HAL open science

# Bisimilarity of Open Terms in Stream GSOS

Filippo Bonchi, Matias David Lee, Jurriaan Rot

## To cite this version:

Filippo Bonchi, Matias David Lee, Jurriaan Rot. Bisimilarity of Open Terms in Stream GSOS. 7th International Conference on Fundamentals of Software Engineering (FSEN), Apr 2017, Teheran, Iran. pp.35-50, 10.1007/978-3-319-68972-2_3 . hal-01760851

**HAL Id: hal-01760851**

**https://inria.hal.science/hal-01760851v1**

Submitted on 6 Apr 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

# Bisimilarity of Open Terms in Stream GSOS*

Filippo Bonchi[1], Matias David Lee[1] and Jurriaan Rot[2]

[1] Univ Lyon, ENS de Lyon, CNRS, UCB Lyon 1, LIP, France
[filippo.bonchi,matias-david.lee]@ens-lyon.fr
[2] Radboud University, Nijmegen, The Netherlands
jrot@cs.ru.nl

**Abstract.** Stream GSOS is a specification format for operations and calculi on infinite sequences. The notion of bisimilarity provides a canonical proof technique for equivalence of closed terms in such specifications. In this paper, we focus on *open terms*, which may contain variables, and which are equivalent whenever they denote the same stream for every possible instantiation of the variables. Our main contribution is to capture equivalence of open terms as bisimilarity on certain Mealy machines, providing a concrete proof technique. Moreover, we introduce an enhancement of this technique, called bisimulation up-to substitutions, and show how to combine it with other up-to techniques to obtain a powerful method for proving equivalence of open terms.

## 1 Introduction

Structural operational semantics (SOS) can be considered the de facto standard to define programming languages and process calculi. The SOS framework relies on defining a *specification* consisting of a set of operation symbols, a set of labels or actions and a set of inference rules. The inference rules describe the behaviour of each operation, typically depending on the behaviour of the parameters. The semantics is then defined in terms of a labelled transition system over *(closed) terms* constructed from the operation symbols. *Bisimilarity* of closed terms ($\sim$) provides a canonical notion of behavioural equivalence.

It is also interesting to study equivalence of *open terms*, for instance to express properties of program constructors, like the commutativity of a non-deterministic choice operator. The latter can be formalised as the equation $\mathcal{X} + \mathcal{Y} = \mathcal{Y} + \mathcal{X}$, where the left and right hand sides are terms with variables $\mathcal{X}, \mathcal{Y}$. Equivalence of open terms ($\sim_o$) is usually based on $\sim$: for all open terms $t_1, t_2$

$$t_1 \sim_o t_2 \text{ iff for all closed substitutions } \phi, \;\; \phi(t_1) \sim \phi(t_2). \tag{1}$$

The main problem of such a definition is the quantification over all substitutions: one would like to have an alternative characterisation, possibly amenable to the

* The research leading to these results has received funding from the European Research Council (FP7/2007-2013, grant agreement nr. 320571; as well as from the LABEX MILYON (ANR-10-LABX-0070, ANR-11-IDEX-0007), the project PACE (ANR-12IS02001) and the project REPAS (ANR-16-CE25-0011).

coinduction proof principle. This issue has been investigated in several works, like [7, 15, 3, 1, 11, 13, 20].

$$
\begin{array}{lllll}
(a) & \dfrac{}{n \xrightarrow{n} 0} & \dfrac{x \xrightarrow{n} x' \quad y \xrightarrow{m} y'}{x \oplus y \xrightarrow{n+m} x' \oplus y'} & \dfrac{x \xrightarrow{n} x' \quad y \xrightarrow{m} y'}{x \otimes y \xrightarrow{n \times m} (n \otimes y') \oplus (x' \otimes y)} & \\
\hline
(b) & \dfrac{}{n \xrightarrow{n} 0} & \dfrac{x \xrightarrow{n} x' \quad y \xrightarrow{m} y'}{x \oplus y \xrightarrow{n+m} x' \oplus y'} & \dfrac{x \xrightarrow{n} x' \quad y \xrightarrow{m} y'}{x \otimes y \xrightarrow{n \times m} (n \otimes y') \oplus (x' \otimes m.y')} & \dfrac{x \xrightarrow{m} x'}{n.x \xrightarrow{n} m.x'} \\
\hline
(c) & \dfrac{}{n \xrightarrow{b|n} 0} & \dfrac{x \xrightarrow{b|n} x' \quad y \xrightarrow{b|m} y'}{x \oplus y \xrightarrow{b|n+m} x' \oplus y'} & \dfrac{x \xrightarrow{b|n} x' \quad y \xrightarrow{b|m} y'}{x \otimes y \xrightarrow{b|n \times m} (n \otimes y') \oplus (x' \otimes m.y')} & \dfrac{x \xrightarrow{b|m} x'}{n.x \xrightarrow{b|n} m.x'} \\
\hline
(d) & \dfrac{}{\mathcal{X} \xrightarrow{\varsigma|\varsigma(\mathcal{X})} \mathcal{X}} & \dfrac{}{n \xrightarrow{\varsigma|n} 0} \quad \cdots & \dfrac{x \xrightarrow{\varsigma|n} x' \quad y \xrightarrow{\varsigma|m} y'}{x \otimes y \xrightarrow{\varsigma|n \times m} (n \otimes y') \oplus (x' \otimes m.y')} & \dfrac{x \xrightarrow{\varsigma|m} x'}{n.x \xrightarrow{\varsigma|n} m.x'}
\end{array}
$$

**Figure 1.** A stream GSOS specification (a) is transformed first into a monadic specification (b), then in a Mealy specification (c) and finally in a specification for open terms (d). In these rules, $n$ and $m$ range over real numbers, $b$ over an arbitrary set $B$, $\mathcal{X}$ over variables and $\varsigma$ over substitutions of variables into reals.

In this paper, we continue this line of research, focusing on the simpler setting of *streams*, which are infinite sequences over a fixed data type. More precisely, we consider stream languages specified in the *stream GSOS format* [10], a syntactic rule format enforcing several interesting properties. We show how to transform a stream specification into a *Mealy machine* specification that defines the operational semantics of open terms. Moreover, a notion of bisimulation – arising in a canonical way from the theory of coalgebras [16] – exactly characterises $\sim_o$ as defined in (1).

Our approach can be illustrated by taking as running example the fragment of the stream calculus [18] presented in Figure 1(a). The first step is to transform a stream GSOS specification (Section 2) into a *monadic* one (Section 3). In this variant of GSOS specifications, no variable in the source of the conclusion appears in the target of the conclusion. For example, in the stream specification in Figure 1(a), the rule associated to $\otimes$ is not monadic. The corresponding monadic specification is illustrated in Figure 1(b). Notice this process requires the inclusion of a family of prefix operators (on the right of Figure 1(b)) that satisfy the imposed restriction.

The second step – based on [8] – is to compute the *pointwise extension* of the obtained specification (Section 4). Intuitively, we transform a specification of streams with outputs in a set $A$ into a specification of Mealy machines with inputs in an arbitrary set $B$ and outputs in $A$, by replacing each transition $\xrightarrow{a}$ (for $a \in A$) with a transition $\xrightarrow{b|a}$ for each input $b \in B$. See Figure 1(c).

In the last step (Section 5), we fix $B = \mathcal{V} \to A$, the set of functions assigning outputs values in $A$ to variables in $\mathcal{V}$. To get the semantics of open terms, it only remains to specify the behaviours of variables in $\mathcal{V}$. This is done with the leftmost rule in Figure 1(d).

As a result of this process, we obtain a notion of bisimilarity over open terms, which coincides with behavioural equivalence of all closed instances, and provides a concrete proof technique for equivalence of open terms. By relating open terms rather than all its possible instances, this novel technique often enables to use *finite* relations, while standard bisimulation techniques usually require relations of infinite size on closed terms. In Section 6 we further enhance this novel proof technique by studying *bisimulation up-to* [14]. We combine known up-to techniques with a novel one which we call *bisimulation up-to substitutions*.

## 2 Preliminaries

We define the two basic models that form the focus of this paper: stream systems, that generate infinite sequences (streams), and Mealy machines, that generate output streams given input streams.

**Definition 2.1.** *A* stream system *with outputs in a set $A$ is a pair $(X, \langle o, d\rangle)$ where $X$ is a set of states and $\langle o, d\rangle \colon X \to A \times X$ is a function, which maps a state $x \in X$ to both an* output value *$o(x) \in A$ and to a* next state *$d(x) \in X$. We write $x \xrightarrow{a} y$ whenever $o(x) = a$ and $d(x) = y$.*

**Definition 2.2.** *A* Mealy machine *with inputs in a set $B$ and outputs in a set $A$ is a pair $(X, m)$ where $X$ is a set of states and $m \colon X \to (A \times X)^B$ is a function assigning to each $x \in X$ a map $m(x) = \langle o_x, d_x\rangle \colon B \to A \times X$. For all inputs $b \in B$, $o_x(b) \in A$ represents an output and $d_x(b) \in X$ a next state. We write $x \xrightarrow{b|a} y$ whenever $o_x(b) = a$ and $d_x(b) = y$.*

We recall the notion of *bisimulation* for both models.

**Definition 2.3.** *Let $(X, \langle o, d\rangle)$ be a stream system. A relation $\mathcal{R} \subseteq X \times X$ is a* bisimulation *if for all $(x, y) \in \mathcal{R}$, $o(x) = o(y)$ and $(d(x), d(y)) \in \mathcal{R}$.*

**Definition 2.4.** *Let $(X, m)$ be a Mealy machine. A relation $\mathcal{R} \subseteq X \times X$ is a* bisimulation *if for all $(x, y) \in \mathcal{R}$ and $b \in B$, $o_x(b) = o_y(b)$ and $(d_x(b), d_y(b)) \in \mathcal{R}$.*

For both kind of systems, we say that $x$ and $y$ are *bisimilar*, notation $x \sim y$, if there is a bisimulation $\mathcal{R}$ s.t. $x \mathrel{\mathcal{R}} y$.

Stream systems and Mealy machines, as well as the associated notions of bisimulation, are instances of the theory of *coalgebras* [16]. Coalgebras provide a suitable mathematical framework to study state-based systems and their semantics at a high level of generality. In the current paper, the theory of coalgebras underlies and enables our main results.

**Definition 2.5.** *Given a functor $F\colon Set \to Set$, an $F$-*coalgebra *is a pair $(X, d)$, where $X$ is a set (called the carrier) and $d\colon X \to FX$ is a function (called the structure). An $F$-*coalgebra morphism *from $d\colon X \to FX$ to $d'\colon Y \to FY$ is a map $h\colon X \to Y$ such that $Fh \circ d = d' \circ h$.*

Stream systems and Mealy machines are $F$-coalgebras for the functors $FX = A \times X$ and $FX = (A \times X)^B$, respectively.

The semantics of systems modelled as coalgebras for a functor $F$ is provided by the notion of *final coalgebra*. A coalgebra $\zeta\colon Z \to FZ$ is called *final* if for every $F$-coalgebra $d\colon X \to FX$ there is a unique morphism $[\![-]\!]\colon X \to Z$ such that $[\![-]\!]$ is a morphism from $d$ to $\zeta$. We call $[\![-]\!]$ the *coinductive extension* of $d$.

Intuitively, a final coalgebra $\zeta\colon Z \to FZ$ defines all possible behaviours of $F$-coalgebras, and $[\![-]\!]$ assigns behaviour to all states $x, y \in X$. This motivates to define $x$ and $y$ to be *behaviourally equivalent* iff $[\![x]\!] = [\![y]\!]$. Under the condition that $F$ preserves weak pullbacks, behavioural equivalence coincides with bisimilarity, i.e., $x \sim y$ iff $[\![x]\!] = [\![y]\!]$ (see [16]). This condition is satisfied by (the functors for) stream systems and Mealy machines. In the sequel, by $\sim$ we hence refer both to bisimilarity and behavioural equivalence.

Final coalgebras for stream systems and Mealy machines will be pivotal for our exposition. We briefly recall them, following [16, 9]. The set $A^\omega$ of streams over $A$ carries a final coalgebra for the functor $FX = A \times X$. For every stream system $\langle o, d\rangle\colon X \to A \times X$, the coinductive extension $[\![-]\!]\colon X \to A^\omega$ assigns to a state $x \in X$ the stream $a_0 a_1 a_2 \ldots$ whenever $x \xrightarrow{a_0} x_1 \xrightarrow{a_1} x_2 \xrightarrow{a_2} \ldots$

Recalling a final coalgebra for Mealy machines requires some more care. Given a stream $\beta \in B^\omega$, we write $\beta{\restriction}_n$ for the prefix of $\beta$ of length $n$. A function $\mathfrak{c}\colon B^\omega \to A^\omega$ is *causal* if for all $n \in \mathbb{N}$ and all $\beta, \beta' \in B^\omega$: $\beta{\restriction}_n = \beta'{\restriction}_n$ entails $\mathfrak{c}(\beta){\restriction}_n = \mathfrak{c}(\beta'){\restriction}_n$. The set $\Gamma(B^\omega, A^\omega) = \{\mathfrak{c}\colon B^\omega \to A^\omega \mid \mathfrak{c} \text{ is causal}\}$ carries a final coalgebra for the functor $FX = (A \times X)^B$. For every Mealy machine $m\colon X \to (A \times X)^B$, the coinductive extension $[\![-]\!]\colon X \to \Gamma(B^\omega, A^\omega)$ assigns to each state $x \in X$ and each input stream $b_0 b_1 b_2 \cdots \in B^\omega$ the output stream $a_0 a_1 a_2 \cdots \in A^\omega$ whenever $x \xrightarrow{b_0|a_0} x_1 \xrightarrow{b_1|a_1} x_2 \xrightarrow{b_2|a_2} \ldots$

### 2.1 System Specifications

Different kinds of transition systems, like stream systems or Mealy machines, can be specified by means of algebraic specification languages. The syntax is given by an *algebraic signature* $\Sigma$, namely a collection of operation symbols $\{f_i \mid i \in I\}$ where each operator $f_i$ has a (finite) arity $n_i \in \mathbb{N}$. For a set $X$, $T_\Sigma X$ denotes the set of $\Sigma$-terms with variables over $X$. The set of closed $\Sigma$-terms is denoted by $T_\Sigma \emptyset$. We omit the subscript when $\Sigma$ is clear from the context.

A standard way to define the operational semantics of these languages is by means of structural operational semantics (SOS) [12]. In this approach, the semantics of each of the operators is described by syntactic rules, and the behaviour of a composite system is given in terms of the behaviour of its components. We recall *stream GSOS* [10], a specification format for stream systems.

**Definition 2.6.** *A* stream GSOS rule *$r$ for a signature $\Sigma$ and a set $A$ is a rule*

$$\frac{x_1 \xrightarrow{a_1} x_1' \quad \cdots \quad x_n \xrightarrow{a_n} x_n'}{f(x_1, \ldots, x_n) \xrightarrow{a} t} \tag{2}$$

*where $f \in \Sigma$ with arity $n$, $x_1, \ldots, x_n, x_1', \ldots, x_n'$ are pairwise distinct variables, $t$ is a term built over variables $\{x_1, \ldots, x_n, x_1', \ldots, x_n'\}$ and $a, a_1, \ldots, a_n \in A$. We say that $r$ is triggered by $(a_1, \ldots, a_n) \in A^n$.*

*A* stream GSOS specification *is a tuple $(\Sigma, A, R)$ where $\Sigma$ is a signature, $A$ is a set of actions and $R$ is a set of stream GSOS rules for $\Sigma$ and $A$ s.t. for each $f \in \Sigma$ of arity $n$ and each tuple $(a_1, \ldots, a_n) \in A^n$, there is only one rule $r \in R$ for $f$ that is triggered by $(a_1, \ldots, a_n)$.*

A stream GSOS specification allows us to extend any given stream system $\langle o, d \rangle \colon X \to A \times X$ to a stream system $\overline{\langle o, d \rangle} \colon TX \to A \times TX$, by induction: the base case is given by $\langle o, d \rangle$, and the inductive cases by the specification. This construction can be defined formally in terms of proof trees, or by coalgebraic means; we adopt the latter approach, which is recalled later in this section.

There are two important uses of the above construction: (A) applying it to the (unique) stream system carried by the empty set $\emptyset$ yields a stream system over closed terms, i.e., of the form $T\emptyset \to A \times T\emptyset$; (B) applying the construction to the final coalgebra yields a stream system of the form $TA^\omega \to A \times TA^\omega$. The coinductive extension $[\![-]\!] \colon TA^\omega \to A^\omega$ of this stream system is, intuitively, the interpretation of the operations in $\Sigma$ on streams in $A^\omega$.

$$\frac{}{\mathsf{a} \xrightarrow{a} \mathsf{a}} \ \forall a \in A \qquad \frac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{\mathsf{alt}(x, y) \xrightarrow{a} \mathsf{alt}(y', x')} \ \forall a, b \in A$$

**Figure 2.** The GSOS-rules of our running example

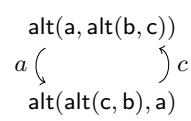


**Figure 3.** A stream system

*Example 2.1.* Let $(\Sigma, A, R)$ be a stream GSOS specification where the signature $\Sigma$ consists of constants $\{\mathsf{a} \mid a \in A\}$ and a binary operation $\mathsf{alt}$. The set $R$ contains the rules in Figure 2. For an instance of (A), the term $\mathsf{alt}(\mathsf{a}, \mathsf{alt}(\mathsf{b}, \mathsf{c})) \in T\emptyset$ defines the stream system depicted in Figure 3. For an instance of (B), the operation $\mathsf{alt} \colon A^\omega \times A^\omega \to A^\omega$ maps streams $a_0 a_1 a_2 \ldots$, $b_0 b_1 b_2 \ldots$ to $a_0 b_1 a_2 b_2 \ldots$.

*Example 2.2.* We now consider the specification $(\Sigma, \mathbb{R}, R)$ which is the fragment of the *stream calculus* [17, 18] consisting of the constants $n \in \mathbb{R}$ and the binary operators *sum* $\oplus$ and *(convolution) product* $\otimes$. The set $R$ is defined in Figure 1 (a). For an example of (A), consider $n \oplus m \xrightarrow{n+m} 0 \oplus 0 \xrightarrow{0} 0 \oplus 0 \xrightarrow{0} \ldots$. For (B), the induced operation $\oplus \colon \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$ is the pointwise sum of streams, i.e., it maps any two streams $n_0 n_1 \ldots$, $m_0 m_1 \ldots$ to $(n_0 + m_0)(n_1 + m_1) \ldots$.

**Definition 2.7.** *We say that a stream GSOS rule $r$ as in* (2) *is* monadic *if $t$ is a term built over variables $\{x'_1, \ldots, x'_n\}$. A stream GSOS specification is* monadic *if all its rules are monadic.*

The specification of Example 2.1 satisfies the monadic stream GSOS format, while the one of Example 2.2 does not since, in the rules for $\otimes$, the variable $y$ occurs in the arriving state of the conclusion.

The notions introduced above for stream GSOS, as well as the analogous ones for standard (labeled transition systems) GSOS [5], can be reformulated in an abstract framework – the so-called *abstract GSOS* [19, 10] – that will be pivotal for the proof of our main result.

In this setting, signatures are represented by *polynomial functors*: a signature $\Sigma$ corresponds to the polynomial functor $\Sigma X = \coprod_{i \in I} X^{n_i}$. For instance, the signature $\Sigma$ in Example 2.1 corresponds to the functor $\Sigma X = A + (X \times X)$, while the signature of Example 2.2 corresponds to the functor $\Sigma X = \mathbb{R} + (X \times X) + (X \times X)$. Models of a signature are seen as algebras for the corresponding functor.

**Definition 2.8.** *Given a functor $F\colon Set \to Set$, an $F$-*algebra *is a pair $(X, d)$, where $X$ is the* carrier set *and $d\colon FX \to X$ is a function. An* algebra homomorphism *from an $F$-algebra $(X, d)$ to an $F$-algebra $(Y, d')$ is a map $h\colon X \to Y$ such that $h \circ d = d' \circ Fh$.*

Particularly interesting are *initial algebras*: an $F$-algebra is called initial if there exists a unique algebra homomorphism from it to every $F$-algebra. For a functor corresponding to a signature $\Sigma$, the initial algebra is $(T\emptyset, \kappa)$ where $\kappa\colon \Sigma T\emptyset \to T\emptyset$ maps, for each $i \in I$, the tuple of closed terms $t_1, \ldots t_{n_i}$ to the closed term $f_i(t_1, \ldots t_{n_i})$. For every set $X$, we can define in a similar way $\kappa_X\colon \Sigma TX \to TX$. The *free monad over $\Sigma$* consists of the endofunctor $T\colon Set \to Set$, mapping every set $X$ to $TX$, together with the natural transformations $\eta\colon Id \Longrightarrow T$ (interpretation of variables as terms) and $\mu\colon TT \Longrightarrow T$ (glueing terms built of terms). Given an algebra $\sigma\colon \Sigma Y \to Y$, for any function $f\colon X \to Y$ there is a unique algebra homomorphism $f^\dagger\colon TX \to Y$ from $(TX, \kappa_X)$ to $(Y, \sigma)$. In particular, the identity function $id\colon X \to X$ induces a unique algebra homomorphism from $TX$ to $X$, which we denote by $\sigma^\sharp\colon TX \to X$; this is the interpretation of terms in $\sigma$.

**Definition 2.9.** *An* abstract GSOS specification *(of $\Sigma$ over $F$) is a natural transformation $\lambda\colon \Sigma(Id \times F) \Longrightarrow FT$. A* monadic abstract GSOS specification *(in short, monadic specification) is a natural transformation $\lambda\colon \Sigma F \Longrightarrow FT$.*

By instantiating the functor $F$ in the above definition to the functor for streams ($FX = A \times X$) one obtains all and only the stream GSOS specifications. Instead, by taking the functor for Mealy machines ($FX = (A \times X)^B$) one obtains the Mealy GSOS format [10]: for the sake of brevity, we do not report the concrete definition here but this notion will be important in Section 5 where, to deal with open terms, we transform stream specifications into Mealy GSOS specifications.

*Example 2.3.* For every set $X$, the rules in Example 2.1 define a function $\lambda_X : A + (A \times X) \times (A \times X) \to (A \times T_\Sigma X)$ as follows: each $a \in A$ is mapped to $(a, \mathsf{a})$ and each pair $(a, x'), (b, y') \in (A \times X) \times (A \times X)$ is mapped to $(a, \mathsf{alt}(y', x'))$ [10].

We focus on monadic distributive laws for most of the paper, and since they are slightly simpler than abstract GSOS specifications, we only recall the relevant concepts for monadic distributive laws. However, we note that the concepts below can be extended to abstract GSOS specifications; see, e.g., [4, 10] for details.

A monadic abstract GSOS specification induces a distributive law $\rho\colon TF \Longrightarrow FT$. This distributive law allows us to extend any $F$-coalgebra $d\colon X \to FX$ to an $F$-coalgebra on terms:

$$TX \xrightarrow{Td} TFX \xrightarrow{\rho_X} FTX$$

This construction generalises and formalises the aforementioned extension of stream systems to terms by means of a stream GSOS specification. In particular, (A) the unique coalgebra on the empty set $!\colon \emptyset \to F\emptyset$ yields an $F$-coalgebra on closed terms $T\emptyset \to FT\emptyset$. If $F$ has a final coalgebra $(Z, \zeta)$, the unique morphism $[\![-]\!]_c\colon T\emptyset \to Z$ defines the *semantics of closed terms*.

$$\begin{tikzcd} T\emptyset \arrow[r, "T!"] \arrow[d, dotted, "{[\![-]\!]_c}"'] \arrow[dr, phantom, "(A)"] & TF\emptyset \arrow[r, "\rho_\emptyset"] & FT\emptyset \arrow[d, dotted, "F[\![-]\!]_c"] \\ Z \arrow[rr, "\zeta"'] & & FZ \end{tikzcd} \qquad \begin{tikzcd} TZ \arrow[r, "T\zeta"] \arrow[d, dotted, "{[\![-]\!]_a}"'] \arrow[dr, phantom, "(B)"] & TFZ \arrow[r, "\rho_Z"] & FTZ \arrow[d, dotted, "F[\![-]\!]_a"] \\ Z \arrow[rr, "\zeta"'] & & FZ \end{tikzcd}$$

Further (B), the final coalgebra $(Z, \zeta)$ yields a coalgebra on $TZ$. By finality, we then obtain a $T$-algebra over the final $F$-coalgebra, which we denote by $[\![-]\!]_a\colon TZ \to Z$ and we call it the *abstract semantics*. We define the *algebra induced by* $\lambda$ as the $\Sigma$-algebra $\sigma\colon \Sigma Z \to Z$ given by

$$\Sigma Z \xrightarrow{\Sigma\eta_Z} \Sigma TZ \xrightarrow{\kappa_Z} TZ \xrightarrow{[\![-]\!]_a} Z\ . \tag{3}$$

## 3 Making arbitrary stream GSOS specifications monadic

The results presented in the next section are restricted to monadic specifications, but one can prove them for arbitrary GSOS specifications by exploiting some auxiliary operators, introduced in [8] with the name of *buffer*. Theorem 6.1 in Section 6 only holds for monadic GSOS specifications. This does not restrict the applicability of our approach: as we show below, arbitrary stream GSOS specifications can be turned into monadic ones.

Let $(\Sigma, A, R)$ be a stream GSOS specification. The extended signature $\tilde{\Sigma}$ is given by $\{\tilde{f} \mid f \in \Sigma\} \cup \{a._ \mid a \in A\}$. The set of rules $\tilde{R}$ is defined as follows:

– For all $a, b \in A$, $\tilde{R}$ contains the following rule

$$\frac{x \xrightarrow{b} x'}{a.x \xrightarrow{a} b.x'} \tag{4}$$

– For each rule $r = \frac{x_1 \xrightarrow{a_1} x'_1 \quad \cdots \quad x_n \xrightarrow{a_n} x'_n}{f(x_1,\ldots,x_n) \xrightarrow{a} t(x_1,\ldots,x_n,x'_1,\ldots,x'_n)} \in R$, the set $\tilde{R}$ contains

$$\tilde{r} = \frac{x_1 \xrightarrow{a_1} x'_1 \quad \cdots \quad x_n \xrightarrow{a_n} x'_n}{\tilde{f}(x_1,\ldots,x_n) \xrightarrow{a} \tilde{t}(a_1.x'_1,\ldots,a'_n.x'_n,x'_1,\ldots,x'_n)} \tag{5}$$

where $\tilde{t}$ is the term obtained from $t$ by replacing each $g \in \Sigma$ by $\tilde{g} \in \tilde{\Sigma}$.

The specification $(\tilde{\Sigma}, A, \tilde{R})$ is now monadic and preserves the original semantics as stated by the following result.

**Theorem 3.1.** *Let $(\Sigma, A, R)$ be a stream GSOS specification and $(\tilde{\Sigma}, A, \tilde{R})$ be the corresponding monadic one. Then, for all $t \in T_\Sigma\emptyset$, $t \sim \tilde{t}$.*

*Example 3.1.* Consider the non-monadic specification in Example 2.2. The corresponding monadic specification consists of the rules in Figure 1 (b) where, to keep the notation light, we used operation symbols $f$ rather than $\tilde{f}$.

## 4 Pointwise Extensions of Monadic GSOS Specifications

The first step to deal with the semantics of open terms induced by a stream GSOS specification is to transform the latter into a Mealy GSOS specification. We follow the approach in [8] which is defined for arbitrary GSOS but, as motivated in Section 3, we restrict our attention to monadic specifications.

Let $(\Sigma, A, R)$ be a monadic stream GSOS specification and $B$ some input alphabet. The corresponding monadic Mealy GSOS specification is a tuple $(\Sigma, A, B, \overline{R})$, where $\overline{R}$ is the least set of Mealy rules which contains, for each stream GSOS rule $r = \frac{x_1 \xrightarrow{a_1} x'_1 \quad \cdots \quad x_n \xrightarrow{a_n} x'_n}{f(x_1,\ldots,x_n) \xrightarrow{a} t(x'_1,\ldots,x'_n)} \in R$ and $b \in B$, the Mealy rule $\overline{r}_b$ defined by

$$\overline{r}_b = \frac{x_1 \xrightarrow{b|a_1} x'_1 \quad \cdots \quad x_n \xrightarrow{b|a_n} x'_n}{f(x_1,\ldots,x_n) \xrightarrow{b|a} t(x'_1,\ldots,x'_n)} \tag{6}$$

An example of this construction is shown in Figure 1 (c).

Recall from Section 2 that any abstract GSOS specification induces a $\Sigma$-algebra on the final $F$-coalgebra. Let $\sigma\colon \Sigma A^\omega \to A^\omega$ be the algebra induced by the stream specification and $\overline{\sigma}\colon \Sigma\Gamma(B^\omega, A^\omega) \to \Gamma(B^\omega, A^\omega)$ the one induced by the corresponding Mealy specification. Theorem 4.1, at the end of this section, informs us that $\overline{\sigma}$ is the *pointwise extension* of $\sigma$.

**Definition 4.1.** *Let $g\colon (A^\omega)^n \to A^\omega$ and $\bar{g}\colon (\Gamma(B^\omega, A^\omega))^n \to \Gamma(B^\omega, A^\omega)$ be two functions. We say that $\bar{g}$ is the* pointwise extension *of $g$ iff for all $\mathfrak{c}_1,\ldots,\mathfrak{c}_n \in \Gamma(B^\omega, A^\omega)$ and $\beta \in B^\omega$, $\bar{g}(\mathfrak{c}_1,\ldots,\mathfrak{c}_n)(\beta) = g(\mathfrak{c}_1(\beta),\ldots,\mathfrak{c}_n(\beta))$. This notion is lifted in the obvious way to $\Sigma$-algebras for an arbitrary signature $\Sigma$.*

*Example 4.1.* Recall the operation $\oplus\colon A^\omega \times A^\omega \to A^\omega$ from Example 2.2 that arises from the specification in Figure 1 (a) (it is easy to see that the same operation also arises from the monadic specification in Figure 1 (b)). Its pointwise extension $\bar{\oplus}\colon \Gamma(B^\omega, \mathbb{R}^\omega) \times \Gamma(B^\omega, \mathbb{R}^\omega) \to \Gamma(B^\omega, \mathbb{R}^\omega)$ is defined for all $\mathfrak{c}_1, \mathfrak{c}_2 \in \Gamma(B^\omega, \mathbb{R}^\omega)$ and $\beta \in B^\omega$ as $(\mathfrak{c}_1 \bar{\oplus} \mathfrak{c}_2)(\beta) = \mathfrak{c}_1(\beta) \oplus \mathfrak{c}_2(\beta)$. Theorem 4.1 tells us that $\bar{\oplus}$ arises from the corresponding Mealy GSOS specification (Figure 1(c)).

In [8], the construction in (6) is generalised from stream specifications to arbitrary abstract GSOS. The key categorical tool is the notion of *costrength* for an endofunctor $F\colon \mathit{Set} \to \mathit{Set}$. Given two sets $B$ and $X$, we first define $\epsilon^b\colon X^B \to X$ as $\epsilon^b(f) = f(b)$ for all $b \in B$. Then, $cs^F_{B,X}\colon F(X^B) \to (FX)^B$ is a natural map in $B$ and $X$, given by $cs^F_{B,X}(t)(b) = (F\epsilon^b)(t)$.

Now, given a monadic specification $\lambda\colon \Sigma F \Longrightarrow FT$, we define $\bar{\lambda}\colon \Sigma(F^B) \Longrightarrow (FT)^B$ as the natural transformation that is defined for all sets $X$ by

$$\Sigma(FX)^B \xrightarrow{cs^\Sigma_{B,FX}} (\Sigma FX)^B \xrightarrow{\lambda^B_X} (FTX)^B \ . \tag{7}$$

Observe that $\bar{\lambda}$ is also a monadic specification, but for the functor $F^B$ rather than the functor $F$. The reader can easily check that for $F$ being the stream functor $FX = A \times X$, the resulting $\bar{\lambda}$ is indeed the Mealy specification corresponding to $\lambda$ as defined in (6).

It is worth to note that the construction of $\bar{\lambda}$ for an arbitrary abstract GSOS $\lambda\colon \Sigma(Id \times F) \Longrightarrow FT$, rather than a monadic one, would not work as in (7). The solution devised in [8] consists of introducing some auxiliary operators as already discussed in Section 3. The following result has been proved in [8] for arbitrary abstract GSOS, with these auxiliary operators. Our formulation is restricted to monadic specifications.

**Theorem 4.1.** *Let $F$ be a functor with a final coalgebra $(Z, \zeta)$, and let $(\bar{Z}, \bar{\zeta})$ be a final $F^B$-coalgebra. Let $\lambda\colon \Sigma F \Longrightarrow FT$ be a monadic distributive law, and $\sigma\colon \Sigma Z \to Z$ the algebra induced by it. The algebra $\bar{\sigma}\colon \Sigma \bar{Z} \to \bar{Z}$ induced by $\bar{\lambda}$ is a pointwise extension of $\sigma$.*

In the theorem above, the notion of pointwise extension should be understood as a generalisation of Definition 4.1 to arbitrary final $F$ and $F^B$-coalgebras. This generalised notion, that has been introduced in [8], will not play a role for our paper where $F$ is fixed to be the stream functor $FX = A \times X$.

## 5 Mealy Machines over Open Terms

We now consider the problem of defining a semantics for the set of open terms $T\mathcal{V}$ for a fixed set of variables $\mathcal{V}$. Our approach is based on the results in the previous sections: we transform a monadic GSOS specification for streams with outputs in $A$ into a Mealy machine with inputs in $A^\mathcal{V}$ and outputs in $A$, i.e., a coalgebra for the functor $FX = (A \times X)^{A^\mathcal{V}}$. The coinductive extension of this

Mealy machine provides the open semantics: for each open term $t \in T\mathcal{V}$ and variable assignment $\psi\colon \mathcal{V} \to A^\omega$, it gives an appropriate output stream in $A^\omega$. This is computed in a stepwise manner: for an input $\varsigma\colon \mathcal{V} \to A$, representing "one step" of a variable assignment $\psi$, we obtain one step of the output stream.

We start by defining a Mealy machine $c\colon \mathcal{V} \to (A \times \mathcal{V})^{A^{\mathcal{V}}}$ on the set of variables $\mathcal{V}$ as on the left below, for all $\mathcal{X} \in \mathcal{V}$ and $\varsigma \in A^{\mathcal{V}}$:

$$c(\mathcal{X})(\varsigma) = (\varsigma(\mathcal{X}), \mathcal{X}) \qquad\qquad \mathcal{X} \circlearrowright \varsigma|\varsigma(\mathcal{X}) \tag{8}$$

Concretely, this machine has variables as states and for each $\varsigma\colon \mathcal{V} \to A$ a self-loop, as depicted on the right. Now, let $\lambda\colon \Sigma(A \times -) \Rightarrow A \times T$ be a monadic stream specification and $\bar{\lambda}\colon \Sigma((A \times -)^{A^{\mathcal{V}}}) \Rightarrow (A \times T(-))^{A^{\mathcal{V}}}$ be the induced Mealy specification, as defined in (7). As mentioned in Section 2, $\bar{\lambda}$ defines a distributive law $\rho\colon T((A \times -)^{A^{\mathcal{V}}}) \Rightarrow (A \times T(-))^{A^{\mathcal{V}}}$, which allows to extend $c$ (see (8)) to a coalgebra $m_\lambda\colon T\mathcal{V} \to (A \times T\mathcal{V})^{A^{\mathcal{V}}}$, given by

$$T\mathcal{V} \xrightarrow{Tc} T(A \times \mathcal{V})^{A^{\mathcal{V}}} \xrightarrow{\rho_{\mathcal{V}}} (A \times T\mathcal{V})^{A^{\mathcal{V}}}. \tag{9}$$

This is the Mealy machine of interest.

*Example 5.1.* Consider the stream specification $\lambda$ of the operation $\mathsf{alt}$, given in Example 2.1. The states of the Mealy machine $m_\lambda$ are the open terms $T\mathcal{V}$. The transitions of terms are defined by the set of rules

$$\frac{}{\mathsf{a} \xrightarrow{\varsigma|a} \mathsf{a}} \qquad \frac{x \xrightarrow{\varsigma|a} x' \quad y \xrightarrow{\varsigma|b} y'}{\mathsf{alt}(x,y) \xrightarrow{\varsigma|a} \mathsf{alt}(y',x')} \qquad \text{for all } \varsigma\colon \mathcal{V} \to A \text{ and } a, b \in A$$

together with the transitions for the variables as in (8). For instance, for each $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \in \mathcal{V}$ and all $\varsigma, \varsigma'\colon \mathcal{V} \to A$, we have the following transitions in $m_\lambda$:

$$\begin{tikzcd} \mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{Y}, \mathcal{Z})) \arrow[loop left, "\varsigma|\varsigma(\mathcal{X})"] \arrow[loop right, "\varsigma'|\varsigma'(\mathcal{Z})"] \\ \mathsf{alt}(\mathsf{alt}(\mathcal{Z}, \mathcal{Y}), \mathcal{X}) \end{tikzcd}$$

*Example 5.2.* For the fragment of the stream calculus introduced in Example 2.2, the Mealy machine over open terms is defined by the rules in Figure 1(d). Below we draw the Mealy machines of some open terms that will be useful later.

$$\begin{tikzcd} \varsigma|\varsigma(\mathcal{X})+\varsigma(\mathcal{Y}) \\ \mathcal{X} \oplus \mathcal{Y} \arrow[loop above] \end{tikzcd} \qquad \begin{tikzcd} \varsigma|\varsigma(\mathcal{Y})+\varsigma(\mathcal{X}) \\ \mathcal{Y} \oplus \mathcal{X} \arrow[loop above] \end{tikzcd} \qquad \begin{tikzcd} \varsigma|(\varsigma(\mathcal{X})+\varsigma(\mathcal{Y}))+\varsigma(\mathcal{Z}) \\ (\mathcal{X} \oplus \mathcal{Y}) \oplus \mathcal{Z} \arrow[loop above] \end{tikzcd} \qquad \begin{tikzcd} \varsigma|\varsigma(\mathcal{X})+(\varsigma(\mathcal{Y})+\varsigma(\mathcal{Z})) \\ \mathcal{X} \oplus (\mathcal{Y} \oplus \mathcal{Z}) \arrow[loop above] \end{tikzcd}$$

We define the open semantics below by the coinductive extension of $m_\lambda$. Let $\tilde{\Gamma} = \Gamma((A^{\mathcal{V}})^\omega, A^\omega)$ be the set of causal functions $\mathfrak{c}\colon (A^{\mathcal{V}})^\omega \to A^\omega$, which is the carrier of the final coalgebra for the functor $FX = (A \times X)^{A^{\mathcal{V}}}$. Notice

that a function $\mathfrak{c}\colon (A^{\mathcal{V}})^\omega \to A^\omega$ can equivalently be presented as a function $\tilde{\mathfrak{c}}\colon (A^\omega)^{\mathcal{V}} \to A^\omega$ (swapping the arguments in the domain). Given such a function $\mathfrak{c}\colon (A^{\mathcal{V}})^\omega \to A^\omega$ and a function $\psi\colon \mathcal{V} \to A^\omega$, in the sequel, we sometimes abuse of notation by writing $\mathfrak{c}(\psi)$ where we formally mean $\tilde{\mathfrak{c}}(\psi)$.

**Definition 5.1.** *Let $\lambda\colon \Sigma(A \times -) \Rightarrow A \times T$ be a monadic stream GSOS specification. The* open semantics *of $\lambda$ is the coinductive extension $[\![-]\!]_o\colon T\mathcal{V} \to \tilde{\Gamma}$ of the Mealy machine $m_\lambda\colon T\mathcal{V} \to (A \times T\mathcal{V})^{A^{\mathcal{V}}}$ defined in* (9).

Behavioural equivalence of open terms can now be checked by means of bisimulations on Mealy machines (Definition 2.4). We define *open bisimilarity*, denoted by $\sim_o$, as the greatest bisimulation on $m_\lambda$. Obviously, for all open terms $t_1, t_2 \in T\mathcal{V}$ it holds that $t_1 \sim_o t_2$ iff $[\![t_1]\!]_o = [\![t_2]\!]_o$. The following result provides another useful characterisation of $[\![-]\!]_o$.

**Lemma 5.1.** *Let $\lambda$ be a monadic stream GSOS specification, with induced algebra $\sigma\colon \Sigma A^\omega \to A^\omega$. Let $\bar{\lambda}$ be the corresponding Mealy specification, with induced algebra $\bar{\sigma}\colon \Sigma\tilde{\Gamma} \to \tilde{\Gamma}$. Then the open semantics $[\![-]\!]_o$ is the unique homomorphism making the diagram below commute:*

$$\begin{tikzcd}
\Sigma T\mathcal{V} \arrow[r, "\Sigma[\![-]\!]_o"] \arrow[d, "\kappa_{\mathcal{V}}"'] & \Sigma\tilde{\Gamma} \arrow[d, "\bar{\sigma}"] \\
T\mathcal{V} \arrow[r, "[\![-]\!]_o"] & \tilde{\Gamma} \\
\mathcal{V} \arrow[u, "\eta_{\mathcal{V}}"] \arrow[ur, "proj"'] &
\end{tikzcd} \tag{10}$$

*where $\eta$ and $\kappa$ are defined by initiality (Section 2), and for each $\mathcal{X} \in \mathcal{V}$ and $\psi\colon \mathcal{V} \to A^\omega$, $proj(\mathcal{X})(\psi) = \psi(\mathcal{X})$.*

Observe that, by virtue of Theorem 4.1, the algebra $\bar{\sigma}$ is the pointwise extension of $\sigma$. This fact will be useful in the next section to relate $\sim_o$ with bisimilarity on the original stream system.

### 5.1 Abstract, Open and Closed Semantics

Recall from Section 2 the abstract semantics $[\![-]\!]_a\colon TA^\omega \to A^\omega$ arising as in (B) from a monadic stream specification $\lambda$. The following proposition is the key to prove Theorem 5.1 relating open bisimilarity and abstract semantics.

**Proposition 5.1.** *Let $[\![-]\!]_a$ and $[\![-]\!]_o$ be the abstract and open semantics respectively of a monadic stream GSOS specification $\lambda$. For any $t \in T\mathcal{V}$, $\psi\colon \mathcal{V} \to A^\omega$:*

$$[\![t]\!]_o(\psi) = [\![(T\psi)(t)]\!]_a\,.$$

As a simple consequence, we obtain the following characterization of $\sim_o$.

**Theorem 5.1.** *For all $t_1, t_2 \in T\mathcal{V}$, $[\![t_1]\!]_o = [\![t_2]\!]_o$ iff for all $\psi\colon \mathcal{V} \to A^\omega$: $[\![T\psi(t_1)]\!]_a = [\![T\psi(t_2)]\!]_a$.*

This is one of the main results of this paper: $T\psi(t_1)$ and $T\psi(t_2)$ are expressions in $TA^\omega$ built from symbols of the signature $\Sigma$ and streams $\alpha_1, \dots \alpha_n \in A^\omega$. By checking $t_1 \sim_o t_2$ one can prove that the two expressions are equivalent for all possible streams $\alpha_1, \dots \alpha_n \in A^\omega$.

*Example 5.3.* By using the Mealy machine $m_\lambda$ in Example 5.1, the relation

$$R = \{(\mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{Y}, \mathcal{Z})), \mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{W}, \mathcal{Z}))), (\mathsf{alt}(\mathsf{alt}(\mathcal{Z}, \mathcal{Y}), \mathcal{X}), \mathsf{alt}(\mathsf{alt}(\mathcal{Z}, \mathcal{W}), \mathcal{X}))\}$$

is easily verified to be a bisimulation (Definition 2.4). In particular this shows that $[\![(\mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{Y}, \mathcal{Z}))]\!]_o = [\![\mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{W}, \mathcal{Z}))]\!]_o$. By Theorem 5.1, we have that $[\![T\psi(\mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{Y}, \mathcal{Z}))]\!]_a = [\![T\psi(\mathsf{alt}(\mathcal{X}, \mathsf{alt}(\mathcal{W}, \mathcal{Z})))]\!]_a$ for all $\psi\colon \mathcal{V} \to A^\omega$, i.e.,

$$\mathsf{alt}(\alpha_1, \mathsf{alt}(\alpha_2, \alpha_3)) \sim \mathsf{alt}(\alpha_1, \mathsf{alt}(\alpha_4, \alpha_3)) \text{ for all } \alpha_1, \alpha_2, \alpha_3, \alpha_4 \in A^\omega.$$

The above law can be understood as an equivalence of program schemes stating that one can always replace the stream $\alpha_2$ by an arbitrary stream $\alpha_4$, without changing the result.

*Example 5.4.* By using the Mealy machines in Example 5.2, it is easy to check that both $\{((\mathcal{X} \oplus \mathcal{Y}) \oplus \mathcal{Z}, \mathcal{X} \oplus (\mathcal{Y} \oplus \mathcal{Z}))\}$ and $\{(\mathcal{X} \oplus \mathcal{Y}, \mathcal{Y} \oplus \mathcal{X})\}$ are bisimulations. This means that $[\![(\mathcal{X} \oplus \mathcal{Y}) \oplus \mathcal{Z}]\!]_o = [\![\mathcal{X} \oplus (\mathcal{Y} \oplus \mathcal{Z})]\!]_o$ and $[\![\mathcal{X} \oplus \mathcal{Y}]\!]_o = [\![\mathcal{Y} \oplus \mathcal{X}]\!]_o$. By Theorem 5.1 we obtain associativity and commutativity of $\oplus$:

$$(\alpha_1 \oplus \alpha_2) \oplus \alpha_3 \sim \alpha_1 \oplus (\alpha_2 \oplus \alpha_3) \text{ and } \alpha_1 \oplus \alpha_2 \sim \alpha_2 \oplus \alpha_1 \text{ for all } \alpha_1, \alpha_2, \alpha_3 \in A^\omega.$$

*Example 5.5.* In a similar way, one can check that $\{((a+b).(\mathcal{X} \oplus \mathcal{Y}), a.\mathcal{X} \oplus b.\mathcal{Y}) \mid a, b \in \mathbb{R}\}$ is a bisimulation. This means that $[\![(a+b).(\mathcal{X} \oplus \mathcal{Y})]\!]_o = [\![a.\mathcal{X} \oplus b.\mathcal{Y}]\!]_o$ for all $a, b \in \mathbb{R}$ and, using again Theorem 5.1, we conclude that $(a+b).(\alpha_1 \oplus \alpha_2) \sim a.\alpha_1 \oplus b.\alpha_2$ for all $\alpha_1, \alpha_2 \in A^\omega$.

Often, equivalence of open terms is defined by relying on the equivalence of closed terms: two open terms are equivalent iff under all possible closed substitutions, the resulting closed terms are equivalent. For $\sim_o$, this property does not follow immediately by Theorem 5.1, where variables range over streams, i.e., elements of the final coalgebra. One could assume that all the behaviours of the final coalgebra are denoted by some term, however this restriction would rule out most of the languages we are aware of: in particular, the stream calculus that can express only the so-called rational streams [18].

The following theorem, that is the second main result of this paper, only requires that the stream GSOS specification is sufficiently expressive to describe arbitrary finite prefixes. We use that any closed substitution $\phi\colon \mathcal{V} \to T\emptyset$ defines $\phi^\dagger\colon T\mathcal{V} \to T\emptyset$ (see Section 2.1).

**Theorem 5.2.** *Suppose $\lambda\colon \Sigma(A \times -) \Rightarrow A \times T_\Sigma$ is a monadic stream GSOS specification which contains, for each $a \in A$, the prefix operator $a.-$ as specified in (4) in Section 3. Further, assume $T\emptyset$ is non-empty.*

*Let $[\![-]\!]_c$ and $[\![-]\!]_o$ be the closed and open semantics respectively of $\lambda$. Then for all $t_1, t_2 \in T\mathcal{V}$: $[\![t_1]\!]_o = [\![t_2]\!]_o$ iff $[\![\phi^\dagger(t_1)]\!]_c = [\![\phi^\dagger(t_2)]\!]_c$ for all $\phi\colon \mathcal{V} \to T\emptyset$.*

$$\frac{x \xrightarrow{a} x'}{f(x) \xrightarrow{a} f(x' \oplus x')} \quad \frac{x \xrightarrow{a} x'}{g(x) \xrightarrow{a} g(x' \oplus x')} \qquad \frac{x \xrightarrow{\varsigma|a} x'}{f(x) \xrightarrow{\varsigma|a} f(x' \oplus x')} \quad \frac{x \xrightarrow{\varsigma|a} x'}{g(x) \xrightarrow{\varsigma|a} g(x' \oplus x')}$$

**Figure 4.** $f$ and $g$, operators over streams **Figure 5.** Pointwise extensions of $f$ and $g$.

*Example 5.6.* The specification in Figure 2 does not include the prefix operator, therefore it does not meet the assumptions of Theorem 5.2. Instead, the monadic GSOS specification in Figure 1(b) contains the prefix. Recall from Example 5.5 that $(a+b).(\mathcal{X} \oplus \mathcal{Y}) \sim_o a.\mathcal{X} \oplus b.\mathcal{Y}$. Using Theorem 5.2, we can conclude that $(a+b).(t_1 \oplus t_2) \sim a.t_1 \oplus b.t_2$ for all $t_1, t_2 \in T\emptyset$.

## 6 Bisimulation up-to substitutions

In the previous section, we have shown that bisimulations on Mealy machines can be used to prove equivalences of open terms specified in the stream GSOS format. In this section we introduce *up-to substitutions*, an enhancement of the bisimulation proof method that allows to deal with smaller, often finite, relations. We also show that up-to substitutions can be effectively combined with other well-known up-to techniques such as up-to bisimilarity and up-to context.

Intuitively, in a bisimulation up-to substitutions $\mathcal{R}$, the states reached by a pair of states do not need to be related by $\mathcal{R}$, but rather by $\theta(\mathcal{R})$, for some substitution $\theta\colon \mathcal{V} \to T\mathcal{V}$. We give a concrete example. Suppose we extend the stream calculus of Example 2.2 with the operators $f$ and $g$ defined by the rules in Figure 4. In Figure 5, we have the pointwise extensions of these new operators. It should be clear that $f(\mathcal{X}) \sim g(\mathcal{X})$. To try to formally prove $f(\mathcal{X}) \sim g(\mathcal{X})$, consider the relation $\mathcal{R} = \{(f(\mathcal{X}), g(\mathcal{X}))\}$. For all $\varsigma\colon \mathcal{V} \to A$, there are transitions $f(\mathcal{X}) \xrightarrow{\varsigma|\varsigma(\mathcal{X})} f(\mathcal{X} \oplus \mathcal{X})$ and $g(\mathcal{X}) \xrightarrow{\varsigma|\varsigma(\mathcal{X})} g(\mathcal{X} \oplus \mathcal{X})$. The outputs of both transitions coincide but the reached states are not in $\mathcal{R}$, hence $\mathcal{R}$ is not a bisimulation. However it is a bisimulation up-to substitutions, since the arriving states are related by $\theta(\mathcal{R})$, for some substitution $\theta$ mapping $\mathcal{X}$ to $\mathcal{X} \oplus \mathcal{X}$. In fact, without this technique, any bisimulation relating $f(\mathcal{X})$ and $g(\mathcal{X})$ should contain infinitely many pairs.

In order to prove the soundness of this technique, as well as the fact that it can be safely combined with other known up-to techniques, we need to recall some notions of the theory of up-to techniques in lattices from [14]. Given a Mealy machine $(X, m)$, we consider the lattice $(\mathcal{P}(X \times X), \subseteq)$ of relations over $X$, ordered by inclusion, and the monotone map $\mathbf{b}\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$ defined for all $\mathcal{R} \subseteq X \times X$ as

$$\mathbf{b}(\mathcal{R}) = \{(s,t) \in X \times X \mid \forall b \in B, o_s(b) = o_t(b) \text{ and } d_s(b)\ \mathcal{R}\ d_t(b)\}. \tag{11}$$

It is easy to see that post fixed points of $\mathbf{b}$, i.e., relations $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathbf{b}(\mathcal{R})$, are exactly bisimulations for Mealy machines (Definition 2.4) and that its greatest fixed point is $\sim$.

For a monotone map $\mathbf{f}\colon \mathcal{P}(X \times X) \to \mathcal{P}(X \times X)$, a *bisimulation up-to* $\mathbf{f}$ is a relation $\mathcal{R}$ such that $\mathcal{R} \subseteq \mathbf{bf}(\mathcal{R})$. We say that $\mathbf{f}$ is *compatible with* $\mathbf{b}$ if $\mathbf{fb}(\mathcal{R}) \subseteq \mathbf{bf}(\mathcal{R})$ for all relations $\mathcal{R}$. Two results in [14] are pivotal for us: first, if $\mathbf{f}$ is compatible and $\mathcal{R} \subseteq \mathbf{bf}(\mathcal{R})$ then $\mathcal{R} \subseteq \sim$; second if $\mathbf{f}_1$ and $\mathbf{f}_2$ are compatible with $\mathbf{b}$ then $\mathbf{f}_1 \circ \mathbf{f}_2$ is compatible with $\mathbf{b}$. The first result informs us that bisimilarity can be proved by means of bisimulations up-to $\mathbf{f}$, whenever $\mathbf{f}$ is compatible. The second result states that compatible up-to techniques can be composed.

We now consider up-to techniques for the Mealy machine over open terms $(T\mathcal{V}, m_\lambda)$ as defined in Section 5. Recall that bisimilarity over this machine is called open bisimilarity, denoted by $\sim_o$. Up-to substitutions is the monotone function $(-)_{\forall\theta}\colon \mathcal{P}(T\mathcal{V} \times T\mathcal{V}) \to \mathcal{P}(T\mathcal{V} \times T\mathcal{V})$ mapping every $\mathcal{R} \subseteq T\mathcal{V} \times T\mathcal{V}$ to

$$(\mathcal{R})_{\forall\theta} = \{(\theta(t_1), \theta(t_2)) \mid \theta\colon \mathcal{V} \to T\mathcal{V} \text{ and } t_1 \ \mathcal{R}\ t_2\}.$$

Similarly, we define up-to context as the monotone function mapping every relation $\mathcal{R} \subseteq T\mathcal{V} \times T\mathcal{V}$ to its contextual closure $\mathcal{C}(\mathcal{R})$ and up-to (open) bisimilarity as the function mapping $\mathcal{R}$ to $\sim_o \mathcal{R} \sim_o = \{(t_1, t_2) \mid \exists t_1', t_2' \text{ s.t. } t_1 \sim_o t_1' \ \mathcal{R}\ t_2' \sim_o t_2\}$.

Compatibility with $\mathbf{b}$ of up-to context and up-to bisimilarity hold immediately by the results in [6]. For the novel technique, up-to substitutions, we have:

**Theorem 6.1.** *The function $(-)_{\forall\theta}$ is compatible with $\mathbf{b}$.*

As a consequence of the above theorem and the results in [14], up-to substitutions can be used in combination with up-to bisimilarity and up-to context (as well as any another compatible up-to technique) to prove open bisimilarity. We will show this in the next, concluding example, for which a last remark is useful: the theory in [14] also ensures that if $\mathbf{f}$ is compatible with $\mathbf{b}$, then $\mathbf{f}(\sim) \subseteq \sim$. By Theorem 6.1, this means that $(\sim_o)_{\forall\theta} \subseteq \sim_o$. The same obviously holds for the contextual closure: $\mathcal{C}(\sim_o) \subseteq \sim_o$.

*Example 6.1.* We prove that the convolution product $\otimes$ distributes over the sum $\oplus$, i.e., $\alpha_1 \otimes (\alpha_2 \oplus \alpha_3) \sim (\alpha_1 \otimes \alpha_2) \oplus (\alpha_1 \otimes \alpha_3)$ for all streams $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}^\omega$. By Theorems 5.1 and 6.1, to prove our statement it is enough to show that $\mathcal{R} = \{(\mathcal{X} \otimes (\mathcal{Y} \oplus \mathcal{Z}), (\mathcal{X} \otimes \mathcal{Y}) \oplus (\mathcal{X} \otimes \mathcal{Z}))\}$ is a bisimulation up-to $\sim_o \mathcal{C}(\sim_o (-)_{\forall\theta} \sim_o) \sim_o$.

By rules in Figure 1(d), for all $\varsigma\colon \mathcal{V} \to \mathbb{R}$, the transitions of the open terms are

- $\mathcal{X} \otimes (\mathcal{Y} \oplus \mathcal{Z}) \xrightarrow{\varsigma | \varsigma(\mathcal{X}) \times (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z}))} (\varsigma(\mathcal{X}) \otimes (\mathcal{Y} \oplus \mathcal{Z})) \oplus (\mathcal{X} \otimes (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})).(\mathcal{Y} \oplus \mathcal{Z}))$
- $(\mathcal{X} \otimes \mathcal{Y}) \oplus (\mathcal{X} \otimes \mathcal{Z}) \xrightarrow{\varsigma | \varsigma(\mathcal{X}) \times \varsigma(\mathcal{Y}) + \varsigma(\mathcal{X}) \times \varsigma(\mathcal{Z})}$
  $((\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y})) \oplus ((\varsigma(\mathcal{X}) \otimes \mathcal{Z}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z}))$

For the outputs, it is evident that $\varsigma(\mathcal{X}) \times (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})) = \varsigma(\mathcal{X}) \times \varsigma(\mathcal{Y}) + \varsigma(\mathcal{X}) \times \varsigma(\mathcal{Z})$. For the arriving states we need a few steps, where for all $\varsigma\colon \mathcal{V} \to \mathbb{R}$ and $\mathcal{X} \in \mathcal{V}$, $\varsigma(\mathcal{X})$ denotes either a real number (used as a prefix) or a constant of the syntax (Example 2.2).

(a) $\mathcal{X} \otimes (\varsigma(\mathcal{Y}).\mathcal{Y} \oplus \varsigma(\mathcal{Z}).\mathcal{Z})\ \mathcal{R}_{\forall\theta}\ (\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z})$.

(b) By Example 5.5 and $\mathcal{C}(\sim_o) \subseteq \sim_o$, we have that:
$\mathcal{X} \otimes (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})).(\mathcal{Y} \oplus \mathcal{Z}) \sim_o \mathcal{X} \otimes (\varsigma(\mathcal{Y}).\mathcal{Y} \oplus \varsigma(\mathcal{Z})).\mathcal{Z})$.

(c) By (b) and (a):
$\mathcal{X} \otimes (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})).(\mathcal{Y} \oplus \mathcal{Z}) \sim_o \mathcal{R}_{\forall\theta} \sim_o (\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z})$.

(d) $\varsigma(\mathcal{X}) \otimes (\mathcal{Y} \oplus \mathcal{Z}) \; \mathcal{R}_{\forall\theta} \; (\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\varsigma(\mathcal{X}) \otimes \mathcal{Z})$.

(e) Using (d) and (c) with context $\mathcal{C} = _ \oplus _$:
$(\varsigma(\mathcal{X}) \otimes (\mathcal{Y} \oplus \mathcal{Z})) \oplus (\mathcal{X} \otimes (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})).(\mathcal{Y} \oplus \mathcal{Z}))$
$\mathcal{C}(\sim_o \mathcal{R}_{\forall\theta} \sim_o) \; ((\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\varsigma(\mathcal{X}) \oplus \mathcal{Z})) \oplus ((\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z}))$.

(f) By Example 5.4 (associativity and commutativity of $\oplus$) and $(\sim_o)_{\forall\rho} \subseteq \sim_o$:
$((\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\varsigma(\mathcal{X}) \oplus \mathcal{Z})) \oplus ((\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z})).\mathcal{Z}))$
$\sim_o ((\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y})) \oplus ((\varsigma(\mathcal{X}) \otimes \mathcal{Z}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z}))$.

(g) By (e) and (f):
$(\varsigma(\mathcal{X}) \otimes (\mathcal{Y} \oplus \mathcal{Z})) \oplus (\mathcal{X} \times (\varsigma(\mathcal{Y}) + \varsigma(\mathcal{Z})).(\mathcal{Y} \oplus \mathcal{Z}))$
$\sim_o \mathcal{C}(\sim_o \mathcal{R}_{\forall\theta} \sim_o) \sim_o ((\varsigma(\mathcal{X}) \otimes \mathcal{Y}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Y}).\mathcal{Y})) \oplus ((\varsigma(\mathcal{X}) \otimes \mathcal{Z}) \oplus (\mathcal{X} \otimes \varsigma(\mathcal{Z}).\mathcal{Z}))$.

## 7 Final remarks

In this paper we have studied the semantics of open terms specified in the stream GSOS format. Our recipe consists in translating the stream specification into a Mealy specification giving semantics to all open terms. Remarkably, this semantics equates two open terms if and only if they are equivalent under all possible interpretations of variables as streams (Theorem 5.1) or under the interpretation of variables as closed terms (Theorem 5.2). Furthermore, semantic equivalence can be checked by means of the bisimulation proof method enhanced with a technique called up-to substitutions (Theorem 6.1).

Our work can be considered as a first step toward a (co)algebraic understanding of the semantics of open terms in the general setting of abstract GSOS [19, 10]. While our approach exploits several peculiarities of the final coalgebra for stream systems, several intermediate results hold in the general setting: for instance, the construction in Section 3 transforming arbitrary stream GSOS specifications into monadic ones, seems to hold for arbitrary abstract GSOS. Another promising clue in this direction comes from the way we specified the semantics of variables in Section 5: it is reminiscent of the technique adopted in [2] for dealing with open terms of process calculi denoting labeled transition systems.

## References


1. L. Aceto, M. Cimini, and A. Ingólfsdóttir. Proving the validity of equations in GSOS languages using rule-matching bisimilarity. *MSCS*, 22(2):291–331, 2012.
2. L. Aceto, W. Fokkink, A. Ingólfsdóttir, and B. Luttik. Finite equational bases in process algebra: Results and open questions. In *Processes, Terms and Cycles*, volume 3838 of *LNCS*, pages 338–367. Springer, 2005.
3. P. Baldan, A. Bracciali, and R. Bruni. A semantic framework for open processes. *Theor. Comput. Sci.*, 389(3):446–483, 2007.

4. F. Bartels. *On generalised coinduction and probabilistic specification formats*. PhD thesis, CWI, Amsterdam, 2004.
5. B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, January 1995.
6. F. Bonchi, D. Petrisan, D. Pous, and J. Rot. A general account of coinduction up-to. *Acta Informatica*, pages 1–64, 2016.
7. R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theor. Comput. Sci.*, 37:245–267, 1985.
8. H. H. Hansen and B. Klin. Pointwise extensions of GSOS-defined operations. *Mathematical Structures in Computer Science*, 21(2):321–361, 2011.
9. H. H. Hansen and J. J. M. M. Rutten. Symbolic synthesis of mealy machines from arithmetic bitstream functions. *Sci. Ann. Comp. Sci.*, 20:97–130, 2010.
10. B. Klin. Bialgebras for structural operational semantics: An introduction. *Theor. Comput. Sci.*, 412(38):5043–5069, 2011.
11. D. Lucanu, E. Goriac, G. Caltais, and G. Rosu. CIRC: A behavioral verification tool based on circular coinduction. In *CALCO 2009. Proc.*, pages 433–442, 2009.
12. M. Mousavi, M. Reniers, and J. Groote. SOS formats and meta-theory: 20 years after. *Theor. Comput. Sci.*, 373(3):238–272, 2007.
13. A. Popescu and E. L. Gunter. Incremental pattern-based coinduction for process algebra and its isabelle formalization. In *FOSSACS, Proc.*, pages 109–127, 2010.
14. D. Pous and D. Sangiorgi. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*. Cambridge, 2012.
15. A. Rensink. Bisimilarity of open terms. *Inf. Comput.*, 156(1-2):345–385, 2000.
16. J. Rutten. Universal coalgebra: a theory of systems. *TCS*, 249(1):3–80, 2000.
17. J. Rutten. Elements of stream calculus (an extensive exercise in coinduction). *ENTCS*, 45:358–423, 2001.
18. J. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *TCS*, 343(3):443–481, 2005.
19. D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *LICS 1997, Proceedings*, pages 280–291, 1997.
20. H. Zantema and J. Endrullis. Proving equality of streams automatically. In *Proceedings of RTA 2011, Novi Sad, Serbia*, pages 393–408, 2011.