f]_{\star,0}(\underline{0}),\ [f]_{1,0})$$

$$[f]_{1,0} = \mu f_{1,0}.\wedge \begin{cases} --+\wedge \begin{cases} \mu f_{1,1}.\wedge \begin{cases} -+\wedge \begin{cases} f_{1,1} \\ ++\wedge \begin{cases} f_{1,0} \end{cases} \end{cases} \\ -++\wedge \begin{cases} f_{1,0} \end{cases} \end{cases} \end{cases}$$

$$= --\overline{-+}$$

$$[f]_{\star,0} = \mu f_{\star,0}.\wedge \begin{cases} +f_{\star,0} \\ ++f_{\star,0} \end{cases}$$

$$= \overline{+}$$

P depends only on flat stream functions, we can **decide data-oblivious productivity**.

We translate P into a pebbleflow net and collapse it to a source:[5]

$$[P] = \mu P.\bullet(\bullet([f](P)))$$
$$= \mu P.\bullet(\bullet(-\overline{-+}(P)))$$
$$\twoheadrightarrow_R \mu P.+\overline{-+}(\bullet(-\overline{-+}(P)))$$
$$\twoheadrightarrow_R \mu P.+\overline{-+}(+\overline{-+}(-\overline{-+}(P)))$$
$$\twoheadrightarrow_R \mu P.+\overline{+-}(-\overline{-+}(P))$$
$$\twoheadrightarrow_R \mu P.++-\overline{-+}(P)$$
$$\twoheadrightarrow_R \underline{\infty}$$

The specification of P is productive.

7.2   Example in Fig. 3 on Page 6

For applying the automated productivity prover to the flat stream specification in Fig. 3 on page 6 of the ternary Thue-Morse sequence, one can use the input:

```
Signature(
  Q, Qprime : stream(char),
```

```
    f : stream(char) -> stream(char),
    a, b, c: char
)
Q = a:Qprime
Qprime = b:c:f(Qprime)
f(a:sigma) = a:b:c:f(sigma)
f(b:sigma) = a:c:f(sigma)
f(c:sigma) = b:f(sigma)
```

On this input the automated productivity prover produces the following output:

The automated productivity prover has been applied to:

```
Signature(
  -- stream symbols --
  Q : stream(char),
  Qprime : stream(char),
  f : stream(char) -> stream(char),

  -- data symbols --
  a : char,
  b : char,
  c : char
)

-- stream layer --
Q = a:Qprime
Qprime = b:c:f(Qprime)
f(a:sigma) = a:b:c:f(sigma)
f(b:sigma) = a:c:f(sigma)
f(c:sigma) = b:f(sigma)

-- data layer --
```

The function symbol f is flat, we can compute the precise data-oblivious lower bound:

$$[f] = \mathsf{gate}([f]_{\star,0}(\underline{0}),\ [f]_{1,0})$$

$$[f]_{1,0} = \mu f_{1,0}.\wedge \begin{cases} -{+}{+}{+} \wedge \left\{ f_{1,0} \right. \\ -{+}{+} \wedge \left\{ f_{1,0} \right. \\ -{+} \wedge \left\{ f_{1,0} \right. \end{cases}$$

$$= \overline{-{+}}$$

$$[f]_{\star,0} = \mu f_{\star,0}.\wedge \begin{cases} +{+}{+}f_{\star,0} \\ +{+}f_{\star,0} \\ +f_{\star,0} \end{cases}$$

$$= \overline{+}$$

Q depends only on flat stream functions, we can **decide data-oblivious productivity**.

We translate Q into a pebbleflow net and collapse it to a source:

$$
\begin{aligned}
[Q] &= \mu Q.\bullet(\mu Qprime.\bullet(\bullet([f](Qprime)))) \\
&= {}^{1}\mu Q.\bullet(\mu Qprime.\bullet(\bullet(\overline{-{+}}(Qprime)))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\mu Qprime.\bullet(\bullet(\overline{-{+}}(Qprime)))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\mu Qprime.{+}\overline{-{+}}(\bullet(\overline{-{+}}(Qprime)))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\mu Qprime.{+}\overline{-{+}}({+}\overline{-{+}}(\overline{-{+}}(Qprime)))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\mu Qprime.{+}{+}{-}(\overline{-{+}}(Qprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\mu Qprime.{+}{+}{-}(Qprime)) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.{+}\overline{-{+}}(\underline{\infty}) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Q.\underline{\infty} \\
&\twoheadrightarrow_{\mathsf{R}} \underline{\infty}
\end{aligned}
$$

The specification of Q is productive.

$Qprime$ depends only on flat stream functions, we can **decide data-oblivious productivity**.

We translate Qprime into a pebbleflow net and collapse it to a source:[7]

$$
\begin{aligned}
[\mathsf{Qprime}] &= \mu Qprime.\bullet(\bullet([\mathsf{f}](Qprime))) \\
&= \mu Qprime.\bullet(\bullet(\overline{-+}(Qprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Qprime.+\overline{-+}(\bullet(\overline{-+}(Qprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Qprime.+\overline{-+}(+\overline{-+}(\overline{-+}(Qprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Qprime.+\overline{+-}(\overline{-+}(Qprime)) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Qprime.+\overline{+-}(Qprime) \\
&\twoheadrightarrow_{\mathsf{R}} \underline{\infty}
\end{aligned}
$$

The specification of Qprime is productive.

7.3   Example in Fig. 4 on Page 7

For applying the automated productivity prover to the pure stream specification in Fig. 4 on page 7 of the ternary Thue-Morse sequence, one can use the input:

```
Signature(
  Q : stream(char),
  M : stream(bit),
  zip : stream(x) -> stream(x) -> stream(x),
  inv : stream(bit) -> stream(bit),
  tail : stream(x) -> stream(x),
  diff : stream(bit) -> stream(char),
  i : bit -> bit,
  X : bit -> bit -> char,
  0, 1 : bit,
  a,b,c : char
)
Q = diff(M)
M = 0:zip(inv(M),tail(M))
zip(x:s,t) = x:zip(t,s)
inv(x:s) = i(x):inv(s)
tail(x:s) = s
diff(x:y:s) = X(x,y):diff(y:s)

i(0) = 1
i(1) = 0
X(0,0) = b
X(0,1) = a
X(1,0) = c
X(1,1) = b
```

The automated productivity prover then gives the following output:

The automated productivity prover has been applied to:

```
Signature(
  -- stream symbols --
  Q : stream(char),
  M : stream(bit),
  zip : stream(x) -> stream(x) -> stream(x),
  inv : stream(bit) -> stream(bit),
  tail : stream(x) -> stream(x),
  diff : stream(bit) -> stream(char),

  -- data symbols --
  i : bit -> bit,
  X : bit -> bit -> char,
  0 : bit,
  1 : bit,
  a : char,
  b : char,
  c : char
)

-- stream layer --
Q = diff(M)
M = 0:zip(inv(M),tail(M))
zip(x:s,t) = x:zip(t,s)
inv(x:s) = i(x):inv(s)
tail(x:s) = s
diff(x:y:s) = X(x,y):diff(y:s)

-- data layer --
i(0) = 1
i(1) = 0
X(0,0) = b
X(0,1) = a
X(1,0) = c
X(1,1) = b
```

Termination of the data layer has been proven automatically.

The function symbol zip is pure, we can compute its precise production modulus:

$$[\mathsf{zip}] = \mathsf{gate}([\mathsf{zip}]_{\star,0}(\underline{0}),\ [\mathsf{zip}]_{1,0},\ [\mathsf{zip}]_{2,0})$$

$$[\mathsf{zip}]_{1,0} = \mu zip_{1,0}.\wedge\left\{-+\wedge\left\{\mu zip_{2,0}.\wedge\left\{+\wedge\left\{zip_{1,0}\right.\right.\right.\right.$$

$$= \overline{-++}$$

$$[\mathsf{zip}]_{2,0} = \mu zip_{2,0}.\wedge\left\{+\wedge\left\{\mu zip_{1,0}.\wedge\left\{-+\wedge\left\{zip_{2,0}\right.\right.\right.\right.$$

$$= \overline{+-+}$$

$$[\mathsf{zip}]_{\star,0} = \mu zip_{\star,0}.\wedge\left\{+zip_{\star,0}\right.$$

$$= \overline{+}$$

The function symbol inv is pure, we can compute its precise production modulus:

$$[\mathsf{inv}] = \mathsf{gate}([\mathsf{inv}]_{\star,0}(\underline{0}),\ [\mathsf{inv}]_{1,0})$$

$$[\mathsf{inv}]_{1,0} = \mu inv_{1,0}.\wedge\left\{-+\wedge\left\{inv_{1,0}\right.\right.$$

$$= \overline{-+}$$

$$[\mathsf{inv}]_{\star,0} = \mu inv_{\star,0}.\wedge\left\{+inv_{\star,0}\right.$$

$$= \overline{+}$$

The function symbol tail is pure, we can compute its precise production modulus:

$$[\mathsf{tail}] = \mathsf{gate}([\mathsf{tail}]_{\star,0}(\underline{0}),\ [\mathsf{tail}]_{1,0})$$

$$[\mathsf{tail}]_{1,0} = \mu tail_{1,0}.\wedge\left\{-\mu x.-+x\right.$$

$$= -\overline{-+}$$

$$[\mathsf{tail}]_{\star,0} = \mu tail_{\star,0}.\wedge\left\{\mu x.+x\right.$$

$$= \overline{+}$$

The function symbol diff is pure, we can compute its precise production modulus:

$$[\mathsf{diff}] = \mathsf{gate}([\mathsf{diff}]_{\star,0}(\underline{0}),\ [\mathsf{diff}]_{1,0})$$

$$[\mathsf{diff}]_{1,0} = \mu diff_{1,0}.\wedge\left\{--+\wedge\left\{\mu diff_{1,1}.\wedge\left\{-+\wedge\left\{diff_{1,1}\right.\right.\right.\right.$$

$$= -\overline{-+}$$

$$[\mathsf{diff}]_{\star,0} = \mu diff_{\star,0}.\wedge\left\{+diff_{\star,0}\right.$$

$$= \overline{+}$$

Q depends only on pure stream functions, we can **decide productivity**. We translate Q into a pebbleflow net and collapse it to a source:

$$[\mathsf{Q}] = \mu Q.[\mathsf{diff}](\mu M.\bullet([\mathsf{zip}]([\mathsf{inv}](M),\ [\mathsf{tail}](M))))$$
$$= \mu Q.\overline{-\overline{-}+}(\mu M.\bullet(\min(\overline{-++}(\overline{-+}(M)),\overline{+-+}(-\overline{-+}(M))))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\overline{-\overline{-}+}(\mu M.\bullet(\min(\overline{-++}(M),+-\overline{-++}(M)))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\overline{-\overline{-}+}(\mu M.+\overline{-+}(\min(\overline{-++}(M),+-\overline{-++}(M)))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\overline{-\overline{-}+}(\mu M.\min(+\overline{-+}(\overline{-++}(M)),+\overline{-+}(+-\overline{-++}(M)))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\overline{-\overline{-}+}(\mu M.\min(\overline{+-+}(M),++-\overline{-++}(M))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\overline{-\overline{-}+}(\min(\mu M.\overline{+-+}(M),\mu M.++-\overline{-++}(M))))$$
$$\twoheadrightarrow_\mathsf{R} \mu Q.\min(-\overline{-+}(\mu M.\overline{+-+}(M)),-\overline{-+}(\mu M.++-\overline{-++}(M))))$$
$$\twoheadrightarrow_\mathsf{R} \min(\mu Q.-\overline{-+}(\mu M.\overline{+-+}(M)),\mu Q.-\overline{-+}(\mu M.++-\overline{-++}(M))))$$
$$\twoheadrightarrow_\mathsf{R} \min(\mu Q.-\overline{-+}(\underline{\infty}),\mu Q.-\overline{-+}(\underline{\infty}))$$
$$\twoheadrightarrow_\mathsf{R} \min(\mu Q.\underline{\infty},\mu Q.\underline{\infty})$$
$$\twoheadrightarrow_\mathsf{R} \min(\underline{\infty},\underline{\infty})$$
$$\twoheadrightarrow_\mathsf{R} \underline{\infty}$$

The specification of Q is productive.

M depends only on pure stream functions, we can **decide productivity**. We translate M into a pebbleflow net and collapse it to a source:

$$[\mathsf{M}] = \mu M.\bullet([\mathsf{zip}]([\mathsf{inv}](M),\ [\mathsf{tail}](M)))$$
$$= \mu M.\bullet(\min(\overline{-++}(\overline{-+}(M)),\overline{+-+}(-\overline{-+}(M))))$$
$$\twoheadrightarrow_\mathsf{R} \mu M.\bullet(\min(\overline{-++}(M),+-\overline{-++}(M)))$$
$$\twoheadrightarrow_\mathsf{R} \mu M.+\overline{-+}(\min(\overline{-++}(M),+-\overline{-++}(M)))$$
$$\twoheadrightarrow_\mathsf{R} \mu M.\min(+\overline{-+}(\overline{-++}(M)),+\overline{-+}(+-\overline{-++}(M)))$$
$$\twoheadrightarrow_\mathsf{R} \mu M.\min(\overline{+-+}(M),++-\overline{-++}(M))$$
$$\twoheadrightarrow_\mathsf{R} \min(\mu M.\overline{+-+}(M),\mu M.++-\overline{-++}(M))$$
$$\twoheadrightarrow_\mathsf{R} \min(\underline{\infty},\underline{\infty})$$
$$\twoheadrightarrow_\mathsf{R} \underline{\infty}$$

The specification of M is productive.

7.4   Example in Fig. 5 on Page 7

For applying the automated productivity prover to the pure stream specification in Fig. 5 on page 7 of the Thue–Morse sequence using the D0L-system $0 \mapsto 01$, $1 \mapsto 10$, one can use the input:

```
Signature(
```

```
    M : stream(bit),
    Mprime : stream(bit),
    h : stream(bit) -> stream(bit),
    0, 1 : bit
)
M = 0:Mprime
Mprime = 1:h(Mprime)
h(0:sigma) = 0:1:h(sigma)
h(1:sigma) = 1:0:h(sigma)
```

The automated productivity prover then gives the following output:

The automated productivity prover has been applied to:

```
    Signature(
      -- stream symbols --
      M : stream(bit),
      Mprime : stream(bit),
      h : stream(bit) -> stream(bit),

      -- data symbols --
      0 : bit,
      1 : bit
    )

    -- stream layer --
    M = 0:Mprime
    Mprime = 1:h(Mprime)
    h(0:sigma) = 0:1:h(sigma)
    h(1:sigma) = 1:0:h(sigma)

    -- data layer --
```

The function symbol h is pure, we can compute its precise production modulus:

$$[\mathsf{h}] = \mathsf{gate}([\mathsf{h}]_{\star,0}(\underline{0}),\ [\mathsf{h}]_{1,0})$$

$$[\mathsf{h}]_{1,0} = \mu h_{1,0}.\ \wedge \begin{cases} -{+}{+} \wedge \begin{Bmatrix} h_{1,0} \end{Bmatrix} \\ -{+}{+} \wedge \begin{Bmatrix} h_{1,0} \end{Bmatrix} \end{cases}$$

$$= \overline{-{+}{+}}$$

$$[\mathsf{h}]_{\star,0} = \mu h_{\star,0}.\ \wedge \begin{cases} {+}{+}h_{\star,0} \\ {+}{+}h_{\star,0} \end{cases}$$

$$= \overline{\mp}$$

M depends only on pure stream functions, we can **decide productivity**.

We translate M into a pebbleflow net and collapse it to a source:[9]

$$
\begin{aligned}
[\mathsf{M}] &= \mu M.\bullet(\mu Mprime.\bullet([\mathsf{h}](Mprime))) \\
&= \mu M.\bullet(\mu Mprime.\bullet(\overline{-++}(Mprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu M.+\overline{--+}(\mu Mprime.\bullet(\overline{-++}(Mprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu M.+\overline{--+}(\mu Mprime.+\overline{--+}(\overline{-++}(Mprime))) \\
&\twoheadrightarrow_{\mathsf{R}} \mu M.+\overline{--+}(\mu Mprime.\overline{+-+}(Mprime)) \\
&\twoheadrightarrow_{\mathsf{R}} \mu M.+\overline{--+}(\infty) \\
&\twoheadrightarrow_{\mathsf{R}} \mu M.\infty \\
&\twoheadrightarrow_{\mathsf{R}} \infty
\end{aligned}
$$

The specification of M is productive.

Mprime depends only on pure stream functions, we can **decide productivity**. We translate Mprime into a pebbleflow net and collapse it to a source:[9]

$$
\begin{aligned}
[\mathsf{Mprime}] &= \mu Mprime.\bullet([\mathsf{h}](Mprime)) \\
&= \mu Mprime.\bullet(\overline{-++}(Mprime)) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Mprime.+\overline{--+}(\overline{-++}(Mprime)) \\
&\twoheadrightarrow_{\mathsf{R}} \mu Mprime.\overline{+-+}(Mprime) \\
&\twoheadrightarrow_{\mathsf{R}} \infty
\end{aligned}
$$

The specification of Mprime is productive.

7.5   Example in Fig. 6 on Page 8

For applying the automated productivity prover to the pure stream specification in Fig. 6 on page 8 of the Thue–Morse sequence using the D0L-system $0 \mapsto 01$, $1 \mapsto 10$, one can use the input:

```
Signature(
  nats, ones : stream(nat),
  conv, add : stream(nat) -> stream(nat) -> stream(nat),
  times : stream(nat) -> nat -> stream(nat),
  0 : nat,
  s : nat -> nat,
  a, m : nat -> nat -> nat
)
nats = 0:conv(ones,ones)
ones = s(0):ones
conv(x:sigma,y:tau) = m(x,y):add(times(tau,x),conv(sigma,y:tau))
times(x:sigma,y) = m(x,y):times(sigma,y)
add(x:sigma,y:tau) = a(x,y):add(sigma,tau)
```

```
a(0,y) = y
a(s(x),y) = s(a(x,y))
m(0,y) = 0
m(s(x),y) = a(y,m(x,y))
```

The automated productivity gives as output:

The automated productivity prover has been applied to:

```
Signature(
  -- stream symbols --
  nats : stream(nat),
  ones : stream(nat),
  conv : stream(nat) -> stream(nat) -> stream(nat),
  add : stream(nat) -> stream(nat) -> stream(nat),
  times : stream(nat) -> nat -> stream(nat),

  -- data symbols --
  0 : nat,
  s : nat -> nat,
  a : nat -> nat -> nat,
  m : nat -> nat -> nat
)

-- stream layer --
nats = 0:conv(ones,ones)
ones = s(0):ones
conv(x:sigma,y:tau) = m(x,y):add(times(tau,x),conv(sigma,y:tau))
times(x:sigma,y) = m(x,y):times(sigma,y)
add(x:sigma,y:tau) = a(x,y):add(sigma,tau)

-- data layer --
a(0,y) = y
a(s(x),y) = s(a(x,y))
m(0,y) = 0
m(s(x),y) = a(y,m(x,y))
```

Termination of the data layer has been proven automatically.

The function symbol conv is friendly nesting, we can compute a data-oblivious

lower bound:

$$[\mathsf{conv}] = \mathsf{gate}([\mathsf{conv}]_{\star,0}(\underline{0}),\ [\mathsf{conv}]_{1,0},\ [\mathsf{conv}]_{2,0})$$

$$[\mathsf{conv}]_{1,0} = \mu conv_{1,0}.\ \wedge \left\{ \mu x.-+x \right.$$
$$= \overline{-+}$$

$$[\mathsf{conv}]_{2,0} = \mu conv_{2,0}.\ \wedge \left\{ \mu x.-+x \right.$$
$$= \overline{-+}$$

$$[\mathsf{conv}]_{\star,0} = \mu conv_{\star,0}.\ \wedge \left\{ \mu x.-+x \right.$$
$$= \overline{-+}$$

The function symbol add is pure, we can compute its precise production modulus:

$$[\mathsf{add}] = \mathsf{gate}([\mathsf{add}]_{\star,0}(\underline{0}),\ [\mathsf{add}]_{1,0},\ [\mathsf{add}]_{2,0})$$

$$[\mathsf{add}]_{1,0} = \mu add_{1,0}.\ \wedge \left\{ -+ \wedge \left\{ add_{1,0} \right.\right.$$
$$= \overline{-+}$$

$$[\mathsf{add}]_{2,0} = \mu add_{2,0}.\ \wedge \left\{ -+ \wedge \left\{ add_{2,0} \right.\right.$$
$$= \overline{-+}$$

$$[\mathsf{add}]_{\star,0} = \mu add_{\star,0}.\ \wedge \left\{ +add_{\star,0} \right.$$
$$= \overline{+}$$

The function symbol times is pure, we can compute its precise production modulus:

$$[\mathsf{times}] = \mathsf{gate}([\mathsf{times}]_{\star,0}(\underline{0}),\ [\mathsf{times}]_{1,0})$$

$$[\mathsf{times}]_{1,0} = \mu times_{1,0}.\ \wedge \left\{ -+ \wedge \left\{ times_{1,0} \right.\right.$$
$$= \overline{-+}$$

$$[\mathsf{times}]_{\star,0} = \mu times_{\star,0}.\ \wedge \left\{ +times_{\star,0} \right.$$
$$= \overline{+}$$

nats depends only on flat stream functions, we **try to prove productivity**.

We translate nats into a pebbleflow net and collapse it to a source:[10]

$$[\mathsf{nats}] = \mu nats.\bullet([\mathsf{conv}](\mu ones.\bullet(ones),\ \mu ones.\bullet(ones)))$$

$$= \mu nats.\bullet(\min(\overline{-+}(\underline{0}), \min(\overline{-+}(\mu ones.\bullet(ones)), \overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \mu nats.+\overline{-+}(\min(\overline{-+}(\underline{0}), \min(\overline{-+}(\mu ones.\bullet(ones)), \overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \mu nats.\min(+\overline{-+}(\overline{-+}(\underline{0})), +\overline{-+}(\min(\overline{-+}(\mu ones.\bullet(ones)), \overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \mu nats.\min(\overline{+-}(\underline{0}), +\overline{-+}(\min(\overline{-+}(\mu ones.\bullet(ones)), \overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\mu nats.\overline{+-}(\underline{0}), \mu nats.+\overline{-+}(\min(\overline{-+}(\mu ones.\bullet(ones)), \overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\mu nats.\underline{1}, \mu nats.\min(+\overline{-+}(\overline{-+}(\mu ones.\bullet(ones))), +\overline{-+}(\overline{-+}(\mu ones.\bullet(ones)))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\mu nats.\underline{1}, \mu nats.\min(\overline{+-}(\mu ones.\bullet(ones)), \overline{+-}(\mu ones.\bullet(ones))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \min(\mu nats.\overline{+-}(\mu ones.\bullet(ones)), \mu nats.\overline{+-}(\mu ones.\bullet(ones))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \min(\mu nats.\overline{+-}(\mu ones.+\overline{-+}(ones)), \mu nats.\overline{+-}(\mu ones.+\overline{-+}(ones))))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \min(\mu nats.\overline{+-}(\underline{\infty}), \mu nats.\overline{+-}(\underline{\infty})))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \min(\mu nats.\underline{\infty}, \mu nats.\underline{\infty}))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \min(\underline{\infty}, \underline{\infty}))$$

$$\twoheadrightarrow_{\mathsf{R}} \min(\underline{1}, \underline{\infty})$$

$$\twoheadrightarrow_{\mathsf{R}} \underline{1}$$

Failed to prove productivity of nats.

ones depends only on pure stream functions, we can **decide productivity**.
We translate ones into a pebbleflow net and collapse it to a source:[10]

$$[\mathsf{ones}] = \mu ones.\bullet(ones)$$

$$= \mu ones.\bullet(ones)$$

$$\twoheadrightarrow_{\mathsf{R}} \mu ones.+\overline{-+}(ones)$$

$$\twoheadrightarrow_{\mathsf{R}} \underline{\infty}$$

The specification of ones is productive.

7.6   Example on Page 12

Now we consider the stream function specification below Def. 3.5 on page 12, which is not a stream specification because it does not contain a stream constant as its root. Still, the automated productivity prover can be used to compute the precise data-oblivious lower bound of the stream function f. On the input:

```
Signature(
  f : stream(bit) -> stream(bit),
  g : stream(bit) -> stream(bit) -> stream(bit),
  0, 1 : bit
)
```

```
f(s) = g(s,s)
g(0:y:s,x:t) = 0:0:g(s,t)
g(1:s,x1:x2:x3:x4:t) = 0:0:0:0:0:g(s,t)
```

the productivity prover generates the following output:

The automated productivity prover has been applied to:

```
Signature(
  -- stream symbols --
  f : stream(bit) -> stream(bit),
  g : stream(bit) -> stream(bit) -> stream(bit),

  -- data symbols --
  0 : bit,
  1 : bit
)

-- stream layer --
f(s) = g(s,s)
g(0:y:s,x:t) = 0:0:g(s,t)
g(1:s,x1:x2:x3:x4:t) = 0:0:0:0:0:g(s,t)

-- data layer --
```

The function symbol f is flat, we can compute the precise data-oblivious lower bound:

$$[f] = \mathsf{gate}([f]_{\star,0}(\underline{0}), \ [f]_{1,0})$$

$$[f]_{1,0} = \mu f_{1,0}. \wedge \left\{ \wedge \begin{cases} \mu g_{1,0}. \wedge \begin{cases} --++ \wedge \{ g_{1,0} \\ -+++++ \wedge \{ g_{1,0} \end{cases} \\ \mu g_{2,0}. \wedge \begin{cases} -++ \wedge \{ g_{2,0} \\ ----+++++ \wedge \{ g_{2,0} \end{cases} \end{cases} \right.$$

$$= ----++-++-+--++-+\overline{-++-}$$

$$[f]_{\star,0} = \mu f_{\star,0}. \wedge \left\{ \mu g_{\star,0}. \wedge \begin{cases} ++ g_{\star,0} \\ +++++ g_{\star,0} \end{cases} \right.$$

$$= \overline{+}$$

The function symbol g is flat, we can compute the precise data-oblivious lower bound:

$$[\mathsf{g}] = \mathsf{gate}([\mathsf{g}]_{\star,0}(\underline{0}),\ [\mathsf{g}]_{1,0},\ [\mathsf{g}]_{2,0})$$

$$[\mathsf{g}]_{1,0} = \mu g_{1,0}.\ \wedge \begin{cases} --++ \wedge \left\{ g_{1,0} \right. \\ -+++++ \wedge \left\{ g_{1,0} \right. \end{cases}$$

$$= \overline{--++}$$

$$[\mathsf{g}]_{2,0} = \mu g_{2,0}.\ \wedge \begin{cases} -++ \wedge \left\{ g_{2,0} \right. \\ ----+++++ \wedge \left\{ g_{2,0} \right. \end{cases}$$

$$= --\overline{--++-++-+}$$

$$[\mathsf{g}]_{\star,0} = \mu g_{\star,0}.\ \wedge \begin{cases} ++ g_{\star,0} \\ +++++ g_{\star,0} \end{cases}$$

$$= \overline{+}$$

8   Conclusion and Further Work

In order to formalize quantitative approaches for recognizing productivity of stream specifications, we defined the notion of d-o rewriting and investigated d-o productivity. For the syntactic class of flat stream specifications (that employ pattern matching on data), we devised a decision algorithm for d-o productivity. In this way we settled the productivity recognition problem for flat stream specifications from a d-o perspective. For the even larger class including friendly nesting stream function rules, we obtained a computable sufficient condition for productivity. For the subclass of pure stream specifications (a substantial extension of the class given in [2]) we showed that productivity and d-o productivity coincide, and thereby obtained a decision algorithm for productivity of pure specifications.

   We have implemented in Haskell the decision algorithm for d-o productivity. This tool, together with more information including a manual, examples, our related papers, and a comparison of our criteria with those of [3, 9, 1] can be found at our web page `http://infinity.few.vu.nl/productivity`. The reader is invited to experiment with our tool.

   It is not possible to obtain a d-o optimal criterion for non-productivity of flat specifications in an analogous way to how we established such a criterion for productivity. This is because d-o upper bounds on the production of stream functions in flat stream specifications are not in general periodically increasing

functions. For example, for the following stream function specification:

$$\mathsf{f}(x : \sigma, \tau) \to x : \mathsf{f}(\sigma, \tau) \,, \qquad\qquad \mathsf{f}(\sigma, y : \tau) \to y : \mathsf{f}(\sigma, \tau) \,,$$

it holds that $\overline{do}(\mathsf{f})(n_1, n_2) = n_1 + n_2$, which is not p-i. While this example is not orthogonal, $\overline{do}(\mathsf{f})$ is also not p-i for the following similar orthogonal example:

$$\mathsf{f}(0 : x : \sigma, y : \tau) \to x : \mathsf{f}(\sigma, \tau) \,, \qquad \mathsf{f}(1 : \sigma, x : y : \tau) \to y : \mathsf{f}(\sigma, \tau) \,.$$

Currently we are developing a method that goes beyond a d-o analysis, one that would, e.g., prove productivity of the example B given in the introduction. Moreover, we study a refined production calculus that accounts for the delay of evaluation of stream elements, in order to obtain a faithful modelling of lazy evaluation, needed for example for S on page 4, where the first element depends on a 'future' expansion of S.

References

1. W. Buchholz. A Term Calculus for (Co-)Recursive Definitions on Streamlike Data Structures. *Annals of Pure and Applied Logic*, 136(1-2):75–90, 2005.
2. J. Endrullis, C. Grabmayer, D. Hendriks, A. Isihara, and J.W. Klop. Productivity of Stream Definitions. In *Proc. of FCT 2007*, LNCS, pages 274–287. Springer, 2007.
3. J. Hughes, L. Pareto, and A. Sabry. Proving the Correctness of Reactive Systems Using Sized Types. In *POPL '96*, pages 410–423, 1996.
4. D. Kapur, P. Narendran, D.J. Rosenkrantz, and H. Zhang. Sufficient-Completeness, Ground-Reducibility and their Complexity. *Acta Informatica*, 28(4):311–350, 1991.
5. J.W. Klop and R. de Vrijer. Infinitary Normalization. In S. Artemov, H. Barringer, A.S. d'Avila Garcez, L.C. Lamb, and J. Woods, editors, *We Will Show Them: Essays in Honour of Dov Gabbay*, volume 2, pages 169–192. College Publications, 2005. Item 95 at `http://web.mac.com/janwillemklop/iWeb/Site/Bibliography.html`.
6. J.J.M.M. Rutten. Behavioural Differential Equations: a Coinductive Calculus of Streams, Automata, and Power Series. *Theoretical Computer Science*, 308(1-3):1–53, 2003.
7. A. Salomaa. *Jewels of Formal Language Theory*. Pitman Publishing, 1981.
8. B.A. Sijtsma. On the Productivity of Recursive List Definitions. *ACM Transactions on Programming Languages and Systems*, 11(4):633–649, 1989.
9. A. Telford and D. Turner. Ensuring Streams Flow. In *AMAST*, pages 509–523, 1997.
10. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

Appendix A   Solving Weakly Guarded IO-Term Specifications (Lemma 5.3)

Let $\mathcal{E} = \{X_\alpha = E_\alpha\}_{\alpha \in A}$ be a weakly guarded finite io-sequence specification, $root \in A$, and σ_{root} the unique solution of $\mathcal{E}$ for X_{root}.

Definition A.1. We define a finite, rooted graph $\mathcal{G}_\mathcal{E} = \langle \mathcal{V}, \mathcal{L}, v_{root} \rangle$ that represents traces through the expressions in $\mathcal{E}$. The set of nodes $\mathcal{V}$ consists of all pairs of $\alpha \in A$ together with positions in E_α, the right-hand side of the corresponding equation in $\mathcal{E}$:

$$\mathcal{V} = \{\langle \alpha, p \rangle \mid \alpha \in A, \; p \in Pos(E_\alpha)\} \,.$$

Note that the set of nodes $\mathcal{V}$ is finite. The root node v_{root} is $\langle root, \langle \rangle \rangle$. The set of labelled edges $\mathcal{L} \subseteq \mathcal{V} \times \{-, +, \epsilon\} \times \mathcal{V}$ is the smallest set such that: for all $\alpha \in A$, and every position $p \in Pos(E_\alpha)$ the following conditions hold:

(i) if $E_\alpha|_p = X_{\alpha'}$ then $\langle \alpha, p \rangle \overset{\epsilon}{\to} \langle \alpha', \langle \rangle \rangle \in \mathcal{L}$;

(ii) if $E_\alpha|_p = -E'$ then $\langle \alpha, p \rangle \overset{-}{\to} \langle \alpha, p \cdot 1 \rangle \in \mathcal{L}$;

(iii) if $E_\alpha|_p = +E'$ then $\langle \alpha, p \rangle \overset{+}{\to} \langle \alpha, p \cdot 1 \rangle \in \mathcal{L}$;

(iv) if $E_\alpha|_p = E' \wedge E''$ then $\langle \alpha, p \rangle \overset{\epsilon}{\to} \langle \alpha, p \cdot 1 \rangle$ and $\langle \alpha, p \rangle \overset{\epsilon}{\to} \langle \alpha, p \cdot 2 \rangle \in \mathcal{L}$.

where we have written $v \overset{\ell}{\to} w$ to denote $\langle v, \ell, w \rangle \in \mathcal{L}$. Moreover, for $\ell \in \{-, +, \epsilon\}$, we let $\overset{\ell}{\to}$ denote the relation $\{\langle v, w \rangle \mid v \overset{\ell}{\to} w\}$.

Note that by weak guardedness of $\mathcal{E}$ the relation $\overset{\epsilon}{\to}$ is terminating.

We proceed by defining the set of traces $T(\mathcal{G}_\mathcal{E}, v)$ through the graph $\mathcal{G}_\mathcal{E}$ starting at node $v \in \mathcal{V}$.

Definition A.2. For a path $\rho : v_1 \overset{l_1}{\to} v_2 \overset{l_2}{\to} \ldots$ in $\mathcal{G}_\mathcal{E}$, where $l_1, l_2, \ldots \in \{-, +, \epsilon\}$, we use $trace(\rho)$ to denote $l_1 \cdot l_2 \cdots \in \pm^\infty$, the word obtained by concatenating the symbols '$-$' and '$+$' along the path ρ, thus treating the label 'ϵ' as the empty word. Note that $trace(\rho) \in \pm^\omega$ whenever the path ρ is infinite by weak guardedness of $\mathcal{E}$. The set of traces $T(\mathcal{G}_\mathcal{E}, v) \subseteq \pm^\omega$ through $\mathcal{G}_\mathcal{E}$ from $v \in \mathcal{V}$ is defined as follows:

$$T(\mathcal{G}_\mathcal{E}, v) := \{trace(\rho) \mid \rho \text{ is an infinite path starting from node } v \text{ in } \mathcal{G}_\mathcal{E}\} \,.$$

For $Z \equiv \{\tau_1, \ldots, \tau_n\} \subseteq \pm^\omega$, let $\bigwedge_{\tau \in Z} \tau$ denote $\tau_1 \wedge \ldots \wedge \tau_n$, where $\wedge$ is the binary operation defined in Def. 4.13. If $Z = \varnothing$, we set $\bigwedge_{\tau \in Z} \tau = \overline{\top}$.

The following lemma establishes a correspondence between the specification $\mathcal{E}$ and the graph $\mathcal{G}_\mathcal{E}$, in particular it implies that the unique solution σ_{root} of $\mathcal{E}$ for X_{root} equals the infimum over all traces through $\mathcal{G}_\mathcal{E}$ from the root node.

Lemma A.3. *For $\alpha \in A$, $p \in Pos(E_\alpha)$, let $\sigma(\alpha, p) \in \pm^\omega$ denote the unique solution for the expression $E_\alpha|_p$ in $\mathcal{E}$, and let $\tau(\alpha, p)$ be shorthand for $\bigwedge_{\tau \in T(\mathcal{G}_\mathcal{E}, \langle \alpha, p \rangle)} \tau$. Then, for all $\alpha \in A$, $p \in Pos(E_\alpha)$, we have $\sigma(\alpha, p) = \tau(\alpha, p)$.*

Proof. By coinduction, i.e., we show that the relation $R \in \pm^\omega \times \pm^\omega$ defined by:

$$R := \{ \langle \sigma(\alpha, p), \tau(\alpha, p) \rangle \mid \alpha \in A, \; p \in Pos(E_\alpha) \}$$

is a bisimulation. We proceed by well-founded induction on pairs $\langle \alpha, p \rangle$ with respect to $\overset{\epsilon}{\to}$.

(Base) If $v := \langle \alpha, p \rangle$ is in normal form with respect to $\overset{\epsilon}{\to}$, then v has precisely one outgoing edge: $v \overset{-}{\to}$ or $v \overset{+}{\to}$. In case $v \overset{-}{\to}$, we have that $E_\alpha|_p = -E'$, and therefore $\sigma(\alpha, p) = -\sigma(\alpha, p \cdot 1)$. Furthermore, we have $\tau(\alpha, p) = -\tau(\alpha, p \cdot 1)$, and clearly $\langle \sigma(\alpha, p \cdot 1), \tau(\alpha, p \cdot 1) \rangle \in R$. The proof for case $v \overset{+}{\to}$ proceeds analogously.

(Step) If $v := \langle \alpha, p \rangle$ is not a normal form with respect to $\overset{\epsilon}{\to}$, then either v has one outgoing edge $v \overset{\epsilon}{\to} v'$ or two outgoing edges $v \overset{\epsilon}{\to} v'$ and $v \overset{\epsilon}{\to} v''$. In the first case we have $E_\alpha|_p = X_{\alpha'}$, and then $v' = \langle \alpha', \langle \rangle \rangle$, $\sigma(\alpha, p) = \sigma(\alpha', \langle \rangle)$ and $\tau(\alpha, p) = \tau(\alpha', \langle \rangle)$, and we conclude by an application of the induction hypothesis for $\langle \alpha', \langle \rangle \rangle$. In the second case we have that $E_\alpha|_p = E' \wedge E''$, and then $v' = \langle \alpha, p \cdot 1 \rangle$ and $v'' = \langle \alpha, p \cdot 2 \rangle$. Thus we get $\sigma(\alpha, p) = \sigma(\alpha, p \cdot 1) \wedge \sigma(\alpha, p \cdot 2)$, and, by associativity of $\wedge$, $\tau(\alpha, p) = \tau(\alpha, p \cdot 1) \wedge \tau(\alpha, p \cdot 2)$. We conclude by a double application of the induction hypothesis. $\qquad\square$

The set $T(\mathcal{G}_\mathcal{E}, v)$ consists of infinite, possibly non-productive io-sequences. Therefore we adopt the definition of the production function $[\![\sigma]\!] : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ to be applicable to non-productive io-sequences σ by dropping ∞ from the domain.

Definition A.4. The *production function* $\tilde{\pi}_\sigma$ of a sequence $\sigma \in \pm^\omega$ is corecursively defined by, for all $n \in \mathbb{N}$, $\tilde{\pi}_\sigma(n) := \tilde{\pi}(\sigma, n)$:

$$\tilde{\pi}(+\sigma, n) = 1 + \tilde{\pi}(\sigma, n) , \qquad \tilde{\pi}(-\sigma, 0) = 0 , \qquad \tilde{\pi}(-\sigma, n+1) = \tilde{\pi}(\sigma, n) .$$

A property of the operation $\wedge$ given in Def. 4.13 is that the mapping $[\![\_]\!] : \pm^\infty \to \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ preserves infimum, see Prop. 4.19.

Clearly, a similar property holds for $\tilde{\pi}$: for all $\sigma_1, \sigma_2 \in \pm^\omega$, we have: $\tilde{\pi}_{\sigma_1 \wedge \sigma_2} = \tilde{\pi}_{\sigma_1} \wedge \tilde{\pi}_{\sigma_2}$. Then, the following corollary directly follows from Lemma A.3.

Corollary A.5. *For all $n \in \mathbb{N}$ we have $\tilde{\pi}_{\sigma_{root}}(n) = \inf\{\tilde{\pi}_\tau(n) \mid \tau \in T(\mathcal{G}_\mathcal{E}, v_{root})\}$. Moreover, given $n \in \mathbb{N}$, we can always pick a trace $\tau \in T(\mathcal{G}_\mathcal{E}, v_{root})$ such that $\tilde{\pi}_\tau(n) = \tilde{\pi}_{\sigma_{root}}(n)$.*

Using $\mathcal{G}_\mathcal{E}$ we define a 'two-dimensional diagram' $D_\mathcal{E} \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ representing the traces of $\mathcal{E}$ starting from X_{root} as step functions. For $v \in \mathcal{V}$, $x, y \in \mathbb{N}$ let $v_{x,y}$ be shorthand for $\langle v, x, y \rangle$. The horizontal axis (x-axis) of T corresponds to input, and the vertical axis (y-axis) to output; that is, a '$-$' in the trace yields a step to the right in the diagram and '$+$' a step upwards.

Definition A.6. For $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ we define $\mathcal{G}_{\mathcal{E}}(U) \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$, the *one-step closure of U under $\mathcal{G}_{\mathcal{E}}$*, as follows:

$$\mathcal{G}_{\mathcal{E}}(U) := U \cup \mathcal{G}_{\mathcal{E},\epsilon}(U) \cup \mathcal{G}_{\mathcal{E},+}(U) \cup \mathcal{G}_{\mathcal{E},-}(U)$$

$$\mathcal{G}_{\mathcal{E},\epsilon}(U) := \{w_{x,y} \mid v_{x,y} \in U,\ v \xrightarrow{\epsilon} w \in \mathcal{L}\}$$

$$\mathcal{G}_{\mathcal{E},+}(U) := \{w_{x,y+1} \mid v_{x,y} \in U,\ v \xrightarrow{+} w \in \mathcal{L}\}$$

$$\mathcal{G}_{\mathcal{E},-}(U) := \{w_{x+1,y} \mid v_{x,y} \in U,\ v \xrightarrow{-} w \in \mathcal{L}\}$$

Furthermore we define

$$\mathcal{G}_{\mathcal{E}}^*(U) := \bigcup_{n \in \mathbb{N}} \mathcal{G}_{\mathcal{E}}^n(U) \,,$$

the many-step closure of U under $\mathcal{G}_{\mathcal{E}}$.

Definition A.7. We define $D_{\mathcal{E}} \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ as $D_{\mathcal{E}} := \mathcal{G}_{\mathcal{E}}^*(\{v_{root0,0}\})$.

The diagram $D_{\mathcal{E}}$ contains the traces through $\mathcal{G}_{\mathcal{E}}$ starting from v_{root} as 'step functions'.

Lemma A.8. *For all $v \in \mathcal{V}$, $x, y \in \mathbb{N}$ we have $v_{x,y} \in D_{\mathcal{E}}$ if and only if there exists a path $\rho : v_{root} \to^* v$ in $\mathcal{G}_{\mathcal{E}}$ such that $x = \#_-(trace(\rho))$, $y = \#_+(trace(\rho))$.*

Proof. Follows immediately from the definition of $D_{\mathcal{E}}$. $\qquad\square$

We proceed by showing that the solution σ_{root} of $\mathcal{E}$ for X_{root} coincides with the 'lower bound' of the traces in $D_{\mathcal{E}}$.

Definition A.9. For $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ and $\mathcal{G}_{\mathcal{E}} = \langle \mathcal{V}, \mathcal{L}, v_{root} \rangle$ as above, we define $low_U : \mathbb{N} \to \overline{\mathbb{N}}$, the *lower bound of U*, as follows:

$$low_U(x) := \inf\{y \in \mathbb{N} \mid v_{x,y} \in U,\ \exists w.\ v \xrightarrow{-} w \in \mathcal{L}\} \,,$$

where $\inf \varnothing = \infty$.

Lemma A.10. *The lower bound of $D_{\mathcal{E}}$ is the solution σ_{root} of $\mathcal{E}$ for X_{root}:*

$$\forall n \in \mathbb{N}.\ low_{D_{\mathcal{E}}}(n) = \tilde{\pi}_{\sigma_{root}}(n)$$

Proof. Let $n \in \mathbb{N}$, we show $\tilde{\pi}_{\sigma_{root}}(n) \geq low_{D_{\mathcal{E}}}(n)$ and $\tilde{\pi}_{\sigma_{root}}(n) \leq low_{D_{\mathcal{E}}}(n)$.

$\geq$ For $\tilde{\pi}_{\sigma_{root}}(n) = \infty$ the proof obligation is trivial, thus assume $\tilde{\pi}_{\sigma_{root}}(n) < \infty$. From Corollary A.5 it follows that there exists a sequence $\tau \in T(\mathcal{G}_{\mathcal{E}}, v_{root})$ such that $\tilde{\pi}_{\tau}(n) = \tilde{\pi}_{\sigma_{root}}(n)$. Since $\tilde{\pi}_{\tau}(n) < \infty$ and $\tau \in \pm^{\omega}$, there exists $m \in \mathbb{N}$ with $\#_-(\tau_{0\ldots m-1}) = n < n+1 = \#_-(\tau_{0\ldots m})$, and $\#_+(\tau_{0\ldots m-1}) = \tilde{\pi}_{\tau}(n)$. Then there is a path $\rho : v_{root} \to^* v \xrightarrow{-} v'$ in $\mathcal{G}_{\mathcal{E}}$ such that $trace(\rho) = \tau_{0\ldots m}$. By Lemma A.8 we have $v_{n,\tilde{\pi}_{\sigma_{root}}(n)} \in D_{\mathcal{E}}$, and together with $v \xrightarrow{-} v'$ we conclude $\tilde{\pi}_{\sigma_{root}}(n) \geq low_{D_{\mathcal{E}}}(n)$ according to Definition A.9.

$\leq$ Assume $low_{D_{\mathcal{E}}}(n) < \infty$. From Definition A.9 it follows that there exist nodes $v, v' \in \mathcal{V}$ with $v_{n, low_{D_{\mathcal{E}}}(n)} \in D_{\mathcal{E}}$ and $v \xrightarrow{-} v'$. Consequently by Lemma A.8 there exists a path $\rho : v_{root} \rightarrow^* v \xrightarrow{-} v'$ in $\mathcal{G}_{\mathcal{E}}$ such that $\#_-(trace(\rho)) = n + 1$ and $\#_+(trace(\rho)) = low_{D_{\mathcal{E}}}(n)$. We choose a $\tau \in \pm^{\omega}$ that has $trace(\rho)$ as a prefix, so that $\tilde{\pi}_{\tau}(n) = low_{D_{\mathcal{E}}}(n)$. Then, by Corollary A.5, we know that $\tilde{\pi}_{\sigma_{root}}(n) \leq low_{D_{\mathcal{E}}}(n)$.

We conclude $\tilde{\pi}_{\sigma_{root}}(n) = low_{D_{\mathcal{E}}}(n)$. $\qquad\square$

Although in principle $D_{\mathcal{E}}$ could be used to calculate σ_{root} for every $n \in \mathbb{N}$, it remains to be shown how a rational representation of the solution σ_{root} of $\mathcal{E}$ for X_{root} can be obtained. For this purpose we construct a 'simpler' diagram $D_{\mathcal{E}}^0 \subseteq D_{\mathcal{E}}$ that has the same lower bound. We construct $D_{\mathcal{E}}^0$ such that there are only finitely many nodes on each vertical line; thereby we employ the idea that whenever there are two nodes $w_{x,y}$, $w_{x,y'}$ with $y < y'$, then we can omit the upper occurrence $w_{x,y'}$ (including all traces rising from $w_{x,y'}$) without having an influence on the lower bound $low_{D_{\mathcal{E}}}(x)$.

Definition A.11. We define functions $\mathcal{G}_{\mathcal{E}}^0, omit : (\mathcal{V} \times \mathbb{N} \times \mathbb{N}) \rightarrow (\mathcal{V} \times \mathbb{N} \times \mathbb{N})$:

$$\mathcal{G}_{\mathcal{E}}^0(U) := U \cup \mathcal{G}_{\mathcal{E},\epsilon}(U) \cup \mathcal{G}_{\mathcal{E},+}(U)$$
$$omit(U) := \{v_{x,y} \mid v_{x,y} \in U, \neg\exists y' < y.\, v_{x,y'} \in U\}$$

Let $D_{\mathcal{E}}^0(n) \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ for all $n \in \mathbb{N}$ be defined as follows:

$$D_{\mathcal{E}}^0(0) := vclosure(\{v_{root 0,0}\})$$
$$D_{\mathcal{E}}^0(n+1) := D_{\mathcal{E}}^0(n) \cup D_{\mathcal{E}}^0|_{n+1}$$
$$D_{\mathcal{E}}^0|_{n+1} := vclosure(step\_right(n))$$
$$step\_right(n) := \{w_{n+1,y} \mid v_{n,y} \in D_{\mathcal{E}}^0(n),\ v \xrightarrow{-} w \in \mathcal{L}\},$$

where $vclosure(U) \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ is defined to be $vco^m(U)$ for the smallest $m \in \mathbb{N}$ such that $vco^{m+1}(U) = vco^m(U)$ where $vco := omit \circ \mathcal{G}_{\mathcal{E}}^0$, the vertical one-step closure followed by an application of $omit$. Furthermore we define

$$D_{\mathcal{E}}^0 := \bigcup_{n \in \mathbb{N}} D_{\mathcal{E}}^0(n)\,.$$

Note that $D_{\mathcal{E}}^0(n)$ is finite for all $n \in \mathbb{N}$. The important step in the construction is the termination of $vclosure(\_)$ which is guaranteed by the following lemma.

Lemma A.12. *The computation of $vclosure(U)$ is terminating for finite sets $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$, that is, there exists $m \in \mathbb{N}$ such that $vco^{m+1}(U) = vco^m(U)$.*

Proof. For $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ we introduce auxiliary definitions:

$$X(U) := \{x \in \mathbb{N} \mid \exists v \in \mathcal{V}, y \in \mathbb{N}.\, v_{x,y} \in U\} \subseteq \mathbb{N}$$
$$\mathcal{V}(U, x) := \{v \in \mathcal{V} \mid \exists y \in \mathbb{N}.\, v_{x,y} \in U\} \subseteq \mathcal{V}$$
$$Y(U, x, v) := \{y \in \mathbb{N} \mid v_{x,y} \in U\} \subseteq \mathbb{N}$$
$$y(U, x, v) := \min(Y(U, x, v)) \in \mathbb{N}\,.$$

Note that $Y(W, x, v) = \{y(W, x, v)\}$ is a singleton set for every $W := omit(U)$, $x \in X(W)$ and $v \in \mathcal{V}(W, x)$ with $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ arbitrary.

Let $U \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$. We define $U_0 := U$ and for $n = 1, 2, \ldots$ let

$$U_n := vco(U_{n-1}) \, .$$

Note that $U_n = vco^n(U)$. From the definition of $\mathcal{G}_{\mathcal{E}}^0$ and *omit* it follows that:

(i) $\forall n \in \mathbb{N}. \, X(U_n) = X(U)$,
(ii) $\forall n \in \mathbb{N}. \, U_n \subseteq \mathcal{G}_{\mathcal{E}}^0(U_n)$,
(iii) $\forall n \in \mathbb{N}, x \in X(U). \, \mathcal{V}(U_n, x) \subseteq \mathcal{V}(U_{n+1}, x)$, since

$$\mathcal{V}(U_n, x) \subseteq \mathcal{V}(\mathcal{G}_{\mathcal{E}}^0(U_n), x) = \mathcal{V}(vco(U_n), x) = \mathcal{V}(U_{n+1}, x)$$

(iv) $\forall n \geq 1, x \in X(U). \, |\mathcal{V}(U_n, x)| \leq |\mathcal{V}|$.

By (iii) and (iv) we conclude that there exists $n_0 \in \mathbb{N}$:

$$\forall n \geq n_0, x \in X(U). \, \mathcal{V}(U_n, x) = \mathcal{V}(U_{n_0}, x) \, , \tag{1}$$

that is, the set of nodes $\mathcal{V}(U_n, x)$ on each vertical strip x becomes fixed after n_0 steps. Furthermore by (ii) and the definition of *omit* we get

$$\forall n \in \mathbb{N}, x \in X(U), v \in \mathcal{V}(U_n, x). \, y(U_{n+1}, x, v) \leq y(U_n, x, v) \, , \tag{2}$$

that is, the heights of the nodes are non-increasing. Therefore for each $x \in X(U)$ and node $v \in \mathcal{V}(U_{n_0}, v)$ there exists $m_{x,v} \geq n_0$ such that:

$$\forall n \geq m_{x,v}. \, y(U_n, x, v) = y(U_{m_{x,v}}, x, v)$$

Let $m := \max\{m_{x,v} \mid x \in X(U), \, v \in \mathcal{V}(U_{n_0}, v)\}$, then the heights of all nodes are fixed from m onwards and consequently we have $\forall n \geq m. \, U_n = U_m$. □

Lemma A.13. $D_{\mathcal{E}}$ and $D_{\mathcal{E}}^0$ have the same lower bound, that is, $low_{D_{\mathcal{E}}} = low_{D_{\mathcal{E}}^0}$.

Proof.

Now we construct a rational representation of the lower bound of $D_{\mathcal{E}}^0$, which by Lemma A.13 and A.10 is the solution σ_{root} of $\mathcal{E}$ for X_{root}. The construction is based on a 'repetition search' on $D_{\mathcal{E}}^0$, that is, we search two vertical strips x_1 and x_2 that contain similar constellations of nodes allowing to conclude that from the vertical position x_1 onwards the lower bound is 'quasi periodic'.

Definition A.14. For $x \in \mathbb{N}$, $v \in \mathcal{V}$ let

$$\mathcal{V}(x) := \{v \in \mathcal{V} \mid \exists y \in \mathbb{N}. \, v_{x,y} \in D_{\mathcal{E}}^0(x)\} \subseteq \mathcal{V}$$

$$y(x, v) := \begin{cases} y & \text{if there exists } y \in \mathbb{N} \text{ with } v_{x,y} \in D_{\mathcal{E}}^0(x) \\ \infty & \text{otherwise} \end{cases} \, .$$

Note that $y(x, v)$ is well-defined since there exists at most one such $y \in \mathbb{N}$.

Definition A.15. For $x \in \mathbb{N}$ we define

$$h(x) := \min\{y(x, v) \mid v \in \mathcal{V}\} \,,$$

the height of the lowest node on the vertical strip x.

Definition A.16. For $x \in \mathbb{N}$, $v \in \mathcal{V}$ let

$$y_r(x, v) := y(x, v) - h(x) \,,$$

the height of the node v relative to the lowest node on strip x.

Definition A.17. For $x_1 < x_2 \in \mathbb{N}$ with $\mathcal{V}(x_1) \neq \varnothing$ and $\mathcal{V}(x_2) \neq \varnothing$ let

$$d(x_1, x_2) := h(x_2) - h(x_1) \,,$$

the height difference between the vertical strips x_1 and x_2.

Definition A.18. For $x_1 < x_2 \in \mathbb{N}$, $v \in \mathcal{V}$ with $v \in \mathcal{V}(x_1)$ and $v \in \mathcal{V}(x_2)$ let

$$d(x_1, x_2, v) := y(x_2, v) - y(x_1, v) \,,$$

the height difference of the node v between the vertical strips x_1 and x_2, and

$$d_r(x_1, x_2, v) := h(x_1, x_2, v) - d(x_1, x_2) \,,$$

the relative height difference of the node v between the vertical strips x_1 and x_2.

Lemma A.19. *For $x_1, x_2 \in \mathbb{N}$, $v \in \mathcal{V}(x_1)$ it holds:*

$$y_r(x_2, v) = y_r(x_1, v) + d_r(x_1, x_2, v) \,.$$

Proof. Immediately from the definitions. □

We proceed with the definition of 'pseudo repetitions' $\langle x_1, x_2 \rangle \in \mathbb{N}^2$. That is, we require that the vertical strips x_1 and x_2 possess a similar constellation of nodes. A pseudo repetition does not yet guarantee 'quasi periodicity' of the lower bound (having found the cyclic part of the io-sequence), as will be explained after the definition.

Definition A.20. A *pseudo repetition* is a pair $\langle x_1, x_2 \rangle \in \mathbb{N}^2$ such that:

(i) The vertical strips x_1 and x_2 contain the same set of nodes $v \in \mathcal{V}$:

$$\mathcal{V}(x_1) = \mathcal{V}(x_2)$$

(ii) From strip x_1 to x_2 all nodes have increased their height by at least $d(x_1, x_2)$:

$$\forall v \in \mathcal{V}. \, d_r(x_1, x_2, v) \geq 0$$

For a pseudo repetition $\langle x_1, x_2 \rangle$ we can distinguish between nodes v for which $d_r(x_1, x_2, v) = 0$ and those with $d_r(x_1, x_2, v) > 0$. If for all nodes $v \in \mathcal{V}(x_1)$ it holds that $d_r(x_1, x_2, v) = 0$ then we have an exact repetition, yielding 'quasi periodicity' of the lower bound. On the other hand assume that there is a node $v \in \mathcal{V}(x_1)$ which increases its relative height and contributes to the lower bound between the vertical strips x_1 and x_2. Then we do not yet have a 'quasi periodic' lower bound, because due to the increasing relative height of v, the contribution v to the lower bound will change. For this reason we will later strengthen the conditions on pseudo repetitions, yielding 'repetitions' which then will guarantee 'quasi periodicity', see Definition A.27.

Lemma A.21. *There exist pseudo repetitions in $D^0_{\mathcal{E}}$.*

Proof. For every $x \in \mathbb{N}$ we have $\mathcal{V}(x) \in \mathcal{P}(\mathcal{V})$ and $\mathcal{P}(\mathcal{V})$ is a finite set. Hence it follows from the Pigeonhole Principle that there exist indices $i_1 < i_2 < i_3 < \ldots$ with $\mathcal{V}(i_1) = \mathcal{V}(i_2) = \mathcal{V}(i_3) = \ldots$, let $\mathcal{V}' := \mathcal{V}(i_1) = \{v_1, \ldots, v_k\}$ with $k = |\mathcal{V}(i_1)|$.
    Let $\leq$ on $\mathbb{N}^k$ be defined as follows:

$$(a_1, \ldots, a_l) \leq (b_1, \ldots, b_l) \Leftrightarrow a_1 \leq b_1 \wedge \ldots \wedge a_l \leq b_l .$$

For $x \in \{i_1, i_2, \ldots\}$ we define the tuple $\boldsymbol{t_x} := (y_r(x, v_1), \ldots, y_r(x, v_k)) \in \mathbb{N}^k$. It is a well-known fact that $\leq$ is an almost full relation on $\mathbb{N}^k$, see [10]. Hence there exist $x_1, x_2 \in \{i_1, i_2, \ldots\}$ with $\boldsymbol{t_{x_1}} \leq \boldsymbol{t_{x_2}}$. Then $d_r(x_1, x_2, v) \geq 0$ for all $v \in \mathcal{V}'$ which establishes that $\langle x_1, x_2 \rangle$ is a pseudo repetition. $\square$

The following definitions and lemmas are auxiliary for the purpose of proving Lemma A.26.

Definition A.22. *For $U, W \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$ and $d_x, d_y \in \mathbb{N}$ we define*

$$U \leq_{d_x, d_y} W \Longleftrightarrow \forall v_{x,y} \in W. \, \exists y' < y - d. \, v_{x - d_x, y'} \in U .$$

Lemma A.23. *For $x_1 < x_2 \in \mathbb{N}$ and $d_y \in \mathbb{N}$ we have:*

$$D^0_{\mathcal{E}}|_{x_1} \leq_{x_2 - x_1, d_y} D^0_{\mathcal{E}}|_{x_2} \Longleftrightarrow \forall v \in \mathcal{V}(x_1). \, y(x_2, v) \geq d_y + y(x_1, v)$$

Proof. Follows from the definitions. $\square$

Lemma A.24. *If $U_1, U_2 \subseteq \mathcal{V} \times \mathbb{N} \times \mathbb{N}$, $d_x, d_y \in \mathbb{N}$ with $U_1 \leq_{d_x, d_y} U_2$ then:*

(i) $vco(U_1) \leq_{d_x, d_y} vco(U_2)$, and
(ii) $vclosure(U_1) \leq_{d_x, d_y} vclosure(U_2)$.

Proof. From $U_1 \leq_{d_x, d_y} U_2$ it follows that $\mathcal{G}^0_{\mathcal{E}}(U_1) \leq_{d_x, d_y} \mathcal{G}^0_{\mathcal{E}}(U_2)$ holds, and then we get $omit \circ \mathcal{G}^0_{\mathcal{E}}(U_1) \leq_{d_x, d_y} omit \circ \mathcal{G}^0_{\mathcal{E}}(U_2)$ by definition of *omit*.
    By induction we get $\forall n \in \mathbb{N}. \, vco^n(U_1) \leq_{d_x, d_y} vco^n(U_2)$. Furthermore there exist $m_1, m_2 \in \mathbb{N}$ such that $vclosure(U_i) = vco^{m_i}(U_i)$ for $i \in \{1, 2\}$ by definition. From $\forall m'_i \geq m_i. \, vco^{m'_i}(U_i) = vco^{m_i}(U_i)$ it follows that for $i \in \{1, 2\}$ we have $vclosure(U_i) = vco^{\max(m_1, m_2)}(U_i)$. Hence $vclosure(U_1) \leq_{d_x, d_y} vclosure(U_2)$. $\square$

Lemma A.25. *If $x_1 < x_2 \in \mathbb{N}$ with $\mathcal{V}(x_1) = \mathcal{V}(x_2)$, then $\mathcal{V}(x_1+1) = \mathcal{V}(x_2+1)$. If moreover $D^0_{\mathcal{E}}|_{x_1} \leq_{x_2-x_1,d_y} D^0_{\mathcal{E}}|_{x_2}$ for $d_y \in \mathbb{N}$, then $D^0_{\mathcal{E}}|_{x_1+1} \leq_{x_2-x_1,d_y} D^0_{\mathcal{E}}|_{x_2+1}$.*

Proof. From $\mathcal{V}(x_1) = \mathcal{V}(x_2)$ and the definitions of $D^0_{\mathcal{E}}(x_1+1)$ and $D^0_{\mathcal{E}}(x_2+1)$ it follows that $\mathcal{V}(x_1+1) = \mathcal{V}(x_2+1)$.

Assume $D^0_{\mathcal{E}}|_{x_1} \leq_{x_2-x_1,d_y} D^0_{\mathcal{E}}|_{x_2}$ and consider Definition A.11. Then it follows that $step\_right(x_1) \leq_{x_2-x_1,d_y} step\_right(x_2)$. We conclude with an application of Lemma A.24 yielding $D^0_{\mathcal{E}}|_{x_1+1} \leq_{x_2-x_1,d_y} D^0_{\mathcal{E}}|_{x_2+1}$.   □

Lemma A.26. *Let $\langle x_1, x_2 \rangle$ be a pseudo repetition. Then the pair $\langle x_1 + m, x_2 + m \rangle$ is a pseudo repetition for all $m \in \mathbb{N}$.*

Proof. It suffices to show that $\langle x_1 + 1, x_2 + 1 \rangle$ is a pseudo repetition, for $m > 1$ the claim follows by induction.

We get $\mathcal{V}(x_1+1) = \mathcal{V}(x_2+1)$ by Lemma A.25. Furthermore by the definition of pseudo repetition we have $\forall v \in \mathcal{V}(x_1).\, d_r(x_1, x_2, v) \geq 0$, which is equivalent to $D^0_{\mathcal{E}}|_{x_1} \leq_{x_2-x_1,d(x_1,x_2)} D^0_{\mathcal{E}}|_{x_2}$ since $\mathcal{V}(x_1) = \mathcal{V}(x_2)$. Emloying Lemma A.25 we get $D^0_{\mathcal{E}}|_{x_1+1} \leq_{x_2-x_1,d(x_1,x_2)} D^0_{\mathcal{E}}|_{x_2+1}$,

that is $\forall v \in \mathcal{V}(x_1 + 1).\, d_r(x_1 + 1, x_2 + 1, v) \geq 0$. Hence $\langle x_1 + 1, x_2 + 1 \rangle$ is a pseudo repetition.   □

We proceed with the definition of 'repetitions'. Then we show that repetitions exist and that these indeed guarantee quasi periodicity of the lower bound.

Definition A.27. A *repetition* is a pseudo repetition $\langle x_1, x_2 \rangle$ if:

(i) $\langle x_1, x_2 \rangle$ is a pseudo repetition
(ii) Let $I(x_1, x_2) := \{v_{x,y} \mid v_{x,y} \in T_0,\ x = x_1,\ i(x_1, x_2, p) = 0\}$. We require that only nodes from $I(x_1, x_2)$ contribute to the lower bound between x_1 and x_2 and these nodes reconstruct themselfs on the strip x_2:

$$\forall x \in \mathbb{N}, x_1 \leq x \leq x_2.\ low_{\mathcal{G}_{\mathcal{E}}(I)}(x) = low_{T_0}(x) \tag{3}$$

$$\forall v_{x,y} \in I(x_1, x_2).\ v_{x_2, y+d(x_1,x_2)} \in \mathcal{G}_{\mathcal{E}}(I) \tag{4}$$

Both conditions can be effectively checked since $\mathcal{G}_{\mathcal{E}}(I)|_{\leq n}$ can be computed for every $n \in \mathbb{N}$. Having such a repetition is is easy to see that the lower bounds behaves 'quasi periodic' from this point onwards, that is the io-sequence is indeed rational. This is because the the nodes with $i(x_1, x_2, v) = 0$ consitute a exactly repeating pattern. Furthermore the nodes with increasing relative height $i(x_1, x_2, v) > 0$ do not contribute to the lower bound between x_1 and x_2, and since their height cannot decrease they will also not contribute in the future.

It remains to show that we indeed always encounter a repetition $\langle x_1, x_2 \rangle$. We know that there exists a pseudo repetition $\langle x_1, x_2 \rangle$. Then we have for all $m \in \mathbb{N}$ we have $\langle x_1 + m \cdot (x_2 - x_1), x_2 + m \cdot (x_2 - x_1) \rangle$ is a pseudo repetition. For all $m \in \mathbb{N}$ and $v \in \mathcal{V}$ we have $i(x_1 + m \cdot (x_2 - x_1),\ x_2 + m \cdot (x_2 - x_1), v) \geq 0$, therefore it follows that there are some nodes for which the relative height increase will eventually stay at 0 while other nodes continue increasing their relative height. The latter will from some point on no longer contribute to the lower bound and the pattern. Hence there exists m_0 for which $\langle x_1 + m_0 \cdot (x_2 - x_1), x_2 + m_0 \cdot (x_2 - x_1) \rangle$ is a repetition.

Appendix B   Termination of the Function Layer
Translation (Lemma 5.10)

It suffices to show that, for every stream function symbol f and argument position i of f, the infinite io-sequence specification $\mathcal{E}_\mathcal{T}$ defined in Def. 5.4 that has $[f]_i$ as unique solution for the recursion variable $X_{\langle f, i, 0 \rangle}$ can be transformed in finitely many steps into an io-sequence specification $\mathcal{E}'$ with the same solution for the recursion variable $X_{\langle f, i, 0 \rangle}$ and the property that only finitely many recursion variables are reachable in $\mathcal{E}'$ from $X_{\langle f, i, 0 \rangle}$. (This is because from such an io-sequence specification $\mathcal{E}'$ eventually a finite io-sequence specification $\mathcal{E}'_0$ with $[f]_i$ as unique solution for $X_{\langle f, i, 0 \rangle}$ can be extracted.) An algorithm for this purpose is obtained as follows. On the input of $\mathcal{E}_\mathcal{T}$, set $\mathcal{E} := \mathcal{E}_\mathcal{T}$ and repeat the following step on $\mathcal{E}$ as long as it is applicable:

(RPC) Detect and remove a reachable *non-consuming pseudo-cycle* from the io-sequence specification $\mathcal{E}$: Suppose that, for a function symbol h, for $j, k, l \in \mathbb{N}$, and for a recursion variable $X_{\langle h, j, k \rangle}$ that is reachable from $X_{\langle f, i, 0 \rangle}$, we have $X_{\langle h, j, k \rangle} \xrightarrow{w} X_{\langle h, j, l \rangle}$ (from the recursion variable $X_{\langle h, j, k \rangle}$ the variable $X_{\langle h, j, l \rangle}$ is reachable via a path in the specification on which the finite io-sequence w is encountered as the word formed by consecutive labels), where $l < k$ and w only contains symbols '+'. Then modify $\mathcal{E}$ by setting $X_{\langle h, j, k \rangle} = +X_{\langle h, j, k \rangle}$.

It is not difficult to show that a step (RPC) preserves weakly guardedness and the unique solution of $\mathcal{E}$, and that, on input $\mathcal{E}_\mathcal{T}$, the algorithm terminates in finitely many steps, producing an io-sequence specification $\mathcal{E}'$ with $[f]_i$ as the solution for $X_{\langle f, i, 0 \rangle}$ and the property that only finitely many recursion variables are reachable in $\mathcal{E}'$ from $X_{\langle f, i, 0 \rangle}$.

Appendix C   Soundness of the Function Layer Translation

Throughout, let $\mathcal{T} = \langle \Sigma, R \rangle$ be a fixed stream specification. First we introduce a few axillary lemmas and definitions.

Definition C.1. For all stream terms $t \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$ we define by

$$\#_\bullet(t) := \sup\{n \in \mathbb{N} \mid t = \bullet^k(t')\}$$

the *number of leading stream pebbles of t*.

Definition C.2. Let $\mathsf{h} \in \Sigma_{sfun}$ be a a stream function symbol and $j, k \in \mathbb{N}$. We define $\mathsf{h}_{j,k} := \mathsf{h}(\ldots \bullet^\infty \ldots, \bullet^k(\sigma), \ldots \bullet^\infty \ldots) \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$ where the j-th argument position is $\bullet^k(\sigma)$ while all other arguments are $\bullet^\infty$.

Definition C.3. For terms

$$s \equiv \mathsf{g}(\bullet^{n_1}(\sigma), \ldots, \bullet^{n_k}(\sigma)) \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$$
$$t \equiv \mathsf{h}(\bullet^{m_1}(\sigma), \ldots, \bullet^{m_k}(\sigma)) \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$$

with $n_1, \ldots, n_k, m_1, \ldots, m_k \in \overline{\mathbb{N}}$, we write $s \lhd t$ if $n_i \leq m_i$ for all i.

Lemma C.4. *If $s, t \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$ as in Def. C.3 with $s \lhd t$ then $\underline{do}_{\mathcal{T}}(s) \leq \underline{do}_{\mathcal{T}}(t)$.*

Proof. Immediately from the definition. $\square$

We proof the soundness of the function layer translation, that is:

Lemma C.5. *Let $\mathcal{T}$ be a stream definition, and let $\mathsf{f} \in \Sigma_{sfun}$.*

(i) If $\mathcal{T}$ is flat, then it holds: $[\![\mathsf{f}]\!] = \underline{do}_{\mathcal{T}}(\mathsf{f})$.
   Hence, $\underline{do}_{\mathcal{T}}(\mathsf{f})$ is a periodically increasing function.
(ii) If $\mathcal{T}$ is friendly nesting, then it holds: $[\![\mathsf{f}]\!] \leq \underline{do}_{\mathcal{T}}(\mathsf{f})$.

Proof. We start with (i). Let $\mathcal{E}_{\mathcal{T}}$ be the weakly guarded io-sequence specification (over variables $\mathcal{X}$) obtained from the translation. Let $\mathcal{V} : \mathcal{X} \to (\mathbb{N} \to \overline{\mathbb{N}})$ be the unique solution of $\mathcal{E}_{\mathcal{T}}$, which exists by Lemma 5.3. Then by definition we have $[\![\mathsf{f}]\!] = \min(\mathcal{V}(X_{\langle \mathsf{f}, 1, 0 \rangle}), \ldots, \mathcal{V}(X_{\langle \mathsf{f}, n, 0 \rangle}))$.

Assume f is not weakly guarded. There exists $\mathsf{g} \in \Sigma_{sfun}$ and $\gamma : \mathsf{f} \leadsto^* \mathsf{g} \leadsto^+ \mathsf{g}$ without production; let Σ_γ be the set of symbols occurring in γ and $\rho_1, \ldots, \rho_n \in R$ the rules applied in γ (all having zero production). We define T_γ as the set of terms $t \in Ter(\lbrack\!\lbrack\Sigma\rbrack\!\rbrack)$ with $root(t) \in \Sigma_\gamma$. Accoring to Lemma 3.9 we can contruct a data-exchange function $\mathcal{G}$ such that for every term $t \in T_\gamma$ either $\mathcal{G}(t)$ is not a redex or a redex with respect to a ρ_i. Then every reduct of a term $t \in T_\gamma$ in $\mathcal{A}_{\mathcal{T}}(\mathcal{G})$ as again in T_γ and has therefore no production. Hence $\underline{do}_{\mathcal{T}}(\mathsf{f}) = 0$ and by definition $[\![\mathsf{f}]\!] = 0$ since $X_{\langle \mathsf{f}, i, 0 \rangle} = X_{\langle \epsilon, - \rangle}$ for all $1 \leq i \leq ar_s(\mathsf{f})$.

The remaining proof obligation is $[\![\mathsf{f}]\!] = \underline{do}_{\mathcal{T}}(\mathsf{f})$ for weakly guarded f. We start with $[\![\mathsf{f}]\!] \geq \underline{do}_{\mathcal{T}}(\mathsf{f})$. Let $p_1, \ldots, p_{ar_s(\mathsf{f})} \in \mathbb{N}$ and define $o := [\![\mathsf{f}]\!](p_1, \ldots, p_{ar_s(\mathsf{f})})$. We have $\mathcal{V}(X_{\langle \mathsf{f}, i, 0 \rangle})(p_i) = o$ for some $1 \leq i \leq ar_s(\mathsf{f})$. Note that

$$\underline{do}_{\mathcal{T}}(\mathsf{f})(p_1, \ldots, p_i, \ldots, p_{ar_s(\mathsf{f})}) \leq \underline{do}_{\mathcal{T}}(\mathsf{f})(\ldots \infty \ldots, p_i, \ldots \infty \ldots) = \underline{do}_{\mathcal{T}}(\mathsf{f}_{i, p_i}) \ .$$

Hence it suffices to show

$$\underline{do}_{\mathcal{T}}(\mathbf{g}_{i,q+p}) \leq \mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p)$$

for all $\mathbf{g} \in \Sigma_{sfun}$, argument positions $1 \leq i \leq ar_\mathbf{s}(\mathbf{g}) \in \mathbb{N}$ and $q, p \in \mathbb{N}$. For the case $\mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p) = \infty$ there is nothing to be shown, hence let $\mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p) < \infty$. Employing well-founded induction with respect to the ordering $>$ over tuples

$$\langle \mathbf{g}, i, q, p \rangle \text{ with } \mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p) < \infty$$

we construct a data-exchange function $\mathcal{G}$ together with an outermost-fair rewrite sequence in $\mathcal{A}_{\mathcal{T}}(\mathcal{G})$ starting from $\mathbf{g}_{i,p+q}$ as witness of $\underline{do}_{\mathcal{T}}(\mathbf{g}_{i,q+p}) \leq \mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p)$. The ordering $>$ is defined as follows:

$$\langle \mathbf{g}, i, q, p \rangle > \langle \mathbf{g}', i', q', p' \rangle \Longleftrightarrow \mathbf{g} \text{ is weakly-guarded}$$
$$\wedge \ (o > o' \ \vee \ (o = o' \wedge \mathbf{g} \leadsto^+ \mathbf{g}'))$$
$$\text{where } o := \mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p), \ o' := \mathcal{V}(X_{\langle \mathbf{g}', i', q' \rangle})(p')$$

Let $\mathcal{G}$ be an arbitrary data-exchange function. The induction basis consists of the tuples $\langle \mathbf{g}, i, q, p \rangle$ where

- $\mathbf{g}$ is not weakly guarded, or
- $\mathcal{V}(X_{\langle \mathbf{g}, i, q \rangle})(p) = 0$ and $\mathbf{g}$ is a normal form with respect to $\leadsto^+$.

The not weakly guarded symbols have been discussed above, therefore let $\mathbf{g}$ be a normal form with respect to $\leadsto^+$. By definition $X_{\langle \mathbf{g}, i, q \rangle} = \bigwedge_{\rho \in R^\mathbf{g}} X_{\langle \mathbf{g}, i, q, \rho \rangle}$, and hence $\mathcal{V}(X_{\langle \mathbf{g}, i, q, \rho \rangle})(p) = 0$ for some $\rho \in R_{sfun}$.

Then $X_{\langle \mathbf{g}, i, q, \rho \rangle}$ is either of shape

$$(\text{shape 1}) \ -^{in(\rho,i) \dot- q} + \ldots \qquad \text{or} \qquad (\text{shape 2}) \ -^{(in(\rho,i) \dot- q)+1} + \ldots$$

To see this consider the case distinction in the definition of $X_{\langle \mathbf{g}, i, q, \rho \rangle}$:

$$X_{\langle \mathbf{g}, i, q, \rho \rangle} = \ -^{in(\rho,i) \dot- q} +^{out(\rho)}$$
$$\begin{cases} \bigwedge_{j \in \phi^{-1}(i)} X_{\langle \mathbf{h}, j, (q \dot- in(\rho,i))+su(\rho,j) \rangle} & \text{for (a)} & (C1) \\ +^{q \dot- in(\rho,i)} X_{\langle \epsilon, -+ \rangle} & \text{for (b) if } j = i & (C2) \\ X_{\langle \epsilon, + \rangle} & \text{for (b) if } j \neq i & (C3) \end{cases}$$

We have (shape 1) in case of:

- (C1) since $\mathbf{g}$ being a $\leadsto$ normal form and therefore $out(\rho) > 0$,
- (C2) if $out(\rho) + (q \dot- in(\rho, i)) > 0$, and
- (C3)

The (shape 2) occurs in case (C2) if $out(\rho) + (q \dot- in(\rho, i)) = 0$. Let $t \equiv \mathbf{g}_{i,q+p}$ and $\mathcal{G}' = \mathcal{G}(t, \rho)$, then:

– *Assume $X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle}$ is of (shape 1).*

From $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = 0$ it follows $in(\rho,i) \mathbin{\dot{-}} q > p$ and hence $in(\rho,i) > q+p$. $\mathcal{G}'(t)$ cannot be a ρ-redex since ρ has at argument position i a consumption greater than $q+p$. Hence $\mathsf{g}_{i,q+p}$ is a normal form in $\mathcal{A}_{\mathcal{T}}(\mathcal{G}')$ by Lem. 3.9 and it follows that $\underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p}) = 0$.

– *Assume $X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle}$ is of (shape 2).*

Then we have $out(\rho) + (q \mathbin{\dot{-}} in(\rho,i)) = 0$, hence $out(\rho) = 0$ and $q \leq in(\rho,i)$. From $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = 0$ it follows that $(in(\rho,i) \mathbin{\dot{-}} q)+1 = in(\rho,i) - q + 1 > p$ and therefore $in(\rho,i) \geq q+p$. Then $\mathsf{g}_{i,q+p}$ is a normal form if $in(\rho,i) > q+p$ or $\mathsf{g}_{i,q+p} \to \sigma$ if $in(\rho,i) = q+p$ in $\mathcal{A}_{\mathcal{T}}(\mathcal{G}')$; in both cases $\underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p}) = 0$.

This concludes the base case of the induction.

Now we consider the induction step. By definition $X_{\langle \mathsf{g},\,i,\,q \rangle} = \bigwedge_{\rho \in R^{\mathsf{g}}} X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle}$, and hence $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q \rangle})(p) = \mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p)$ for some $\rho \in R_{sfun}$. Let $t \equiv \mathsf{g}_{i,q+p}$ and $\mathcal{G}' = \mathcal{G}(t,\rho)$. Again we consider the case distinction (C1),(C2) and (C3) in the definition of $X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle}$.

If $in(\rho,i) \mathbin{\dot{-}} q > p$ then in all cases $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = 0$ and $\mathsf{g}_{i,q+p}$ is a normal form in $\mathcal{A}_{\mathcal{T}}(\mathcal{G}')$, consequently $\underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p}) = 0$. Therefore assume $in(\rho,i) \mathbin{\dot{-}} q \leq p$.

(C1) $X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle} = {}^{-in(\rho,i) \mathbin{\dot{-}} q + out(\rho)} \bigwedge_{j \in \phi^{-1}(i)} X_{\langle \mathsf{h},\,j,\,(q \mathbin{\dot{-}} in(\rho,i)) + su(\rho,j) \rangle}$

There exists $j \in \phi^{-1}(i)$ with $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = e^{\mathcal{V}}(p)$ where

$$e := {}^{-in(\rho,i) \mathbin{\dot{-}} q + out(\rho)} X_{\langle \mathsf{h},\,j,\,q' \rangle} \qquad q' := (q \mathbin{\dot{-}} in(\rho,i)) + su(\rho,j) \,.$$

Since $in(\rho,i) \mathbin{\dot{-}} q \leq p$, let $p' := p - (in(\rho,i) \mathbin{\dot{-}} q)$. We get

$$\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = out(\rho) + \mathcal{V}(X_{\langle \mathsf{h},\,j,\,q' \rangle})(p') \,. \tag{5}$$

$\mathcal{G}'(\mathsf{g}_{i,q+p})$ is a ρ-redex by Lem. 3.9, hence

$$\mathsf{g}_{i,q+p} \to_{\mathcal{A}_{\mathcal{T}}(\mathcal{G}')} \bullet^{out(\rho)}\bigl(\mathsf{h}(\bullet^{n_1}(\sigma), \ldots, \bullet^{n_{ar_{\mathsf{S}}(\mathsf{h})}}(\sigma))\bigr)$$

where by definition of ρ:

$$n_j = su(\rho,j) + (q+p - in(\rho,i)) = q' + p' \,.$$

Therefore

$$\mathsf{h}(\bullet^n_1(\sigma), \ldots, \bullet^n_{ar_{\mathsf{S}}(\mathsf{h})}(\sigma)) \lhd \mathsf{h}_{j,q'+p'} \,.$$

We have $\mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) > \mathcal{V}(X_{\langle \mathsf{h},\,j,\,q',\,\rho \rangle})(p')$ or $\mathsf{g} \rightsquigarrow \mathsf{h}$ by (5), therefore we get $\underline{do}_{\mathcal{T}}(\mathsf{h}_{j,q'+p'}) \leq \mathcal{V}(X_{\langle \mathsf{h},\,j,\,q' \rangle})(p')$ by induction hypothesis and

$$\underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p}) \leq out(\rho) + \underline{do}_{\mathcal{T}}(\mathsf{h}_{j,q'+p'})$$
$$\leq out(\rho) + \mathcal{V}(X_{\langle \mathsf{h},\,j,\,q' \rangle})(p') = \mathcal{V}(X_{\langle \mathsf{g},\,i,\,q,\,\rho \rangle})(p) = \mathcal{V}(X_{\langle \mathsf{g},\,i,\,q \rangle})(p)$$

which proves the claim.

(C2) $X_{\langle \mathsf{g}, i, q, \rho \rangle} = {}^{-in(\rho,i) \dotdiv q + out(\rho) + q \dotdiv in(\rho,i)} X_{\langle \epsilon, -+ \rangle}$

Then

$$\begin{aligned} \mathcal{V}(X_{\langle \mathsf{g}, i, q, \rho \rangle})(p) &= out(\rho) + (q \dotdiv in(\rho,i)) + \mathcal{V}(X_{\langle \epsilon, -+ \rangle})(p - (in(\rho,i) \dotdiv q)) \\ &= out(\rho) + (q \dotdiv in(\rho,i)) + (p - (in(\rho,i) \dotdiv q)) \\ &= out(\rho) + q + p - in(\rho,i) \end{aligned}$$

Furthermore in case (C2) ρ is collapsing with $i = j$, therefore

$$\mathsf{g}_{i,q+p} \to_{\mathcal{A}_{\mathcal{T}}(\mathcal{G}')} \bullet^{out(\rho)+(q+p-in(\rho,i))}(\sigma) \ .$$

Hence

$$\underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p}) \leq out(\rho) + q + p - in(\rho,i) = \mathcal{V}(X_{\langle \mathsf{g}, i, q, \rho \rangle})(p) \ .$$

(C3) $X_{\langle \mathsf{g}, i, q, \rho \rangle} = {}^{-in(\rho,i) \dotdiv q + out(\rho)} X_{\langle \epsilon, + \rangle}$

Then $\mathcal{V}(X_{\langle \mathsf{g}, i, q, \rho \rangle})(p) = \infty = \underline{do}_{\mathcal{T}}(\mathsf{g}_{i,q+p})$ since ρ is collapsing with $i \neq j$.

This concludes the direction $[\![\mathsf{f}]\!] \geq \underline{do}_{\mathcal{T}}(\mathsf{f})$.

The remaining proof obligation is $[\![\mathsf{f}]\!] \leq \underline{do}_{\mathcal{T}}(\mathsf{f})$. Let $p_1, \ldots, p_{ar_{\mathsf{s}}(\mathsf{f})} \in \mathbb{N}$ and define $o := \underline{do}_{\mathcal{T}}(\mathsf{f})(p_1, \ldots, p_{ar_{\mathsf{s}}(\mathsf{f})}) = \underline{do}_{\mathcal{T}}(\mathsf{f}(\bullet^{p_1}, \ldots, \bullet^{p_{ar_{\mathsf{s}}(\mathsf{f})}}))$. Assume $o < \infty$, otherwise there is nothing to be shown. Then there exists a data-exchange function $\mathcal{G}$ and a finite reduction sequence

$$\mathsf{f}(\bullet^{p_1}(\sigma), \ldots, \bullet^{p_{ar_{\mathsf{s}}(\mathsf{f})}}(\sigma)) \to^{n}_{\mathcal{A}_{\mathcal{T}}(\mathcal{G}')} \bullet^{o}(t)$$

such that $\underline{do}_{\mathcal{T}}(t) = 0$ and in particular

- t is a normal form in $\mathcal{A}_{\mathcal{T}}(\mathcal{G}')$, or
- g is not weakly guarded.

W.l.o.g. choose $\mathcal{G}$ and the reduction sequence such that n is minimal. We apply induction on the length of the reduction n to show

- There exists $1 \leq i \leq ar_{\mathsf{s}}(\mathsf{f})$ such that

$$\underline{do}_{\mathcal{T}}(\mathsf{f}(\bullet^{p_1}(\sigma), \ldots, \bullet^{p_{ar_{\mathsf{s}}(\mathsf{f})}}(\sigma))) = \underline{do}_{\mathcal{T}}(\mathsf{f}(\bullet^{\infty}, \ldots, \bullet^{p_i}, \ldots, \bullet^{\infty})) \ ,$$

- and $[\![\mathsf{f}]\!] \leq \underline{do}_{\mathcal{T}}(\mathsf{f}(\bullet^{p_1}(\sigma), \ldots, \bullet^{p_{ar_{\mathsf{s}}(\mathsf{f})}}(\sigma)))$.

Appendix D   Soundness of the Stream Layer Translation (Lemma 5.17 and Lemma 5.18)

D.1   Proof Sketch of Lemma 5.17

We only sketch a proof of statement (i) of Lem. 5.17 because item (ii) of this lemma can be demonstrated analogously.

Proof (of Lem. 5.17, (i)). Let $\mathcal{T}$ be an SCS, and let $\mathcal{F} = \{\gamma_f\}_{f \in \Sigma_{sfun}}$ be a family of gates such that, for all $f \in \Sigma_{sfun}$, $[\![\gamma_f]\!] = \underline{do}_{\mathcal{T}}(f)$. We give the ideas for the proofs of the inequations "$\leq$" and "$\geq$" that make part of the equation

$$\underline{do}_{\mathcal{T}}(M_0) = [\![[M_0]^{\mathcal{F}}]\!] \, . \tag{6}$$

We assume, without loss of generality, that every $f \in \Sigma_{sfun}$ has precisely one occurrence in the SCS-layer. Note that, by replacing for every $f \in \Sigma_{sfun}$ the individual occurrences of f in the SCS-layer of $\mathcal{T}$ by individual occurrences of the symbols $f, f', f'', \ldots$, respectively, and introducing defining rules for f', f'', $\ldots$ in the SFS-part analogous to the defining rules for f that are already present there, an SCS $\mathcal{T}'$ is obtained that satisfies the restriction, and for which it is easy to establish that it holds: $\underline{do}_{\mathcal{T}}(M_0) = \underline{do}_{\mathcal{T}'}(M_0)$, $[M_0]^{\mathcal{F}} = [M_0]^{\mathcal{F}'}$, and hence $[\![[M_0]^{\mathcal{F}}]\!] = [\![[M_0]^{\mathcal{F}'}]\!]$, where $\mathcal{F}' = \{\gamma_f\}_{f \in \Sigma'_{sfun}}$ is the family of gates (Σ'_{sfun} is the stream function signature of $\mathcal{T}'$) by adding to $\mathcal{F}$, for all symbols $f \in \Sigma_{sfun}$, the corresponding copies $\gamma_{f'}, \gamma_{f''}, \ldots$ of γ_f. Note that we also have, for all $f' \in \Sigma'_{sfun}$, $[\![\gamma_{f'}]\!] = \underline{do}_{\mathcal{T}}(f')$.

"$\leq$": We only have to consider the case in which for $k := [\![[M_0]^{\mathcal{F}}]\!]$ it holds $k < \infty$. Hence we assume that $k < \infty$. Then there exists a finite rewrite sequence

$$\rho : \ [M_0]^{\mathcal{F}} \twoheadrightarrow_{\mathsf{P}} \bullet^k(p') \, .$$

Then with $p := [M_0]^{\mathcal{F}}$ it follows that p' has the same 'gate structure' as p, but possibly the io-sequences in the gates have changed, and there are likely to appear many pebbles queuing inside p'. By the definition of k it follows that $[\![p']\!] = 0$. Without loss of generality, we assume that during ρ all extractable pebbles have been extracted from 'inside' the gates in p', that is, there is no occurrence of a redex in p' with respect to the pebbleflow rule $\min(\bullet(p_1), \bullet(p_2)) \to \bullet(\min(p_1, p_2))$. Note that the only occurrences of the symbol $\min$ in terms of ρ are part of gates. (If this would not be the case for ρ, then we would extend ρ by finitely many $\to_{\mathsf{P}}$-steps to a rewrite sequence ρ' which has this property, and base the further argumentation on ρ'.)
Due to the assumption that function symbols in Σ_{sfun} only occur once in the SCS-layer, to every $f \in \Sigma_{sfun}$ there corresponds in p (via the net translation in Def. 5.16) a unique gate $\gamma_f = [f]$. We now monitor, for every gate γ_f in p, the consumption/production behaviour during ρ. Let $f \in \Sigma_{sfun}$. Suppose that during ρ, γ_f consumes $n_1, \ldots, n_{ar_s(f)}$ pebbles from its input components, respectively, and that in total it produces n pebbles. By the assumption

that no extractable pebbles are stuck inside γ_f at the end of ρ, in particu-lar, $[\![\gamma_f]\!](n_1, \ldots, n_{ar_s(f)}) = n$ follows. Moreover, by the assumption that the data-oblivious lower bound $\underline{do}_{\mathcal{T}}(f)$ of a symbol $f \in \Sigma_{sfun}$ coincides with the production function of the gate γ_f, we get $\underline{do}_{\mathcal{T}}(f)(n_1, \ldots, n_{ar_s(f)}) = n$. As a consequence, there is a rewrite sequence:

$$\tilde{\rho}_f : f(\bullet^{n_1} : \sigma_1, \ldots, \bullet^{n_{ar_s(f)}} : \sigma_{ar_s(f)}, \underbrace{\bullet, \ldots, \bullet}_{ar_d(f)}) \twoheadrightarrow_{\mathcal{A}_{\mathcal{T},\mathcal{G}}} \bullet^n : t$$

in an ARS $\mathcal{A}_{\mathcal{T},\mathcal{G}}$ such that $\tilde{\rho}_f$ can be extended to an outermost-fair rewrite sequence that does not produce further pebbles. Then from $\tilde{\rho}_f$ we extract a rational gate $\tilde{\gamma}_f$ with the properties: (a) $\tilde{\gamma}_f$ precisely describes the con-sumption/production behaviour that happens during $\tilde{\rho}_f$, (b) thereafter $\tilde{\gamma}_f$ continues like the gate corresponding to f in p'. We gather the new gates in the new family of gates: $\tilde{\mathcal{F}} := \{\tilde{\gamma}_f\}_{f \in \Sigma_{sfun}}$. Note that, for all $f \in \Sigma_{sfun}$, $[\![\gamma_f]\!] \le [\![\tilde{\gamma}_f]\!]$ holds. Nevertheless, it can be shown that there is actually also a rewrite sequence:

$$\tilde{\rho} : [M_0]^{\tilde{\mathcal{F}}} \twoheadrightarrow_P \bullet^k(\tilde{p}') .$$

such that $[\![\tilde{p}']\!] = 0$. Using this pebbleflow rewrite sequence $\tilde{\rho}$ as well as the rewrite sequences $\tilde{\rho}_f$ on the abstracted versions of $\mathcal{T}$, it is possible to obtain a rewrite sequence in an ARS $\mathcal{A}_{\mathcal{T},\mathcal{G}}$ of the form:

$$\bar{\rho} : \boldsymbol{T}_{\mathcal{T}}(M_0) \twoheadrightarrow_{\mathcal{A}_{\mathcal{T},\mathcal{G}}}^{m\omega} \bullet^k : s ,$$

where m is the length of the reduction sequence $\tilde{\rho}$, that can be extended in $\mathcal{A}_{\mathcal{T},\mathcal{G}}$ to a rewrite sequence $\bar{\bar{\rho}}$ which consists of outermost-fair sequences of rewrite sequences of length ω in which respectively, for some $g \in \Sigma_{sfun}$, g-redexes are reduced uniformously in all unfoldings, and such that further-more $\bar{\bar{\rho}}$ does not contain terms with a prefix of more than k pebbles. The existence of the rewrite sequence $\bar{\bar{\rho}}$ on M_0 in an ARS $\mathcal{A}_{\mathcal{T},\mathcal{G}}$ demonstrates that $\underline{do}_{\mathcal{T}}(M_0) \le k$.

"$\ge$": This direction of the proof is a consequence of three statements that are mentioned below. First, it holds that:

$$\text{For all } p \in \mathcal{P}: \quad [\![\boldsymbol{T}(p)]\!] = [\![p]\!] , \tag{7}$$

where $\boldsymbol{T}(p)$ denotes the infinite unfolding of p into a possibly infinite 'peb-bleflow tree'. Second:

$$\text{For all stream terms } s \text{ in } \mathcal{T}: \quad \Pi_{\mathcal{T}}(s) = \Pi_{\mathcal{T}}(\boldsymbol{T}_{\mathcal{T}}(s)) , \tag{8}$$

where by $s \in Ter(\Sigma)$ we mean the result of infinitely unfolding all stream constants in s. Finally, third, it holds that:

$$\text{For all stream terms } s \text{ in } \mathcal{T}: \quad \underline{do}_{\mathcal{T}}(\boldsymbol{T}_{\mathcal{T}}(s)) \ge [\![\boldsymbol{T}([s]^{\mathcal{F}})]\!] . \tag{9}$$

Putting (7), (8), and (9) together, we obtain:

$$\underline{do}_{\mathcal{T}}(M_0) = \underline{do}_{\mathcal{T}}(\boldsymbol{T}_{\mathcal{T}}(M_0)) \ge [\![\boldsymbol{T}([M_0]^{\mathcal{F}})]\!] = [\![[M_0]^{\mathcal{F}}]\!] ,$$

which shows "$\ge$".

This concludes our sketch of the proof of Lem. 5.17 (i).

D.2   Proof of Lemma 5.18

Let $\mathcal{T}$ be a stream specification, and let $\mathcal{F} = \{\gamma_f\}_{f \in \Sigma_{sfun}}$ be a family of gates such that, for all $f \in \Sigma_{sfun}$, the arity of γ_f equals the stream arity of f. Suppose that one of the following statements holds:

(a) $\llbracket \gamma_f \rrbracket \leq \underline{do}_{\mathcal{T}}(f)$ for all $f \in \Sigma_{sfun}$;
(b) $\llbracket \gamma_f \rrbracket \geq \underline{do}_{\mathcal{T}}(f)$ for all $f \in \Sigma_{sfun}$;
(c) $\overline{do}_{\mathcal{T}}(f) \leq \llbracket \gamma_f \rrbracket$ for all $f \in \Sigma_{sfun}$;
(d) $\overline{do}_{\mathcal{T}}(f) \geq \llbracket \gamma_f \rrbracket$ for all $f \in \Sigma_{sfun}$.

In this section we show that then, for all $M \in \Sigma_{scon}$, the corresponding one of the following statements holds:

(a) $\llbracket [M]^{\mathcal{F}} \rrbracket \leq \underline{do}_{\mathcal{T}}(M)$;
(b) $\llbracket [M]^{\mathcal{F}} \rrbracket \geq \underline{do}_{\mathcal{T}}(M)$;
(c) $\overline{do}_{\mathcal{T}}(M) \leq \llbracket [M]^{\mathcal{F}} \rrbracket$;
(d) $\overline{do}_{\mathcal{T}}(M) \geq \llbracket [M]^{\mathcal{F}} \rrbracket$.

Multiple Numbered Contexts

Definition D.1. *Let* $\mathcal{T} = \langle \Sigma, R \rangle$ *be a stream specification, and* $k, l \in \mathbb{N}$. *A (multiple numbered)* stream context $C[]$ *over stream holes* $\square_1, \ldots, \square_k$, *and data holes* $\boxdot_1, \ldots, \boxdot_\ell$ *is a (stream) term*

$$C[] \in \mathit{Ter}_\infty(\Sigma \uplus \{\square_1, \ldots, \square_k\} \uplus \{\boxdot_1, \ldots, \boxdot_\ell\}, X)_S$$

where the $\square_i :: S$ *and* $\boxdot_i :: D$ *are distinct constant symbols not occurring in* Σ.
   We denote by $C[t_1, \ldots, t_k; u_1, \ldots, u_\ell]$ *the result of replacing the occurrences of* $\square_i$ *by* s_i *and* $\boxdot_j$ *by* t_j *in* $C[]_{\mathcal{T}}$, *respectively.*

Remark D.2. Each of the hole symbols $\square_1, \ldots, \square_k$, and $\boxdot_1, \ldots, \boxdot_\ell$ may have multiple occurrences in a stream context $C[]$ over $\square_1, \ldots, \square_k$ and $\boxdot_1, \ldots, \boxdot_\ell$, and this includes the case that it does not occur in $C[]$ at all.

Example D.3. Let $\mathcal{T}$ be a stream specification with data constants b and c, a unary stream function symbol f, and a binary stream function symbol g. Then the expression $b : f(c : g(\boxdot_1 : \square_1, g(\square_2, g(\square_1, \square_3))))$ is a stream context over $\square_1$, $\square_2$, $\square_3$ and $\boxdot_1$. Also $f(b : \square_3)$ is a stream context over $\square_1$, $\square_2$, $\square_3$ and $\boxdot_1$. But $b : \square_1 : \sigma$ is not a stream context over $\square_1$, because the symbol $\square_1$ occurs at a 'data position' (this is excluded by many-sortedness, $: :: D \to S \to S$ and $\square_1 :: S$).

   We extend the d-o production function of stream function symbols $f \in \Sigma_{sfun}$, Def. 3.5, to multiple numbered contexts.

Definition D.4. Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification, and $C[]$ a multiple numbered stream context over stream holes $\square_1, \ldots, \square_k$, and data holes $\boxdot_1, \ldots, \boxdot_\ell$. The *d-o production range* $\overline{\underline{do}}_{\mathcal{T}}(C[]) : \mathbb{N}^k \to \overline{\mathbb{N}}$ *of the context $C[]$* is:

$$\overline{\underline{do}}_{\mathcal{T}}(C[])(n_1, \ldots, n_k) := \overline{\underline{do}}_{\mathcal{T}}(\, (\!|C[]|\!)((\bullet^{n_1} : \sigma), \ldots, (\bullet^{n_k} : \sigma))) \,,$$

where $\bullet^m : \sigma := \overbrace{\bullet : \ldots : \bullet :}^{m \text{ times}} \sigma$.

The *d-o lower and upper bounds on the production of* g are defined by $\underline{do}_{\mathcal{T}}(C[]) := \inf(\overline{\underline{do}}_{\mathcal{T}}(C[]))$ and $\overline{do}_{\mathcal{T}}(C[]) := \sup(\overline{\underline{do}}_{\mathcal{T}}(C[]))$, respectively.

Note that it is indeed an extension of Def. 3.5, for every $\mathsf{g} \in \Sigma_{sfun}$ we have

$$\overline{\underline{do}}_{\mathcal{T}}(\mathsf{g}) = \overline{\underline{do}}_{\mathcal{T}}(\mathsf{g}(\square_1, \ldots, \square_{ar_s(\mathsf{g})}, \boxdot_1, \ldots, \boxdot_{ar_d(\mathsf{g})})) \,.$$

The following lemma states that data-oblivious lower and upper bounds are reachable.

Lemma D.5. *Let $\mathcal{T} = \langle \Sigma, R \rangle$ be a stream specification, and $s \in Ter_\infty(\Sigma)_S$.*

Colouring Terms, Tracing Data Flow For the purpose of tracing the flow of data-elements during rewrite sequences, we introduce auxiliary 'tracker symbols' and partition terms into 'coloured' multiple numbered contexts. Usual rewriting will then be allowed only within contexts. Rewrite steps crossing the border between contexts are prohibited. In order to circumvent the loss of possible reduction steps, we introduce rules for exchanging pebbles between neighboring contexts (recolouring of the pebbles).

We introduce auxiliary, unary function symbols $\diamond_c$ and $\otimes_c$ in the pebbleflow rewrite system as well as in the stream specification $\mathcal{T}$. The symbols $\diamond_c$ mark the top of a context of colour c and at the same time terminate the context above it. The symbols $\otimes_c$ only close, but do not open a context. We call them *tracker symbols* in the sequel, since they are also used to track pebble/data exchange between neighboring contexts. Therefore we enrich the pebbleflow rewrite system with the following rules

$$\diamond_c(\bullet(p)) \to \bullet(\diamond_c(p)) \qquad\qquad \otimes_c(\bullet(p)) \to \bullet(\otimes_c(p)) \qquad\qquad (10)$$

and the stream specification $\mathcal{T}$ with rules

$$\diamond_c(t : \sigma) \to t : \diamond_c(\sigma) \qquad\qquad \otimes_p(t : \sigma) \to t : \otimes_c(\sigma) \qquad\qquad (11)$$

for every function symbol $\diamond_c$ and $\otimes_c$. We define $\diamond_c, \otimes_c :: \mathsf{stream} \to \mathsf{stream}$, that is, the tracker symbols cannot be placed at data positions.

Definition D.6. For terms s, t we say that *t is an enrichment of s with tracker symbols*, denoted $s \angle t$, if s can be obtained from t by removing tracker symbols, that is, $t \to^* s$ via rules $\diamond_c(u) \to u$ and $\otimes_c(u) \to u$.

Lemma D.7. *Every pebbleflow reduction can be lifted to enrichted terms:*

$$\forall s, s' \in \mathcal{P}, s \twoheadrightarrow_\mathsf{P} s', s \angle t. \exists t'. t \twoheadrightarrow_{\mathsf{P}\diamond} t' \wedge s' \angle t'$$

and stream specifications $\mathcal{T}$

Every pebbleflow rewrite sequence and every stream specification rewrite sequence can be lifted to enriched terms, i.e. $\forall s' \twoheadleftarrow s \angle t. \exists t'. t \rightsquigarrow^* t' \wedge s' \angle t'$ where $\rightsquigarrow \; = \; \rightarrow \cup \rightarrow_{(10)}$ (or $\rightsquigarrow \; = \; \rightarrow \cup \rightarrow_{(11)}$). Sometimes the tracker symbols are in the way of rule application, but, because of the special structure of pebbleflow (resp. SCS) rules, it is always possible to move them away using rules (10) (resp. (11)).

Now we sketch the proof for the inequalities "$\leq$" (i) and "$\geq$" (ii).

"$\leq$": We need to show $[\![[M]^{\mathcal{F}}]\!] \leq \Pi_{\mathcal{T}}(M)$. For the proof of (i) we map a given finite pebbleflow rewrite sequence on $[M]^{\mathcal{F}}$ producing n pebbles to a $\mathcal{T}$ rewrite sequence on the infinite unfolding of M that produces at least n data elements. (For (ii) the proof proceeds the other way arround.)

Thereby not every rewrite step will be mapped, but we trace 'essential' steps and map them to a many-step (possibly infinite) reduction on 'the other side'. For this purpose we partition the pebbleflow term $[M]^{\mathcal{F}}$ as well as infinite unfolding of M into coloured contexts (using the same colours for the net and the unfolding of M). The partitioning is arranged in such a way that contexts of the same colour have the same production function. In the sequel we will refer to this property as the *context production property* (CPP).

Usual rewriting will then be allowed only within contexts. Rewrite steps crossing the border between contexts are prohibited. In order to circumvent the loss of possible reduction steps, we introduce additional rules for exchanging pebbles between neighboring contexts (recolouring of the pebbles). The construction of the mapped reduction sequence will be driven by the goal to uphold CPP. It can be shown pebbleflow rewriting and SCS rewriting does not change the production function of a context. Hence in order to maintain CPP it is not necessary to map internal steps ocurring within contexts. The 'essential' steps are the exchange steps between the contexts.

We use C_c^{P} to refer to the context of colour c in the pebbleflow net $[M]^{\mathcal{F}}$. Likewise we use C_c^{T} for the context of colour c in the infinite unfolding of the stream constant M. Although there may be multiple, typically infinitely many, occurrences of a context with colour c on the term level, contexts with the same colour are syntactically equal. Initially all contexts are of one of the following (simple) shapes:

Clearly $[\![\_]\!]_{C_c^{\mathsf{P}}} = [\![\_]\!]_{C_c^{\mathsf{T}}}$ for case 1. An important step in the proof is to show that the translation of functions symbols into rational gates preserves the quantitative production behaviour, yielding $[\![\_]\!]_{C_c^{\mathsf{P}}} = [\![\_]\!]_{C_c^{\mathsf{T}}}$ for case 2.

The proof now continues as follows. Every pebble exchange step in the pebbleflow net is mapped to an infinite rewrite sequence on the corresponding coloured SCS term. For the definition of these corresponding rewrite steps we apply the assumption $[\![\_]\!]_{C_c^{\mathsf{P}}} = [\![\_]\!]_{C_c^{\mathsf{T}}}$ guaranteeing that a data element

case	C_c^{P}	C_c^{T}
1	$\bullet(\square_1)$	$d : \square_1$
2	$\min(\sigma_{\mathsf{f},1}(\square_1), \ldots, \sigma_{\mathsf{f},r}(\square_r))$	$\mathsf{f}(\square_1, \ldots, \square_r)$

Fig. 15: Initial context configurations

can also be extracted from the context on SCS term level, and that after the step we have again $[\![\_]\!]_{C_c^{\mathsf{P}}} = [\![\_]\!]_{C_c^{\mathsf{T}}}$ for all colours c. In this way, we define a rewrite sequence $\rho_{\mathcal{T}}$ in $\mathcal{T}$ by induction on the number of rewrite steps of the given pebbleflow reduction ρ such that $\rho_{\mathcal{T}}$ and ρ produce the same amount of data elements and pebbles, respectively.

"$\geq$": Here we need to show $[\![[M]^{\mathcal{F}}]\!] \geq \Pi_{\mathcal{T}}(M)$. The proof of this inequality proceeds similarly to that of "$\leq$". Let $\rho_{\mathcal{T}}$ in $\mathcal{T}$ be a reduction sequence in $\mathcal{T}$. In order to map $\rho_{\mathcal{T}}$ back to a pebbleflow rewrite sequence ρ it is crucial that the contexts of one colour are changed synchronously and in the same way. It cannot be assumed that $\rho_{\mathcal{T}}$ posesses this property. For this purpose $\rho_{\mathcal{T}}$ is transformed into a sequence $\rho'_{\mathcal{T}}$ of complete developments. This is always possible since complete developments in orthogonal TRSs are a cofinal rewrite strategy. For mapping $\rho'_{\mathcal{T}}$ to a rewrite sequence ρ with the same production on the pebbleflow net, we can then proceed by induction on the length of $\rho'_{\mathcal{T}}$, in converse direction to the argument for "$\leq$", above.