Processes as formal power series: a coinductive approach to denotational semantics*

Michele Boreale
Dipartimento di Sistemi e Informatica, Università di Firenze
Viale Morgagni 65, 50134 Firenze, Italia.
boreale@dsi.unifi.it

Fabio Gadducci
Dipartimento di Informatica, Università di Pisa
via Buonarroti 2, 56125 Pisa, Italia.
gadducci@di.unipi.it

May 3, 2006

Abstract

We characterize must testing equivalence on CSP in terms of the unique homomorphism from the Moore automaton of CSP processes to the final Moore automaton of partial formal power series over a certain semiring. The final automaton is then turned into a CSP-algebra: operators and fixpoints are defined, respectively, via *behavioural differential equations* and simulation relations. This structure is then shown to be preserved by the final homomorphism. As a result, we obtain a fully abstract compositional model of CSP phrased in purely set-theoretical terms.

Keywords: process calculi, bisimulation, testing equivalence, coinduction, formal power series.

1 Introduction

The present paper elaborates on two themes. On one hand, we try to reconcile two well-known proposals for process semantics, bisimulation and testing equivalence. On the other hand, we explore a simplified – in particular, purely set-theoretic – treatment of denotational semantics in process calculi. The trait d'union between these two themes is represented by the concept of *formal power series* over a semiring.

*Extended and revised version of [4]. Research partly supported by the the EU within the project SENSORIA. Corresponding author: Michele Boreale, Università di Firenze, Dipartimento di Sistemi e Informatica, Viale Morgagni 65, I-50134, Firenze. Email: boreale@dsi.unifi.it.

Testing equivalence [9, 10] and bisimilarity [17] are two classical proposals for process calculi semantics. They offer different tradeoffs between mathematical tractability and accuracy of process description. Bisimilarity comes equipped with a nice coinductive proof technique. However, it lacks a natural denotational model, and is often blamed of being over-discriminating. Testing equivalence offers perhaps a more faithful picture of reality, with e.g. a proper distinction between termination and divergence, and comes equipped with a fully abstract denotational model. Unfortunately, it lacks tractable proof techniques.

In this paper, we make an attempt at reconciling testing and bisimulation, while keeping the benefits of both. The key to reconciliation is given by the concept of formal power series over a semiring, and the related finality and coinduction principle, as presented in work by Rutten [21, 22].

A formal power series is a function from the set of words over an alphabet A to a semiring $\mathcal{K}$. The set of such functions, denoted $\mathcal{K}\langle A\rangle$, can be given a Moore automaton structure, with inputs in A and outputs in $\mathcal{K}$. This particular automaton is *final*, in the sense that there is a unique homomorphism from every automaton on $\mathcal{K}$ to $\mathcal{K}\langle A\rangle$. It enjoys moreover a *coinduction principle*, by which the unique homomorphism maps two bisimilar states into the same formal power series.

In this paper, we consider a simple process calculus and introduce a semiring for testing, $\mathcal{K}_T$. Next, we turn the process calculus into an automaton $\mathcal{A}ut$ over the semiring $\mathcal{K}_T$ and show that bisimulation over this automaton coincides with (must) testing equivalence. Hence, the unique homomorphism from $\mathcal{A}ut$ to $\mathcal{K}_T\langle A\rangle$ yields a fully abstract semantics for testing equivalence. Finally, we define a set of operators on $\mathcal{K}_T\langle A\rangle$ and show that the final homorphism does preserve these operators, meaning that the resulting model is a truly compositional one. Recursion is modelled via least fixpoints. One nontrivial point of this construction is the treatment of divergence – the possibility for a process of getting engaged in an infinite sequence of internal actions – that is not easily dealt with via bisimulation. In fact, we found it convenient to introduce *partial* formal power series, and to modify the notions of bisimulation and homomorphism accordingly.

Concerning the other theme of the paper, simplifying denotational semantics of processes, the benefits of the above methodology can be summarized as follows:

- Simplicity of the semantic domain. In particular, we dispense with continous (order-theoretic, topological,...) structures and functions. Existence of least fixpoints relies solely on the automaton structure of formal power series.

- Abstract definitions of operators. On the semantic domain, we can specify *behavioural differential equations* (BDE's, [21]) whose unique solutions define the wanted operators. This benefit shows up clearly upon comparison of BDE's with the somewhat intricate definitions often found in a standard, say CPO-based, denotational setting (see e.g. [10]).

- Coinductive reasoning. Proofs by coinduction, which amount to exhibiting

appropriate (bi)simulation relations, are used to show existence of least fixpoints, full abstraction and compositionality of the semantics.

We have chosen CSP for a concrete illustration of our construction, mainly for ease of presentation. The straightforward extensions to CCS and to trace equivalence are also outlined. We hope that the concepts we present here might be relevant for other process calculi. More generally, we have deliberately confined ourselves to a set-theoretic setting, much in the spirit of [21, 22]. However, we expect that a more abstract presentation of our results would not require much effort.

The rest of the paper is organized as follows. In Section 2, we introduce Hoare's CSP [11] and testing equivalence. In Section 3, building on [21], we introduce partial formal power series and the related coinduction and finality principles. In Section 4 we introduce the semiring for testing $\mathcal{K}_T$, and the CSP automaton $\mathcal{A}ut$, and show that the resulting final homomorphism is fully abstract for testing equivalence. In Section 5 we present a few BDE's defining CSP-like operators on the final automaton, we define least fixpoints to model recursion and we prove that the final homomorphism is indeed compositional with respect to these constructions. Section 6 discusses the relationship between our model and the classical *acceptance trees* model for testing equivalence. Section 7 outlines two extensions of the preceding construction. Section 8 discusses directions for further research and related work. The proof of a technical lemma has been confined to Appendix A.

2 CSP and testing semantics

We introduce a process calculus, essentially Hoare's CSP [11], and recall the definition of testing equivalence [9, 10].

2.1 Syntax and operational semantics

We assume a countable set of *visible actions*, denoted by A and ranged over by $a, b, \ldots$; an *invisible action* $\tau \notin A$, with the set $A \cup \{\tau\}$ and $\wp_f(A)$ (the finite subsets of A) ranged over by μ and L, respectively. A set X of *agent variables*, ranged over by $x, y, \ldots$ is also assumed. Open terms are built according to the following syntax

$$P ::= x \mid \mathsf{nil} \mid \mu.P \mid P \oplus P \mid P + P \mid P[a] \mid P \parallel_L P \mid rec_x.P\,.$$

A term is *closed* if each occurrence of a variable x is in the scope of a $rec_x.$ operator. The set of closed terms, or *processes*, is denoted by $\mathcal{P}$ and ranged over by $P, Q, R, \ldots$.

The constant nil represents the terminated process. The action prefix $\mu.P$ can perform an atomic action μ and then evolve to P. The operator $\oplus$ describes non-deterministic *internal* choice: $P \oplus Q$ may evolve via an invisible action either to P or to Q. Summation $+$ denotes non-deterministic *external* choice: $P + Q$ behaves either as P or as Q, the choice being triggered by the environment via synchronization at a visible action. In the parallel composition $P \parallel_L Q$, processes P and

Q must evolve synchronously with respect to each action $a \in L$, while they may evolve independently from each other with respect to actions $\mu \notin L$. The process $P[a]$ behaves like P, except that any execution of the visible action a is turned into the invisible τ, hence hidden from environment. The intended meaning of *recursion* $rec_x.P$ is the behaviour defined by the equation $x = P$.

The operational semantics of $\mathcal{P}$ is described by a *labelled transition system* (LTS) defined in the standard SOS style by the set of rules of Table 1 (where, for the sake of brevity, the symmetric rules for parallel composition and for the two choice operators are not shown).

We recall below a result on the LTS of $\mathcal{P}$ that will be useful later on. Let us denote the composition of two binary relation R_1 and R_2 by R_1R_2. For any $s \in (\mathcal{A} \cup \{\tau\})^*$, $s = \mu_1 \cdots \mu_n$, define the relation $\xrightarrow{s}$ as the composition of relations $\xrightarrow{\mu_1} \cdots \xrightarrow{\mu_n}$. We will often abbreviate $\exists Q : P \xrightarrow{s} Q$ as $P \xrightarrow{s}$.

Lemma 2.1. *The labelled transition system of $\mathcal{P}$ is finitely branching, that is, for each P, the set $\{(\mu, P') \mid P \xrightarrow{\mu} P'\}$ is finite. Furthermore, for each P and $s \in (\mathcal{A} \cup \{\tau\})^*$, also the set $\{\mu \mid P \xrightarrow{s\mu}\}$ is finite.*

$$act : \frac{-}{\mu.P \xrightarrow{\mu} P} \qquad rec : \frac{-}{rec_x.P \xrightarrow{\tau} P[{}^{rec_x.P}/_{x}]} \qquad plus : \frac{-}{P \oplus Q \xrightarrow{\tau} P}$$

$$sum_a : \frac{P \xrightarrow{a} P'}{P + Q \xrightarrow{a} P'} \qquad sum_\tau : \frac{P \xrightarrow{\tau} P'}{P + Q \xrightarrow{\tau} P' + Q}$$

$$hide_a : \frac{P \xrightarrow{a} P'}{P[a] \xrightarrow{\tau} P'[a]} \qquad hide_\mu : \frac{P \xrightarrow{\mu} P'}{P[a] \xrightarrow{\mu} P'[a]} \ \mu \neq a$$

$$par_L : \frac{P \xrightarrow{a} P',\ Q \xrightarrow{a} Q'}{P \parallel_L Q \xrightarrow{a} P' \parallel_L Q'} \ a \in L \qquad par_\mu : \frac{P \xrightarrow{\mu} P'}{P \parallel_L Q \xrightarrow{\mu} P' \parallel_L Q} \ \mu \notin L$$

Table 1: SOS rules for $\mathcal{P}$.

2.2 Testing semantics

The idea underlying *testing* semantics is that two processes should be considered equivalent whenever they pass the same tests proposed by an external observer (see [9, 10]). An "external observer" is any process running in parallel with the given two, while "passing a test" means reaching a state where the observer can fire a "success" action. Within the testing approach, one distinguishes between a

may and a *must* approach, depending on whether one requires that proposed tests may or must be passed by the observed processes. Informally, the may semantics is meant to preserve safety properties of processes, while the must semantics is meant to preserve liveness ones. The must variant, which will be considered in the sequel, is by far more challenging, mainly because it takes into account both the branching structure of processes and the notion of *divergence*. For technical convenience, we shall rely on an alternative, observer-independent characterization of this equivalence, given below. This definition is easily proved to coincide with the original one (see e.g. [8, 10]).

Definition 2.2 (basic relations)**.** *Let P be a process, and let w range over A^*. We define the following relations and sets*

- $\Rightarrow \stackrel{\text{def}}{=} (\stackrel{\tau}{\longrightarrow})^*$*;*
- $\stackrel{a}{\Longrightarrow} \stackrel{\text{def}}{=} \Rightarrow \stackrel{a}{\longrightarrow} \Rightarrow$ *for any* $a \in A$*;*
- $\stackrel{w}{\Longrightarrow} \stackrel{\text{def}}{=} \stackrel{a_1}{\Longrightarrow} \ldots \stackrel{a_n}{\Longrightarrow}$ *for any* $w = a_1 \cdots a_n \in A^*$*;*
- $I(P) \stackrel{\text{def}}{=} \{a \mid \exists Q : P \stackrel{a}{\Longrightarrow} Q\}$*;*
- $\mathcal{A}(P,w) \stackrel{\text{def}}{=} \{I(P') \mid P \stackrel{w}{\Longrightarrow} P'\}$.

Furthermore, we define the following predicates

- $P \Downarrow$ *(read as* P converges*) iff there is no infinite sequence of* τ*-transitions* $P \stackrel{\tau}{\longrightarrow} \stackrel{\tau}{\longrightarrow} \cdots$ *starting from* P *(otherwise* $P \Uparrow$ *holds);*
- $P \Downarrow w$ *(read as* P converges on w*) iff for each prefix* w' *of* w*, whenever* $P \stackrel{w'}{\Longrightarrow} P'$ *then* $P' \Downarrow$ *(otherwise* $P \Uparrow w$ *holds).*

Finally, let $F, G \subseteq_{\text{fin}} \mathcal{P}_f(A)$ be two finite families of finite sets, and let $\bigcup F$ denote $\cup_{X \in F} X$. Then

- $F \asymp G$ *iff* $\bigcup F = \bigcup G$ *and for each* $X \in F$ *there is* $Y \in G$ *such that* $Y \subseteq X$*, and vice-versa.*

The set $\mathcal{A}(P,w)$ is also known as the *acceptance set of P after w*. If one thinks of an "acceptance state" as a set of possible next actions a process is willing to perform, then $\mathcal{A}(P,w)$ represents the set of all possible acceptance states of P after performing w. Then the definition below requires that two equivalent processes exhibit equivalent sets of acceptance states after performing the same (convergent) w's.

We first need a technical lemma, standard from the theory of testing equivalence (see e.g. [10]).

Lemma 2.3. *Let P be a process, and $w \in A^*$. If $P \Downarrow w$, then the sets $\mathcal{A}(P,w)$ and $\{P' \mid P \stackrel{w}{\Longrightarrow} P'\}$ are finite.*

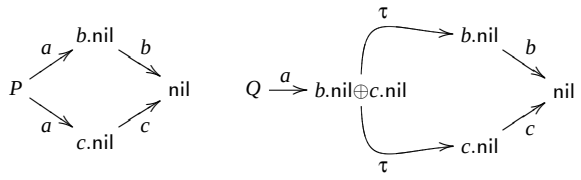

Figure 1: The processes $P \stackrel{\text{def}}{=} a.b.\mathsf{nil} + a.c.\mathsf{nil}$ (left) and $Q \stackrel{\text{def}}{=} a.(b.\mathsf{nil} \oplus c.\mathsf{nil})$ (right).

Definition 2.4 (must testing equivalence [9, 10])**.** *Let P, Q be processes. We say that they are* (must) testing equivalent*, and write $P \simeq Q$, if for each $w \in A^*$*

(a) $P \Downarrow w$ iff $Q \Downarrow w$, and

(b) $P \Downarrow w$ implies $\mathcal{A}(P,w) \asymp \mathcal{A}(Q,w)$.

Example 2.5. *Let us consider the processes $P \stackrel{\text{def}}{=} a.b.\mathsf{nil} + a.c.\mathsf{nil}$ and $Q \stackrel{\text{def}}{=} a.(b.\mathsf{nil} \oplus c.\mathsf{nil})$: the fragments of the transition system associated to the processes are depicted in Figure 1.*

Clearly, $P \simeq Q$. Note also that P and Q are not bisimilar *in the sense of [17].*

We record some useful facts about $\simeq$ in the proposition below (see [9, 10] for a proof in the case of CCS). A *context* $C[\cdot]$ is an open term where a single process variable x may occur free; $C[P]$ denotes the term obtained from $C[\cdot]$ by replacing x with a closed term P.

Proposition 2.6 (properties of $\simeq$)**.** *Let P, Q be processes. If $P \simeq Q$ then $C[P] \simeq C[Q]$ holds for each context $C[\cdot]$. Moreover, $+$ and $\oplus$ are associative and commutative with respect to $\simeq$.*

3 Coinduction on partial formal power series

This section presents a few definitions and results on formal power series, Moore automata and the related coinduction principle. They are directly inspired by [21]. However, Rutten's treatment is extended in order to take into account *partially defined* formal power series.

3.1 Moore automata, homomorphisms and bisimulations

We first recall some notation on partial functions, before introducing *partial* power series. Let X, Y be sets. We denote by $Y_\perp$ the extension of Y with a new element $\perp$, and we model a *partial* function from X to Y as a total function $f : X \to Y_\perp$, writing $f(x) \uparrow$ if $f(x) = \perp$, and $f(x) \downarrow$ otherwise. A function $f : X^1_\perp \times \cdots \times X^n_\perp \to Y_\perp$ is *strict* if f yields $\perp$ whenever one of its argument is $\perp$. For any function $f : X^1 \times \cdots \times X^n \to Y_\perp$, we let $f_\perp : X^1_\perp \times \cdots \times X^n_\perp \to Y_\perp$ denote the *strict extension*

of f, defined as expected. For any relation $R \subseteq X \times Y$, we denote by $R_\perp$ the relation $R \cup \{\langle \perp, \perp \rangle\}$.

Definition 3.1 (partial Moore automaton)**.** *Let A, K be sets. A* (partial) Moore automaton *with inputs in A and outputs in K is a pair $\langle S, (o_S, \delta_S) \rangle$ consisting of a set S of* states*, and of a pair of functions: an* output function $o_S : S \to K_\perp$ *and a* transition function $\delta_S : S \times A \to S_\perp$*, such that if $o_S(s) \uparrow$, then $\delta_S(s,a) \uparrow$ for each $a \in A$.*

Note that, in coalgebraic terms, a partial Moore automaton with inputs in A and outputs in K is just a coalgebra of the set-valued functor $S \mapsto \{\perp\} + K_T \times (A \to S_\perp)$ (see e.g. [15]). In words, each state s the output function yields a (possibly undefined) observation $o_S(s) \in K_\perp$, while for each state s with a defined observation and for each input symbol a, the transition function δ_S yields the (possibly undefined) state $\delta_S(s,a)$ reached from s after the consumption of a. In the rest of the paper, we shall often slightly abuse notation by denoting the set of states of an automaton S by S itself.

For the sake of readability, and unless otherwise indicated, in the rest of the section we let S, T be automata with inputs in set A and outputs in set K.

Definition 3.2 (homomorphism)**.** *A* homomorphism *between S and T is a function $f : S \to T$ preserving output and transition functions, i.e., such that $o_S(s) = o_T(f(s))$ and $f_\perp(\delta_S(s,a)) = \delta_T(f(s),a)$ for each $s \in S$ and $a \in A$.*

Hence, definendness is not only preserved, but also reflected by homomorphims (much in the spirit of closed homomorphims for partial algebras).

Definition 3.3 (bisimulation)**.** *A* bisimulation *is a relation $R \subseteq S \times T$ preserving output and transition functions, i.e., such that if $\langle s,t \rangle \in R$ then $o_S(s) = o_T(t)$ and $\langle \delta_S(s,a), \delta_T(t,a) \rangle \in R_\perp$ for each $a \in A$.*

Let S be an automaton, and let $s, s' \in S$: s and s' are bisimilar *(denoted by $s \sim s'$) if there exists a bisimulation R between S and itself, such that R contains $\langle s, s' \rangle$.*

It is immediate to show that the relation $\sim$ on S is itself a bisimulation. Moreover, since the diagonal relation is a bisimulation, and that bisimulations are closed under union, then $\sim$ is also an equivalence relation on S.

3.2 Formal power series, finality and coinduction

We define partial formal power series as functions with a prefix-closed domain. More formally, we have the following definition:

Definition 3.4 (partial formal power series)**.** *Let A, K be sets. A* partial formal power series *(also partial* FPS*) on K and A is a function $\sigma : A^* \to K_\perp$, such that for all words $w \in A^*$, if $\sigma(w) \uparrow$ then $\sigma(wa) \uparrow$ for each $a \in A$. The set of all partial* FPS*'s on K and A is denoted by $K\langle A \rangle$.*

Let σ be a FPS *in $K\langle A\rangle$, and let $a \in A$. The a-*input derivative of σ, *written σ_a, is the partial* FPS *defined by $\sigma_a(w) = \sigma(aw)$ for all words $w \in A^*$.*

*More generally, the w-*input derivative *of σ is defined by $\sigma_w(w') = \sigma(ww')$, for all words $w' \in A^*$.*

Let $\sigma \in K\langle A\rangle$, and let $w \in A^*$: the *coefficient* of w with respect to σ is the value $\sigma(w)$; $\sigma(\varepsilon)$ is called the *constant* coefficient of the series. Moreover, a series σ is *total* if $\sigma(w) \downarrow$ for each $w \in A^*$. Of special interest is the FPS Ω yielding $\perp$ for all words (modelling the everywhere undefined series).

Depending on the set K, coefficients bear different interpretations. For example, if $A = \{X\}$ and K is the set of real numbers, then a total FPS represents a power series in the usual sense (interpreting the word $X \cdots X$, the element X replicated n times, as X^n). If A is any set, and K is the set of boolean values (i.e., true and false), a total FPS represents a subset of A^*, hence, a language over A. There is no obvious interpretation for non total FPS's in these cases. As we shall see, partiality is well suited to represent divergence in processes.

Now, let us fix sets A and K. We recall below how to turn the set of FPS's into an automaton that is final in the class of automata on A and K, and additionally satisfies a coinduction principle, in the sense that over this automaton bisimilarity coincides with the identity relation. We also take partiality into account.

Definition 3.5 (the automaton of partial FPS's)**.** *Let $\mathcal{M}\langle K\langle A\rangle\rangle$ be the partial Moore automaton with inputs in A and outputs in $\mathcal{K}$ given by the pair $\langle K\langle A\rangle, (o_{\mathcal{M}}, \delta_{\mathcal{M}})\rangle$, where the output and transition functions are defined by $o_{\mathcal{M}}(\sigma) = \sigma(\varepsilon)$ and $\delta_{\mathcal{M}}(\sigma, a) = \sigma_a$ for all $\sigma \in K\langle A\rangle$ and $a \in A$.*

The proof of the following proposition is straightforward and goes along the same lines of the corresponding result of [21].

Proposition 3.6 (finality and coinduction)**.** *The automaton $\mathcal{M}\langle K\langle A\rangle\rangle$ satisfies the* coinduction principle*: for all series $\sigma, \sigma' \in \mathcal{K}\langle A\rangle$, if $\sigma \sim \sigma'$ then $\sigma = \sigma'$.*

Moreover, $\mathcal{M}\langle \mathcal{K}\langle A\rangle\rangle$ is final*: for any automaton S with inputs in A and outputs in $\mathcal{K}$ there exists a unique homomorphism $l : S \to \mathcal{K}\langle A\rangle$, additionally satisfying $s \sim s'$ in S iff $l(s) = l(s')$.*

We will show in Section 5 that this final automaton can be equipped with an algebraic structure that is well-suited for defining a denotational interpretation of processes. The crucial point is that operators on the final automaton can be specified in a uniform and simple fashion as (unique) solutions of *behavioural differential equations* (BDE's). The presentation of this construction is deferred after the next section, where a suitable semiring for interpreting process will be introduced.

4 A semiring for testing equivalence

The main result of this section is that, for an appropriate choice of the set K, the set of CSP processes can be turned into a partial Moore automaton over K and A, in

such a way that testing equivalence on the original transition system corresponds to bisimulation on this automaton. This will yield a fully abstract semantics for testing equivalence, in terms of the unique homomorphism from this automaton into the final automaton of partial FPS's. We will take K to be the carrier of a suitable *semiring* $\mathcal{K}$. The basic intuition is to use the semiring's operation to interpret the two fundamental forms of nondeterminism: sum will be used for interpreting internal non-determinism, while product will be used for interpreting external nondeterminism.

4.1 The semiring $\mathcal{K}_T$

We give the general notion of semiring first.

Definition 4.1 (semirings)**.** *A (commutative, unitary) semiring is a five-tuple* $\mathcal{K} = \langle K, \oplus_{\mathcal{K}}, \otimes_{\mathcal{K}}, 0_{\mathcal{K}}, 1_{\mathcal{K}} \rangle$ *for a set* K*, elements* $0_K, 1_K \in K$*, and binary operators* $\oplus_{\mathcal{K}}, \otimes_{\mathcal{K}} : K \times K \to K$ *making the triples* $\langle K, \oplus_{\mathcal{K}}, 0_{\mathcal{K}} \rangle$ *and* $\langle K, \otimes_{\mathcal{K}}, 1_{\mathcal{K}} \rangle$ *commutative monoids additionally satisfying*

- $x \otimes_{\mathcal{K}} (y \oplus_{\mathcal{K}} z) = (x \otimes_{\mathcal{K}} y) \oplus (x \otimes_{\mathcal{K}} z)$ *for all* $x, y, z \in K$*;*
- $0_{\mathcal{K}} \otimes_{\mathcal{K}} x = 0_{\mathcal{K}}$ *for all* $x \in K$*.*

In the sequel, we shall drop the subscript $\_{\mathcal{K}}$ when denoting semiring operations and constants if no confusion arises about $\mathcal{K}$.

For automata with outputs on the carrier of a semiring $\mathcal{K}$, relevant are the FPS's **0**, yielding 0 for all words, and **1**, with constant 1 and 0 elsewhere. We introduce now a concept used in characterizations of testing equivalence, *saturation* [9, 10]. The rationale behind saturation, as stated in the lemma below, is turning the relation $\asymp$ on families of sets into plain equality.

Definition 4.2 (saturated sets [9, 10])**.** *Let* V *be a set and* F *a finite family of finite subsets of* V*, i.e.* $F \subseteq_{\text{fin}} \wp_f(V)$*. We say that* F *is* saturated *if for all* $X \in F$*, whenever there exists* Y *such that* $X \subseteq Y \subseteq \bigcup F$*, then* $Y \in F$*.*

The saturation *of* F*, written* $\mathcal{S}(F)$*, is the smallest saturated family of subsets of* V *that contains* F*. The set of all saturated families on* V *is denoted by* $\mathcal{F}(V)$*.*

Lemma 4.3. *Let* V *be a set and* $F, G \subseteq_{\text{fin}} \wp_f(V)$*. Then,* $F \asymp G$ *iff* $\mathcal{S}(F) = \mathcal{S}(G)$*.*

We can now turn the saturated families on the set of actions A into a semiring, as follows. The proof of the following proposition is straightforward and omitted (see also the next subsection, where a more general situation is tackled).

Proposition 4.4 (the semiring $\mathcal{K}_T$)**.** *The five-tuple* $\mathcal{K}_T \stackrel{\text{def}}{=} \langle \mathcal{F}(A), \oplus_{\mathcal{K}_T}, \otimes_{\mathcal{K}_T}, \emptyset, \{\emptyset\} \rangle$ *is a semiring, for*

- $F \oplus_{\mathcal{K}_T} G \stackrel{\text{def}}{=} \mathcal{S}(F \cup G)$*;*

- $F \otimes_{\mathcal{K}_T} G \stackrel{\text{def}}{=} \mathcal{S}(\{X \cup Y \mid X \in F, Y \in G\})$.

Example 4.5. *Let us consider*

$$F = \{\{a\}, \{a,b,c\}\} \quad \text{and} \quad G = \{\{a\}, \{a,b\}, \{a,c\}\}.$$

It holds that $F \asymp G$ *and* $\mathcal{S}(F) = \mathcal{S}(G) = \{\{a\}, \{a,b\}, \{a,c\}\{a,b,c\}\}$. *Let* $H = \{\{b,c\}\}$, *which is saturated. Then* $\mathcal{S}(F) \oplus_{\mathcal{K}_T} H = \mathcal{S}(F) \cup H$ *and* $\mathcal{S}(F) \otimes_{\mathcal{K}_T} H = \{\{a,b,c\}\}$.

Another operator on $\mathcal{K}_T$ we shall rely upon is the element-wise set difference, that is, given a set $Y \subseteq A$ and a family $F \subseteq_{\text{fin}} \wp_f(A)$, the family $F \div Y$ is defined as $\{X \setminus Y \mid X \in F\}$. Note that $F \div Y$ is saturated if Y is.

4.2 A detour to tropical semirings

The semiring construction given above is an instance of a general construction, where $\wp_f(A)$ can be replaced by a generic idempotent commutative monoid. In this subsection, we take a brief detour to illustrate this construction. The rest of the paper does not depend on this subsection; however, the presentation may help to convince the reader that alternative instantiations or generalizations of the presented approach are possible.

We will be mainly concerned with *tropical* semirings [19], that is, commutative and unary semirings where the plus operation is idempotent ($x \oplus x = x$).

Proposition 4.6 (power semiring)**.** *Let* $\langle M, \cdot, 1\rangle$ *be a commutative monoid. Then, the five-tuple* $\langle \wp_f(M), \oplus_M, \otimes_M, \emptyset, \{1\}\rangle$ *is a tropical semiring, for*

- $F \oplus_M G = F \cup G$;
- $F \otimes_M G = \{X \cdot Y \mid X \in F, Y \in G\}$.

The powerset construction is quite standard in formal languages theory, and it can be further refined if the $\cdot$ operator is idempotent. In this case, exploiting a general notion of saturation, one can give a semiring construction where the multiplication operation is idempotent. This is essential in process semantics, as the the nondeterminism operators we intend to model are in turn idempotent, and is in fact the "meta-reason" for introducing the notion of saturation. We take the necessary steps below.

Lemma 4.7 (ordered monoid)**.** *Let* $\langle M, \cdot, 1\rangle$ *be an idempotent commutative monoid. Then, the relation* $\leq$, *defined as* $F \leq G$ *iff* $F \cdot G = G$, *is a sup semi-lattice, with* 1 *as bottom and* $\text{lub}\{F, G\} = F \cdot G$.

We say that a finite set $F \subseteq_{\text{fin}} M$ is *finitely generated* if the set $\{Y \in M \mid Y \leq \text{lub}\, F\}$ is finite. Note that the least upper bound always exists for finite sets, since $\leq$ is a sup semi-lattice.

Definition 4.8 (saturation)**.** *Let $\langle M, \cdot, 1\rangle$ be an idempotent commutative monoid. Then, a finitely generated set $F \subseteq_{\text{fin}} M$ is* saturated *if*

- $\text{lub}\, F \in F$*;*
- $\forall X \in F : \forall Y \in M : X \leq Y \leq \text{lub}\, F \Rightarrow Y \in F$.

Let $F \subseteq_{\text{fin}} M$ be a finitely generated set. The saturation *of F, written $\mathcal{S}(F)$, is the smallest saturated set that contains F. The set of all saturated sets of M is denoted by $\mathcal{F}(M)$.*

Note that for a finitely generated set $F \subseteq_{\text{fin}} M$ its saturation $\mathcal{S}(F)$ is also finite and finitely generated.

Proposition 4.9 (saturated semiring)**.** *Let $\langle M, \cdot, 1\rangle$ be an idempotent commutative monoid. Then, the five-tuple $\langle \mathcal{F}(M), \oplus^{\mathcal{S}}_M, \otimes^{\mathcal{S}}_M, \emptyset, \{1\}\rangle$ is a tropical semiring, with idempotent $\otimes^{\mathcal{S}}_M$, for*

- $F \oplus^{\mathcal{S}}_M G = \mathcal{S}(F \oplus_M G)$*;*
- $F \otimes^{\mathcal{S}}_M G = \mathcal{S}(F \otimes_M G)$.

Note that in general $\otimes_M$ is not idempotent, while $\otimes^{\mathcal{S}}_M$ is so, since for any saturated set F the condition $\text{lub}\, F \in F$ implies that F is closed under the composition operator $\cdot$ of the monoid.

In order to prove Proposition 4.9, it is enough the following result, holding for sets $\text{infs}\, F = \{X \in F \mid \nexists Y \in F \setminus \{X\} : Y \leq X\}$.

Lemma 4.10. *Let $\langle M, \cdot, 1\rangle$ be an idempotent commutative monoid, and let $F, G \subseteq_{\text{fin}} M$ be finitely generated sets. Then $\text{lub}\, F = \text{lub}\, G$ and $\text{infs}\, F = \text{infs}\, G$ iff $\mathcal{S}(F) = \mathcal{S}(G)$.*

Explicitly, the condition $\text{infs}\, F = \text{infs}\, G$ boils down to requiring that $\forall X \in F : \exists Y \in G : Y \leq X$, and viceversa. Note that $\mathcal{K}_T$ is the saturated semiring associated to the idempotent commutative monoid $\langle \wp_f(A), \cup, \emptyset\rangle$.

4.3 Consistent formal power series

In order to underline the relevance of the semiring, let us denote by $\mathcal{K}_T\langle A\rangle$ the set of all partial FPS's with coefficients in the carrier of $\mathcal{K}_T$. Our aim is to individuate a subset (in fact, a sub-automaton) of $\mathcal{K}_T\langle A\rangle$ that may act as an interpretation domain for CSP processes. To this purpose, we shall introduce *consistent* series. The intuition is that, when considered as a state of the final automaton, a consistent series encodes a state of a finitely branching LTS. Each FPS σ is decorated with a (saturated) acceptance set, $o(\sigma) = \sigma(\varepsilon)$. In a consistent σ, this output characterizes the next-step behaviour of the represented LTS.

Let us introduce some terminology. For any series σ, we want to regard those derivatives σ_a leading to a state with a 0 output value as being "null". So, let us define the *support of* σ, written $\mathrm{supp}(\sigma)$, as the set of actions $\{a \mid \sigma_a \neq \mathbf{0}\}$.

The definition below requires that in all convergent states reachable from σ, all and only the actions in the support occur in its output (this condition implies that at convergent states the support is finite). Note that the FPS $\mathbf{0}$ is not, by definition, consistent: its behaviour does not represent any LTS (it might be rather viewed as a form of deadlock).

Definition 4.11 (consistent partial FPS's)**.** *A partial* FPS σ *is* consistent *if* $\sigma \neq \mathbf{0}$ *and, for any* $w \in A^*$ *with* $\sigma(w) \downarrow$ *and* $\sigma_w \neq \mathbf{0}$*, it holds that* $\sigma_w(\varepsilon) \neq \emptyset$ *and* $\mathrm{supp}(\sigma_w) = \bigcup(\sigma_w(\varepsilon))$*. The set of consistent* FPS*'s is denoted by* $\mathcal{K}_T^c\langle A\rangle$*.*

Example 4.12. *The series* $\sigma = \mathbf{1}$ *is consistent, as well as the series defined by* $\sigma(\varepsilon) = \{\{a\}\}$, $\sigma_a = \mathbf{1}$ *and* $\sigma_w = \mathbf{0}$ *for any* $w \notin \{\varepsilon, a\}$*. In the latter case, note that if we set* $\sigma_a = \mathbf{0}$ *the series would not be consistent, as we would have* $\mathrm{supp}(\sigma) = \emptyset \neq \bigcup(\sigma(\varepsilon)) = \{a\}$.

Remark 4.13 (sub-automaton of consistent FPS's)**.** *Let* σ *be a consistent* FPS*, and let* $w \in A^*$*. By the conditions imposed, if* $\sigma_w(\varepsilon) = \emptyset$*, then* $\sigma_w = \mathbf{0}$*. Hence, whenever* $\sigma(w) = \sigma_w(\varepsilon) = \emptyset$ *we have* $\sigma_{ww'} = \mathbf{0}$ *for any* $w' \in A^*$*. From this fact it easily follows that the set* $\mathcal{K}_T^c\langle A\rangle \cup \{\mathbf{0}\}$ *is closed under derivatives, in other words it forms a sub-automaton of* $\mathcal{K}_T\langle A\rangle$*.*

An alternative characterisation of this sub-automaton could be obtained along the following lines. First, note the existence of an endofunctor $\mathcal{S}$ *on the category* **Set***, mapping a set* A *to the set of saturated families of finite subsets of* A*. Then, a consistent power series could be seen as a coalgebra of the functor* $\langle o_S, \lambda(\delta_S)\rangle : S \to (\mathcal{S}(A) \times (A \to S_\perp))_\perp$ *by requiring for all* $s \in S$ *with* $o_S(s) \downarrow$ *that* $\mathrm{dom}(\lambda(\delta_S)(s)) = \bigcup o_S(s)$*. Hence, the sub-automaton* $\mathcal{K}_T^c\langle A\rangle \cup \{\mathbf{0}\}$ *could be characterized as the final coalgebra of that functor satisfying the consistency requirement. We do not further elaborate on this remark, since it does not fall in the focus of our paper.*

As an easy consequence of the the above definition and considerations, we record the following fact for future use.

Lemma 4.14. *For* σ *a consistent* FPS *and* $a \in A$*, one and only one of the following three cases holds: (i) there exists* n *such that* $\sigma_{a^k} = \mathbf{0}$ *for each* $k \geq n$*; (ii) there exists* n *such that* $\sigma_{a^k} = \Omega$ *for each* $k \geq n$*; (iii) for each* $n \geq 0$, $\sigma_{a^n} \neq \mathbf{0}, \Omega$*.*

4.4 A Moore automaton for testing

The next step is turning the set of processes $\mathcal{P}$ into an automaton with inputs in A and outputs in (the carrier of) $\mathcal{K}_T$. In view of dealing with the interpretation of open terms, we find it convenient to consider a larger set than $\mathcal{P}$, including constants symbols for all consistent FPS's.

Definition 4.15 (extended CSP)**.** *Let $\mathcal{P}^{\mathrm{e}}$ be the set of all closed terms built from* CSP *operators (Section 2) plus a set of distinct constants $\underline{\sigma}$, one for each $\sigma \in \mathcal{K}_T^{\mathrm{c}}\langle A\rangle$.*

Clearly, $\mathcal{P} \subseteq \mathcal{P}^{\mathrm{e}}$. In order to extend the transition system of $\mathcal{P}$ to $\mathcal{P}^{\mathrm{e}}$, we introduce some additional notation. Let $D = \{P_1, P_2, \ldots, P_n\} \subseteq_{\mathrm{fin}} \mathcal{P}^{\mathrm{e}}$. If $D \neq \emptyset$, we let $\bigoplus_{P\in D} P$ denote the process $P_1 \oplus P_2 \oplus \cdots \oplus P_n \in \mathcal{P}^{\mathrm{e}}$, with the summands P_i arranged in some fixed order. For any D, we let $\Sigma_{P\in D} P$ denote the process $P_1 + P_2 + \cdots + P_n \in \mathcal{P}^{\mathrm{e}}$, with the proviso that if $D = \emptyset$ then this term denotes the process nil. The transition system defined in Table 1 is extended to $\mathcal{P}^{\mathrm{e}}$ via a set of CCS-style recursive definitions. Specifically, we fix a set of equations, one for each constant $\underline{\sigma} \in \mathcal{P}^{\mathrm{e}}$, as follows

$$\underline{\sigma} \stackrel{\mathrm{def}}{=} \begin{cases} \bigoplus_{L\in\sigma(\varepsilon)} \Sigma_{a\in L} a.\underline{\sigma_a} & \text{if } \sigma(\varepsilon)\downarrow \\ rec_x.\tau.x & \text{if } \sigma(\varepsilon)\uparrow \end{cases}$$

and we add the following operational rule

$$const: \frac{\underline{\sigma} \stackrel{\mathrm{def}}{=} P,\ \ P \xrightarrow{\mu} P'}{\underline{\sigma} \xrightarrow{\mu} P'}.$$

Note that the above definition is well-given, since $\sigma(\varepsilon) \neq \emptyset$, and σ_a is consistent for each $a \in \mathrm{supp}(\sigma) = \bigcup(\sigma(\varepsilon))$. The extended transition system is still finitely branching, since consistent FPS's have finite support: in particular Lemma 2.1 carries over to $\mathcal{P}^{\mathrm{e}}$. Also note that testing equivalence on $\mathcal{P}^{\mathrm{e}}$ conservatively extends testing equivalence on $\mathcal{P}$ and it is easily proven to be still a congruence: Proposition 2.6 carries over to $\mathcal{P}^{\mathrm{e}}$.

We build now a Moore automaton out of $\mathcal{P}^{\mathrm{e}}$. We first note that Lemma 2.3 carries over $\mathcal{P}^{\mathrm{e}}$, hence, that for any $P \in \mathcal{P}^{\mathrm{e}}$ and any $w \in A^*$ such that $P \Downarrow w$, the sets $\mathcal{A}(P,w)$ and $\{P' |\ P \stackrel{w}{\Longrightarrow} P'\}$ are finite. In the definition below, we use a new constant $\underline{\mathbf{0}}$ as a "sink" state of the automaton. From now onward, we stipulate that $\bigoplus D$ is $\underline{\mathbf{0}}$ if $D = \emptyset$.

Definition 4.16 ($\mathcal{P}^{\mathrm{e}}$ as a Moore automaton)**.** *The Moore automaton $\mathcal{A}ut$ is the pair $\langle \mathcal{P}^{\mathrm{e}} \cup \{\underline{\mathbf{0}}\}, (o,\delta)\rangle$, where δ and o are defined as follows. Let $a \in A$. First, let $o(\underline{\mathbf{0}}) = \emptyset$ and $\delta(\underline{\mathbf{0}}, a) = \underline{\mathbf{0}}$; then, for $P \neq \underline{\mathbf{0}}$*

$$o(P) \stackrel{\mathrm{def}}{=} \begin{cases} \mathcal{S}(\mathcal{A}(P,\varepsilon)) & \text{if } P \Downarrow \\ \bot & \text{if } P \Uparrow \end{cases}$$

$$\delta(P,a) \stackrel{\mathrm{def}}{=} \begin{cases} \bigoplus\{Q |\ P \stackrel{a}{\Longrightarrow} Q\} & \text{if } P \Downarrow a \\ \bot & \text{if } P \Uparrow a\,. \end{cases}$$

In the sequel, we shall use P_a be a shorthand for $\delta(P,a)$, for any P in $\mathcal{A}ut$ and $a \in A$. More generally, P_w will denote the w-derivative of P in $\mathcal{A}ut$. Our next task is to show that bisimilarity on this automaton precisely captures testing equivalence.

Proposition 4.17 (testing vs. bisimulation)**.** *Let $P,Q \in \mathcal{P}^{\mathrm{e}}$. Then, $P \simeq Q$ if and only if $P \sim Q$ in $\mathcal{A}ut$.*

PROOF: The proof rests upon the following equivalences, whose verification is straightforward. For any a such that $P_a \notin \{\underline{\mathbf{0}}, \bot\}$ and any $w \in A^*$

1. $P \Downarrow aw$ iff $P_a \Downarrow w$
2. $P \Downarrow aw$ implies $\mathcal{A}(P,aw) \asymp \mathcal{A}(P_a,w)$.

We only show that $P \simeq Q$ implies $P \sim Q$ in $\mathcal{A}ut$, as the other direction is easier. Hence, it suffices to prove that $\simeq^{\underline{\mathbf{0}}} \stackrel{\text{def}}{=} \simeq \cup\{\langle \underline{\mathbf{0}}, \underline{\mathbf{0}} \rangle\}$ is a bisimulation on $\mathcal{A}ut$.

Let P and Q be such that $P \simeq Q$. According to the definition of bisimulation, it will suffice to show that (a) $o(P) = o(Q)$ and that (b) $P_a \simeq^{\underline{\mathbf{0}}}_{\bot} Q_a$, for each $a \in A$. The proof is trivial if $P \Uparrow$, so let us assume that $P \Downarrow$. Fact (a) is a direct consequence of the definitions of $o(\cdot)$ and of Lemma 4.3. Concerning (b), it trivially follows from the definition of $\simeq^{\underline{\mathbf{0}}}$ if $P \Uparrow a$ or if $P_a = \underline{\mathbf{0}}$ (in the latter case, it must necessarily be $Q_a = \underline{\mathbf{0}}$, and vice-versa, i.e. $a \notin I(P) = I(Q)$). So let us assume that $P_a, Q_a \neq \underline{\mathbf{0}}$ and that $P \Downarrow a$, hence $P_a, Q_a \neq \bot$. Let us suppose that $P_a \Downarrow w$ for some w. We have to show that (i) $Q_a \Downarrow w$ and that (ii) $\mathcal{A}(P_a,w) \asymp \mathcal{A}(Q_a,w)$. But (i) and (ii) are consequences of $P \simeq Q$ and of facts (1) and (2), respectively, stated above. □

Let l_T be the unique homomorphism $\mathcal{A}ut \to \mathcal{M}\langle \mathcal{K}_T\langle A\rangle\rangle$ induced by finality (Proposition 3.6) and let $[\![\cdot]\!]$ denote its restriction to $\mathcal{P}^{\mathrm{e}}$.

Corollary 4.18 (full abstraction)**.** *The mapping $[\![\cdot]\!] : \mathcal{P}^{\mathrm{e}} \to \mathcal{K}^{\mathrm{c}}_T\langle A\rangle$ is fully abstract for testing equivalence on $\mathcal{P}^{\mathrm{e}}$, i.e., $P \simeq Q$ if and only if $[\![P]\!] = [\![Q]\!]$.*

PROOF: First, note that $\mathcal{A}ut \supseteq \mathcal{P}^{\mathrm{e}}$ and that $l_T(\mathcal{P}^{\mathrm{e}}) \subseteq \mathcal{K}^{\mathrm{c}}_T\langle A\rangle$ (the latter a consequence of the definition of homomorphism). Next, the coinduction principle (Proposition 3.6) and Proposition 4.17 ensure that $[\![\cdot]\!]$ is fully abstract for testing equivalence, hence the thesis. □

5 A compositionality theorem

The next step is to show that the final automaton $\mathcal{M}\langle \mathcal{K}_T\langle A\rangle\rangle$ can be equipped with an algebraic structure which is well suited to interpret processes. We also show how to define least fixpoints to interpret recursion. Next, we prove that the mapping $[\![\cdot]\!]$ defined in the previous section preserves this structure. In other words, $[\![\cdot]\!]$ yields a compositional denotational semantics.

5.1 Algebraic operators

A behavioural differential equation (BDE) specifies a series σ by means of an initial condition (the value of $\sigma(\varepsilon)$) and a condition on its input derivatives (the FPS's σ_a, for each $a \in A$). We can state this as a general definition, as follows.

Definition 5.1 (behavioural differential equations)**.** *A* behavioural differential equation *is a given by an initial condition* $\sigma(\varepsilon) = k$ *(*$k \in \mathcal{K}$*) and by a set of equations on input derivatives of the form* $\sigma_a = f(a, \sigma_a, \sigma)$*, one for each* $a \in A$*.*

For example, the conditions $\sigma(\varepsilon) = 1$ and $\sigma_a = \sigma$ (for each a) define a BDE whose unique solution is the FPS that associates 1 to every word in A^*. In general, the coinduction principle on the final automaton $\mathcal{M}\langle \mathcal{K}_T\langle A\rangle\rangle$ allows us to show the existence and uniqueness of the solution of a BDE. In particular, BDE's can be used as a means to defining operators on $\mathcal{K}_T\langle A\rangle$ in an elegant and uniform fashion.

Table 2 displays the BDE's defining a set of operators on $\mathcal{K}_T^c\langle A\rangle$, clearly inspired by the CSP operators. In fact, by abuse of notation, we shall use symbols drawn from CSP syntax to denote some of these operators. The equations in Table 2 deserve some explanation, but let us introduce the relevant notation first. For a finite set $D = \{\sigma_1, \ldots, \sigma_n\}$ of FPS's, let $\bigoplus D$ denote $\sigma_1 \oplus \cdots \oplus \sigma_n$, with the summands arranged in some fixed order, and stipulating that $\bigoplus D$ is $\mathbf{0}$ if $D = \emptyset$.

First, note the form of the derivative for the $\otimes$ operator, intended for modelling of external nondeterminism $+$, which is reminiscent of the CSP law $a.P + a.Q \simeq a.(P \oplus Q)$. In the equation for the hiding operator $[b](\sigma)$, the totally undefined series Ω models divergence, which may arise either because there is a finite sequence of b actions leading σ to a divergent state, or because σ has an infinite sequence of b actions. As noted in Lemma 4.14, either of these two cases arises precisely when there is no sequence of b's leading to $\mathbf{0}$. Note that all the involved sums ($\bigoplus_{\ldots}$) are finite under the side condition that $\sigma_{b^i} = \mathbf{0}$, given that $\sigma_{b^i w} = \mathbf{0}$ for each $w \in A^*$. Finally, the constant coefficient of the parallel composition $\sigma \parallel_L \rho$ is the product of σ and ρ's constants, but synchronized actions (in L) that are not in the support of both are subtracted away from the result.

Theorem 5.2 (operators on $\mathcal{K}_T^c\langle A\rangle$ via BDE's)**.** *On* $\mathcal{K}_T^c\langle A\rangle$*, there exist unique unary operators* $[b]$ *and* b*. (for each* $b \in A$*) and unique binary operators* $\oplus$*,* $\otimes$*,* $\parallel_L$ *(for each* $L \subseteq_{\text{fin}} A$*), satisfying the* BDE*'s of Table 2, for all* $\sigma, \rho \in \mathcal{K}_T^c\langle A\rangle$ *and* $a \in A$*.*

Moreover, $\oplus$ *and* $\otimes$ *are associative and commutative.*

PROOF: Consider the unique homomorphism $l_T : \mathcal{A}ut \to \mathcal{K}_T\langle A\rangle$. We define the operators in $\mathcal{K}_T^c\langle A\rangle$ we are after as follows: $b.(\sigma) \stackrel{\text{def}}{=} l_T(b.\underline{\sigma})$, $\sigma \oplus \rho \stackrel{\text{def}}{=} l_T(\underline{\sigma} \oplus \underline{\rho})$, $\sigma \otimes \rho \stackrel{\text{def}}{=} l_T(\underline{\sigma} + \underline{\rho})$, $[b](\sigma) \stackrel{\text{def}}{=} l_T(\underline{\sigma}[b])$ and finally $\sigma \parallel_L \rho \stackrel{\text{def}}{=} l_T(\underline{\sigma} \parallel_L \underline{\rho})$ (note that symbols on the left-hand side denote operators on FPS's while symbols on the right-hand side denote syntactic operators of the calculus). It is an easy consequence of the homomorphism properties of l_T that $l_T(\mathcal{A}ut) \subseteq \mathcal{K}_T^c\langle A\rangle \cup \{\mathbf{0}\}$ and that the only state of $\mathcal{A}ut$ mapped to $\mathbf{0}$ is $\underline{\mathbf{0}}$: thus the above operators are well-defined.

Also note that the additional properties of $\oplus$ and $\otimes$ on FPS's are a direct consequence of associativity and commutativity of the operators $\oplus$ and $+$ on $\mathcal{P}^e$ with respect to $\simeq$ (for associativity, also the law $\underline{\sigma} \sim \underline{l_T(\underline{\sigma})}$ is needed, which is in turn a consequence of fact (1) stated below and of the coinduction principle).

To prove that the given operators satisfy the BDE's in Table 2, one first shows that the equations hold in $\mathcal{A}ut$ when replacing each σ with $\underline{\sigma}$, each operator on FPS's

$$(b.(\sigma))_a \quad = \quad \begin{cases} \sigma & \text{if } a = b \\ \mathbf{0} & \text{otherwise} \end{cases} \qquad \text{with} \quad (b.(\sigma))(\varepsilon) \quad = \quad \{\{b\}\}$$

$$\begin{array}{lcllcl} (\sigma \oplus \rho)_a & = & \sigma_a \oplus \rho_a & \text{with} & (\sigma \oplus \rho)(\varepsilon) & = & \sigma(\varepsilon) \oplus_{\mathcal{K}} \rho(\varepsilon) \\ (\sigma \otimes \rho)_a & = & \sigma_a \oplus \rho_a & \text{with} & (\sigma \otimes \rho)(\varepsilon) & = & \sigma(\varepsilon) \otimes_{\mathcal{K}} \rho(\varepsilon) \end{array}$$

$$([b](\sigma))_a \quad = \quad \begin{cases} \mathbf{0} & \text{if } a = b \text{ and } \exists j : \sigma_{b^j} = \mathbf{0} \\ \bigoplus_{i \geq 0, \sigma_{b^i a} \neq \mathbf{0}} [b](\sigma_{b^i a}) & \text{if } a \neq b \text{ and } \exists j : \sigma_{b^j} = \mathbf{0} \\ \Omega & \text{otherwise} \end{cases}$$

$$\text{with} \quad ([b](\sigma))(\varepsilon) = \begin{cases} \bigoplus_{\mathcal{K}, i \geq 0} \sigma(b^i) \div \{b\} & \text{if } \exists j : \sigma_{b^j} = \mathbf{0} \\ \bot & \text{otherwise} \end{cases}$$

$$(\sigma \parallel_L \rho)_a \quad = \quad \begin{cases} \sigma_a \parallel_L \rho_a & \text{if } a \in L \\ (\sigma_a \parallel_L \rho) \oplus (\sigma \parallel_L \rho_a) & \text{otherwise} \end{cases}$$

$$\text{with} \quad (\sigma \parallel_L \rho)(\varepsilon) = (\sigma(\varepsilon) \otimes_{\mathcal{K}} \rho(\varepsilon)) \div (L \setminus (\mathrm{supp}(\sigma) \cap \mathrm{supp}(\rho)))$$

NB: $\oplus_{\mathcal{K}}$, $\otimes_{\mathcal{K}}$ and $\div$ denote the *strict* extensions of the semiring operations of $\mathcal{K}_T$. By convention, in the RHS's of the equations we assume that $\sigma \oplus \mathbf{0} = \mathbf{0} \oplus \sigma = \sigma$; and that $\sigma ||_L \rho = \mathbf{0}$ if either σ or ρ is $\mathbf{0}$.

Table 2: Behavioural differential equations on $\mathcal{K}_T^{\mathrm{c}}\langle A \rangle$.

with the corresponding syntactic operator and equality on series with bisimilarity. Then it will be a consequence of the coinduction principle that the equations hold on FPS's as well. The proof that the equations hold in $\mathcal{A}ut$ is based on the following, easily shown facts about consistent FPS's. Below, we assume $\sigma, \rho \in \mathcal{K}_T^{\mathrm{c}}\langle A \rangle \cup \{\mathbf{0}\}$ and stipulate that for each $w \in A^*$, $\underline{\mathbf{0}} \stackrel{w}{\not\Longrightarrow}$, $\underline{\mathbf{0}} \Downarrow w$ and $\mathcal{A}(\underline{\mathbf{0}}, w) = \emptyset$.

$$\begin{array}{llllllll} (1) & l_T(\underline{\sigma}) & = & \sigma & (3) & \underline{\sigma} \Uparrow & \Leftrightarrow & \sigma = \Omega \\ (2) & \underline{\sigma_w} & \sim & \underline{\sigma}_w & (4) & \underline{\sigma} \Downarrow & \Rightarrow & \sigma(\varepsilon) = \mathcal{A}(\underline{\sigma}, \varepsilon) \end{array}$$

Moreover, assuming $\underline{\sigma} \Downarrow w$, also the facts below hold.

$$(5) \quad \underline{\sigma} \stackrel{w}{\Longrightarrow} \quad \Leftrightarrow \quad (\underline{\sigma}_w \quad \not\sim \quad \underline{\mathbf{0}}) \qquad\qquad (6) \qquad \underline{\sigma}_w \quad \sim \quad \bigoplus \{P \mid \underline{\sigma} \stackrel{w}{\Longrightarrow} P\}$$

$$(7) \quad \mathcal{A}(\underline{\sigma}, w) \quad = \quad \mathcal{A}(\underline{\sigma}_w, \varepsilon)$$

Below, using the facts listed above, we cover in detail the BDE for the hiding operator.

Let us consider the case when there exists $j \geq 0$ such that $\sigma_{b^j} = \mathbf{0}$, as the other case is more easily dealt with. First, note that $\underline{\sigma}[b] \Downarrow$. Indeed, by the SOS rules for hiding, $\underline{\sigma}[b] \Uparrow$ iff either there exists i such that $\underline{\sigma_{b^i}} \sim \underline{\Omega}$, or $\underline{\sigma}$ has an infinite sequence of $\stackrel{b}{\Longrightarrow}$ actions; in the first case, one would have by (2) and (3) above that $\sigma_{b^i} = \Omega$, hence, by the properties of consistent FPS's, it could not exist j with $\sigma_{b^j} = \mathbf{0}$ (see Lemma 4.14); in the second case, one would have from (5) above that either $\underline{\sigma} \Uparrow b^k$ for some k, hence $\underline{\sigma}[b] \Uparrow$, or that $\underline{\sigma_{b^k}} \not\sim \underline{\mathbf{0}}$ for all k, hence by (1) and (2) $\sigma_{b^k} \neq \mathbf{0}$ for all k, contradicting again $\sigma_{b^j} = \mathbf{0}$.

The next step is to show separately that

$$\text{(i)} \quad o(\underline{\sigma}[b]) = \left(\bigoplus_{\mathcal{K}, i \geq 0} \sigma_{b^i}(\varepsilon)\right) \div \{b\}, \qquad \text{and}$$

$$\text{(ii)} \quad (\underline{\sigma}[b])_a \sim \bigoplus_{i \geq 0, \sigma_{b^i a} \neq \mathbf{0}} (\underline{\sigma_{b^i a}})[b] \qquad \text{for } a \neq b$$

(the case $a = b$ is more easily dealt with). Concerning (i), by definition $o(\underline{\sigma}[b]) = \mathcal{S}(L)$, where $L = \mathcal{A}(\underline{\sigma}[b], \varepsilon)$. Now, we have:

$$\begin{array}{rcll} L & = & (\bigcup_{i \geq 0} \{I(P) | \underline{\sigma} \stackrel{b^i}{\Longrightarrow} P\}) \div \{b\} & \text{(by the sos rules for hiding)} \\ & = & (\bigcup_{i \geq 0} \mathcal{A}(\underline{\sigma}, b^i)) \div \{b\} & \text{(by def. of } \mathcal{A}(\cdot,\cdot)) \\ & = & (\bigcup_{i \geq 0} \mathcal{A}(\underline{\sigma_{b^i}}, \varepsilon)) \div \{b\} & \text{(by (7) above; note that } \underline{\sigma_{b^i}} \Downarrow) \\ & = & (\bigcup_{i \geq 0} \sigma_{b^i}(\varepsilon)) \div \{b\} & \text{(by (4) above)} \end{array}$$

from which (i) follows by definition of $\oplus$ on the semiring $\mathcal{K}_T$. Concerning (ii), if $\underline{\sigma}[b] \Uparrow a$, then clearly the two sides of the equation are both bisimilar to $\underline{\Omega}$. Thus let us assume $\underline{\sigma}[b] \Downarrow a$. Then we have

$$\begin{array}{rcll} (\underline{\sigma}[b])_a & \sim & \bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \{P[b] | \underline{\sigma} \stackrel{b^i a}{\Longrightarrow} P\} & \text{(by the sos rules for hiding)} \\ & \sim & (\bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \{P | \underline{\sigma} \stackrel{b^i a}{\Longrightarrow} P\})[b] & \text{(by the law } \bigoplus_j (Q_j[b]) \sim (\bigoplus_j Q_j)[b]) \\ & \sim & (\bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \underline{\sigma_{b^i a}})[b] & \text{(by (6) above; note that } \underline{\sigma_{b^i}} \Downarrow a) \\ & \sim & \bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \underline{\sigma_{b^i a}}[b] & \text{(by the law } \bigoplus_j (Q_j[b]) \sim (\bigoplus_j Q_j)[b]) \end{array}$$

which proves (ii) for this case.

Now, from (i) and the homomorphism property of l_T, the equality for $([b](\sigma))(\varepsilon)$ immediately follows. Let us consider now the equality for $([b](\sigma))_a$. Using the already remarked fact that $\underline{\rho} \sim \overline{l_T(\underline{\rho})}$, for each consistent ρ, and the fact that $\simeq$, hence $\sim$ is a congruence over $\mathcal{P}^{\mathrm{e}}$, we have the following equalities

$$\begin{array}{rcll} ([b](\sigma))_a & = & (l_T(\underline{\sigma}[b]))_a & \text{(by definition of } [b](\cdot)) \\ & = & l_T(\underline{\sigma}[b]_a) & \text{(by homomorphism of } l_T(\cdot)) \\ & = & l_T(\bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \underline{\sigma_{b^i a}}[b]) & \text{(by (ii) above and coinduction)} \\ & = & l_T(\bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} \underline{l_T(\underline{\sigma_{b^i a}}[b])}) & \text{(by } \underline{\rho} \sim \overline{l_T(\underline{\rho})}\text{, congr. and coind.)} \\ & = & \bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} l_T(\underline{\sigma_{b^i a}}[b]) & \text{(by definition of } \oplus \text{ over FPS's)} \\ & = & \bigoplus_{i \geq 0, \underline{\sigma_{b^i a}} \neq \underline{\mathbf{0}}} l_T(\underline{\sigma_{b^i a}}[b]) & \text{(by (2) above and coinduction)} \\ & = & \bigoplus_{i \geq 0, \sigma_{b^i a} \neq \mathbf{0}} [b](\sigma_{b^i a}) & \text{(by definition of } [b](\cdot)). \end{array}$$

Concerning uniqueness, suppose that there are operators on FPS's $b'.(\cdot)$, $\oplus'$, $\otimes'$ and so on satisfying the given BDE's. It is easy to prove, by exhibiting a suitable bisimulation relation, that $b.(\sigma) \sim b'.$, $\sigma \otimes \rho \sim \sigma \otimes' \rho$, $\sigma \parallel_L \rho \sim \sigma \parallel'_L \rho$, and so on, for each σ, ρ. Hence, by the coinduction principle (Proposition 3.6), it follows that $b' . = b.$, $\parallel'_L = \parallel_L$, $\otimes' = \otimes$ and so on. □

5.2 The denotational mapping

In order to model recursion, we have to prove the existence of fixpoints in $\mathcal{K}_T^c\langle A\rangle$. First, consider the partial ordering on $\mathcal{K}_T^c\langle A\rangle$ given by inclusion, i.e., define: $\sigma \subseteq \tau$ iff for each $w \in A^*$, $\sigma(w) \downarrow$ implies $\sigma(w) = \tau(w)$. We use $\subseteq$ as a criterion to select minimal solutions to recursive equations on $\mathcal{K}_T^c\langle A\rangle$. In order to do so, we need a coinductive notion of *simulation*. The latter can be given in general terms as follows.

Definition 5.3 (simulation)**.** *Let S an automaton with inputs in A and outputs in $\mathcal{K}$. A* simulation *on S is a relation $R \subseteq S \times S$ preserving output and transition functions* whenever defined*, i.e., such that if $\langle s,t\rangle \in R$ then $o_S(s) \downarrow$ implies $o_S(s) = o_S(t)$, and $\delta_S(s,a) \downarrow$ implies $\langle \delta_S(s,a), \delta_S(t,a)\rangle \in R$ for each $a \in A$.*

Let $\preceq$ denote the greatest simulation relation over S: it is a preorder on S and the coinduction principle carries over to simulation. More precisely, we have the following principle, which is valid for any semiring $\mathcal{K}$, and whose proof mimics that for bisimulation, as stated in Proposition 3.6.

Proposition 5.4 (coinduction principle for simulation)**.** *Let S be a partial Moore automaton with inputs in A and outputs in $\mathcal{K}$. The unique homomorphism l from S to the final automaton on $\mathcal{K}\langle A\rangle$ (see Proposition 3.6) transforms $\preceq$ into $\subseteq$, that is, $s \preceq t$ in S iff $l(s) \subseteq l(t)$ in $\mathcal{K}\langle A\rangle$.*

Giving semantics to recursive terms is usually accomplished by taking open terms into account via environments. Although this would be technically possible in our case, we prefer to take advantage of the extended syntax of $\mathcal{P}^e$ and dispense with environments. Informally, we want to internalize environments by regarding $[\![P]\!]_E$, where E is the environments mapping $\tilde{x}$ to $\tilde{\sigma}$ component-wise, as $[\![P[\tilde{\sigma}/\tilde{x}]]\!]$.

In the sequel, we write $P[x]$ to denote an open $\mathcal{P}^e$ term where only variable x may occur free, and write $P[Q]$ for $P[Q/x]$, for any Q. The proposition below proves the existence of least fixpoints in $\mathcal{K}_T^c\langle A\rangle$, for mappings that correspond to denotations of open processes $P[x]$. The key technical point is represented by the following lemma, proven in Appendix A. Important ingredients of its proof are a *simulation up to* $\sim$ technique in the vein of [23], and the use of ordinary strong bisimulation [17] for bounding sequences of τ-transitions, when proving convergence of processes.

Lemma 5.5. *Let $P[x]$ be an open term and Q a process in $\mathcal{P}^e$. In $\mathcal{A}ut$, we have that if $Q \sim P[Q]$ then $rec_x.P \preceq Q$.*

Proposition 5.6 (fixpoints)**.** *Let $P[x]$ be an open $\mathcal{P}^{\mathrm{e}}$ term, and let $F : \mathcal{K}_T^{\mathrm{c}}\langle A\rangle \to \mathcal{K}_T^{\mathrm{c}}\langle A\rangle$ be the function defined as $F(\sigma) = [\![P[\underline{\sigma}]]\!]$ for each $\sigma \in \mathcal{K}_T^{\mathrm{c}}\langle A\rangle$. Then F has a least (with respect to $\subseteq$) fixpoint σ_0, and it holds that $\sigma_0 = [\![rec_x.P]\!]$.*

PROOF: Let $\sigma_0 \stackrel{\mathrm{def}}{=} [\![rec_x.P]\!]$. It is easy to see that σ_0 is a fixpoint. Indeed, first note that $\underline{\sigma_0} \sim rec_x.P$ (a consequence of the equality $[\![\underline{\sigma}]\!] = \sigma$, holding for each σ – see proof of Theorem 5.2 – and of the full abstraction of $[\![\cdot]\!]$). Using this fact, and that $P[rec_x.P] \sim rec_x.P$ (immediate from the SOS rule for *rec*), that testing equivalence is a congruence and that the coinduction principle holds for bisimulation, one obtains $F(\sigma_0) = [\![P[\underline{\sigma_0}]]\!] = [\![P[rec_x.P]]\!] = [\![rec_x.P]\!] = \sigma_0$.

The difficult part is showing that σ_0 is the *least* fixpoint. First, given any fixpoint σ of F, since $[\![\underline{\sigma}]\!] = \sigma = F(\sigma) = [\![P[\underline{\sigma}]]\!]$, we have by the coinduction principle for bisimulation that $\underline{\sigma} \sim P[\underline{\sigma}]$. Hence by Lemma 5.5 $rec_x.P \preceq \underline{\sigma}$, and by the coinduction principle for simulation (Proposition 5.4) the thesis follows. □

We denote by $\mathrm{fix}(F)$ the least fixpoint of any $F : \mathcal{K}_T^{\mathrm{c}}\langle A\rangle \to \mathcal{K}_T^{\mathrm{c}}\langle A\rangle$, whenever this fixpoint exists.

Example 5.7. *Let us consider the open term $P[x] = a.(x[a])$. The least fixpoint of the corresponding mapping on $\mathcal{K}_T^{\mathrm{c}}\langle A\rangle$ is the $[\![\cdot]\!]$-image of the process $rec_x.P$, that is $a.(\Omega)$. There are also non-minimal fixpoints for this mapping, like e.g. (the image of) the process $Q = a.b.\mathsf{nil}$. Indeed, it is easy to check that $rec_x.P \preceq Q$.*

Theorem 5.8 (compositionality)**.** *The mapping $[\![\cdot]\!] : \mathcal{P}^{\mathrm{e}} \to \mathcal{K}_T^{\mathrm{c}}\langle A\rangle$ is a morphism with respect to the operators of* CSP *syntax and the operators on $\mathcal{K}_T^{\mathrm{c}}\langle A\rangle$ defined in Table 2. In particular, the equalities in Table 3 hold.*

$$[\![\underline{\sigma}]\!] = \sigma \qquad [\![\mathsf{nil}]\!] = \mathbf{1} \qquad [\![a.P]\!] = a.([\![P]\!]) \qquad [\![\tau.P]\!] = [\![P]\!]$$

$$[\![P \oplus Q]\!] = [\![P]\!] \oplus [\![Q]\!] \qquad [\![P + Q]\!] = [\![P]\!] \otimes [\![Q]\!]$$

$$[\![P[a]]\!] = [a]([\![P]\!]) \qquad [\![P \parallel_L Q]\!] = [\![P]\!] \parallel_L [\![Q]\!] \quad [\![rec_x.P]\!] = \mathrm{fix}(\lambda\sigma.[\![P[\underline{\sigma}/x]]\!])$$

Table 3: The denotational equalities.

PROOF: The first equation has already been remarked (see proof of Theorem 5.2). The equation for **1** is just an instance of the first one. The last equation follows from Proposition 5.6. The remaining equations are easy consequences of the coinduction principle and of the definition of the operators (as seen in proof of Theorem 5.2). Here we show only one case, the parallel operator $\parallel_L$, the others being the same modulo renaming of the involved operator. From the already noted fact $P \sim \underline{l_T(P)}$ for each P, and congruence, it follows $P \parallel_L Q \sim \underline{l_T(P)} \parallel_L \underline{l_T(Q)}$. By coinduction

then: $[\![P \parallel_L Q]\!] = [\![\underline{l_T(P)} \parallel_L \underline{l_T(Q)}]\!] = l_T(P) \parallel_L l_T(Q)$ (by definition of $\parallel_L$ on FPS's, see proof of Theorem 5.2). But the latter is the same as $[\![P]\!] \parallel_L [\![Q]\!]$.

□

Example 5.9. *Let us consider the processes* $P = a.b.\mathsf{nil} + a.c.\mathsf{nil}$ *and* $Q = a.(b.\mathsf{nil} \oplus c.\mathsf{nil})$*, and their denotations in* $\mathcal{K}_T^c\langle A\rangle$*. Dropping a few parentheses, we have* $[\![P]\!] = a.b.\mathbf{1} \otimes a.c.\mathbf{1}$ *and* $[\![Q]\!] = a.(b.\mathbf{1} \oplus c.\mathbf{1})$*. Note now that the following principle can be proven by coinduction: if* $\sigma \in \mathcal{K}_T^c\langle A\rangle$ *then* $\sigma = \langle\sigma(\varepsilon)\rangle \oplus \bigotimes_{a\in \mathrm{supp}(\sigma)} a.\sigma_a$ *(where* $\langle k\rangle$ *denotes the* FPS *with constant* k *and* 0 *elsewhere). By applying this principle, and using the* BDE*'s for* $\oplus$ *and* $\otimes$*, it is immediate to check that* $[\![P]\!] = \langle\{\{a\}\}\rangle \oplus a.(b.\mathbf{1} \oplus c.\mathbf{1}) = [\![Q]\!]$.

6 Discussion: acceptance trees and FPS's

The classical fully abstract denotational model of must testing is described in [10] in terms of *acceptance trees* (AT's).

There is a close analogy between AT's and FPS's. Essentially, a finite AT is a deterministic tree with arcs labelled by actions, and nodes labelled by elements of $\mathcal{K}_T$, with some further consistency and convergence requirements. Indeed, it is easy to see that a finite AT can always be obtained as the unfolding of a suitable consistent FPS. In [10], the set of finite AT's partially ordered by a relation that reflects the *must preorder* on processes is turned into an algebraic CPO by ideal completion. The resulting domain is then used to interpret the operators of a process calculus. In particular, continuity arguments are used to prove existence of least fixpoints when assigning meaning to recursive terms. Within our approach, one need not to deal with continuity arguments explicitly, as the existence of a denotational mapping is guaranteed by the automaton structure of FPS's.

The analogy between the algebraic tree model of [10] and our coalgebraic model is not a coincidence: after Barr [3], it is known that final coalgebras can often be characterized as Cauchy completions of the corresponding initial algebras. We can make the analogy between FPS's and trees more specific following [2]. A Moore automaton can be considered as a coalgebra of the functor $S \mapsto \{\bot\} + \mathcal{K}_T \times (A \to S_\bot)$, which is ω-continuous. The initial algebra for this functor is the set of finite, non-empty trees, with leaves labelled by $\bot$, nodes labelled by elements of $\mathcal{K}_T$ and arcs labelled by elements in A. Accordingly, the corresponding final coalgebra is the set of all infinite trees, equipped with the partial order induced by the operation of replacing subtrees with $\bot$-leaves. This partial order reflects the simulation preorder we have used in our definition of the fixpoint operator; over FPS's, the two coincide.

So far for the analogies. Now, it is worth to notice that the simulation preorder we consider here is strictly finer than the must preorder of [10], despite the fact that in both cases the kernel coincides with the testing equivalence $\simeq$. In the must preorder, one also takes advantage of a preorder on acceptance sets whose kernel is

the relation $\asymp$. For example, in the must preorder one has that $a.\mathsf{nil} \oplus b.\mathsf{nil}$ is smaller than $a.\mathsf{nil}$, which is not true in the simulation preorder. We further elaborate on this point in the concluding section.

7 Extensions

We outline two extensions of the results presented in the previous sections. We shall consider another well-known calculus and an alternative semantics.

7.1 The Calculus of Communicating Systems

The syntax of Milner's CCS (actually, of its tau-less variant, see [10]) is obtained from CSP's by replacing the operators $||_L$ and $[a]$ with parallel composition $|$ and restriction $\backslash a$, respectively. Moreover, an involution $\bar{\cdot} : A \to A$ is assumed on visible actions, i.e., a bijection such that it does not coincide with identity and $a = \bar{\bar{a}}$; this function extends to A^* as expected. The new operational rules are

$$par: \frac{P \xrightarrow{\mu} P'}{P|Q \xrightarrow{\mu} P'|Q} \qquad com: \frac{P \xrightarrow{a} P',\ Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'} \qquad res: \frac{P \xrightarrow{\mu} P',\ \mu \neq a, \bar{a}}{P\backslash a \xrightarrow{\mu} P'\backslash a}$$

(symmetric rule for *par* not shown). The definition of must testing and of the $\mathcal{A}ut$ automaton remain formally unchanged. Concerning BDE's, the equations for $||_L$ and $[a]$ are substituted with the following

$$((\sigma)\backslash b)_a = \begin{cases} \mathbf{0} & \text{if } b \in \{a, \bar{a}\} \text{ and } \sigma \neq \Omega \\ (\sigma_a)\backslash b & \text{if } b \notin \{a, \bar{a}\} \text{ and } \sigma \neq \Omega \\ \Omega & \text{otherwise} \end{cases}$$

$$\text{with} \quad ((\sigma)\backslash b)(\varepsilon) = \sigma(\varepsilon) \div \{b\}$$

$$(\sigma|\rho)_a = \begin{cases} \Omega & \text{if } \forall k \geq 0\, \exists w : |w| = k \text{ and } \sigma_w, \rho_{\bar{w}} \neq \mathbf{0} \\ \bigoplus_{w:\sigma_w,\rho_{\bar{w}} \neq \mathbf{0}} (\sigma_{wa}|\rho_{\bar{w}}) \oplus (\sigma_w|\rho_{\bar{w}a}) & \text{otherwise} \end{cases}$$

$$\text{with} \quad (\sigma|\rho)(\varepsilon) = \sigma(\varepsilon) \otimes_{\mathcal{K}} \rho(\varepsilon)\ .$$

It is a matter of a routine check to verify that our results on full abstraction for CSP carry over to this calculus.

7.2 Trace semantics

Two CSP processes P and Q are *convergent-trace equivalent*, written $P \simeq_{\mathrm{ctr}} Q$, if for each $w \in A^*$

- $P \Downarrow w$ iff $Q \Downarrow w$, and

- $P \stackrel{w}{\Longrightarrow}$ iff $Q \stackrel{w}{\Longrightarrow}$.

This semantics corresponds to language equivalence on convergent traces. We obtain a fully abstract model for this semantics by taking the semiring $\mathcal{K}_{\mathrm{ctr}} \stackrel{\mathrm{def}}{=} \{0,1\}$, i.e., the boolean semiring.

The definition of Moore automaton for convergent trace equivalence is given by changing the clause for $o(\cdot)$ in Definition 4.16 as follows: let $o(\underline{\mathbf{0}}) = 0$, and for $P \neq \underline{\mathbf{0}}$

$$o(P) \stackrel{\mathrm{def}}{=} \begin{cases} 1 & \text{if } P \Downarrow \\ \bot & \text{if } P \Uparrow . \end{cases}$$

Concerning the BDE's for CSP, we need only to change the initial conditions of the equations for $b.$ and $||_L$ by setting $(b.(\sigma))(\varepsilon) = 1$ and $(\sigma||_L\rho)(\varepsilon) = \sigma(\varepsilon) \otimes_{\mathcal{K}} \rho(\varepsilon)$; the other equations listed in Table 2 remain unchanged. We note that results on coalgebraic characterization of trace semantics are well-known (see the concluding section).

8 Conclusions and related work

The paper proposes a coinductive denotational semantics for testing equivalence, building on Rutten's work on the coalgebraic presentation of formal power series. Although results in this vein are known for trace semantics (see below), we are not aware of previous work concerning branching-time semantics, like must testing. More generally, we are not aware of other coalgebraic presentations giving a full account of a nontrivial process calculus, including those aspects related to invisible actions and divergence.

We believe that our characterization of the testing model via FPS's suggests a methodology – whose core lies in the definition of operators via BDE's and in the choice of an appropriate semiring – for different equivalences and/or process calculi. Some extensions have been outlined in the paper. It would be interesting to see how smoothly the present approach carries over to name passing-calculi, like the π-calculus [18]: we plan to make this the subject of a further study. Dually, it would be interesting to see if any sensible semantics or language extensions are suggested by a domain of FPS's itself, for appropriate choices of the semiring. As an example, with our semiring $\mathcal{K}_T$, the presence of the series $\mathbf{0}$ suggests inclusion of a *deadlock* operator in the language.

The relationship between the algebraic AT model of [10] and our model also deserves further consideration. In particular, one wonders precisely how that model can be cast into the present coalgebraic setting. As hinted in Section 6, the AT model can be obtained by ideal completion of a partial order that reflects the must preorder on processes. With reference to Definition 5.3, the must preorder can be phrased on Moore automata over $\mathcal{K}_T$ by changing the requirement on outputs into "$o_S(s) \downarrow$ implies $o_S(t) \subseteq o_S(s)$". It is a matter of further consideration if the correspondence between the order imposed on trees (hence, on FPS's) by simulation and

the order imposed by the algebra/coalgebra duality (in the sense of Adámek [2]) can be extended. We also note that the must preorder induces an order on the functor of the Moore automaton, as defined in [12], in principle making it amenable to the analysis proposed there on the algebraic CPO structure of final coalgebras.

As a further line of research, one would like to consider if the coalgebraic logic machinery [15] gives rise to interesting modal logics for processes when instantiated to the present setting.

Concerning related work, most related to ours appears to be a paper by Cleaveland and Hennessy [5]. They present a bisimulation-like characterization of testing equivalence, but do not work a denotational model out of that. The work of Wolter [25] shares some similarities with our proposal; the considered model is the class of partial nondeterministic automata corresponding to the functor $S \mapsto \mathcal{P}(\mathcal{P}(A)) \times (A \to \wp_f(S)_\perp)$. Differently from our presentation using BDE's, the denotational mapping is obtained by resorting to explicit constructions on automata (with some syntactic limitations). The proposed model fails to achieve full abstraction, due to certain features connected to both nondeterminism (lack of saturation) and internal actions (hiding).

Concerning coalgebraic characterizations of other process semantics, we are aware of a few works on trace semantics, a thread initiated by Power and Turi in [20], and more recently considered also in [13, 16]. Our work is similar in spirit to theirs. However, differently from these contributions, we exploit the concrete, set-theoretic setting of FPS's, partly motivated by our dealing with must testing, which is more challenging than trace equivalence. The first application of the finality principle to concurrency can be found in Aczel's book on non well-founded sets [1]. In fact, this work can be considered as the root of most of the present day interest in coalgebraic methods in semantics.

More generally, the search of coinductive characterization for those equivalences belonging to the so-called van Glabbeek spectrum, as started in [7], appears to be a promising area of research.

Loosely related to ours, a strong thread of research has focused on the extension and generalization of finality results for so-called *bialgebras*, broadly searching whenever the format of the inference rules defining the operational semantics of a calculus ensures that the corresponding final coalgebra semantics preserves the operators of the calculus. The reader is referred to e.g. [6, 24] and, as far as deterministic automata and the semiring monad are concerned, to [14].

References

[1] P. Aczel. *Non-well-founded sets*. Center for the Study of Language and Information, Stanford University. CSLI Lecture Notes, Volume 14, 1988.

[2] J. Adámek. Final coalgebras are ideal completions of initial algebras. *Journal of Logic and Computation*, 12:217–242, 2002.

[3] M. Barr. Terminal coalgebras in well-founded set theory. *Theor. Comp. Sci.*, 114:299–315, 1993.

[4] M. Boreale and F. Gadducci. Denotational testing semantics in coinductive form. In B. Rovan and P. Vojtás, editors, *Mathematical Foundations of Computer Science*, volume 2747 of *Lect. Notes in Comp. Sci.*, pages 279–289. Springer, 2003.

[5] R. Cleaveland and M. Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 5:1–20, 1993.

[6] A. Corradini, R. Heckel, and U. Montanari. Compositional SOS and beyond: A coalgebraic view of open systems. *Theor. Comp. Sci.*, 280:163–192, 2002.

[7] D. de Frutos Escrig and C.G. Rodriguez. Bisimulation up-to for the linear time branching time spectrum. In M. Abadi and L. de Alfaro, editors, *Concurrency Theory*, volume 3653 of *Lect. Notes in Comp. Sci.*, pages 278–292. Springer, 2005.

[8] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[9] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theor. Comp. Sci.*, 34:83–133, 1984.

[10] M. Hennessy. *Algebraic Theory of Processes*. MIT Press, 1988.

[11] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[12] J. Hughes and B. Jacobs. Simulations in coalgebra. *Theor. Comp. Sci.*, 327:71–108, 2004.

[13] B. Jacobs. Trace semantics for coalgebras. In J. Adámek and S. Milius, editors, *Coalgebraic Methods in Computer Science*, volume 106 of *Electr. Notes in Theor. Comp. Sci.*, pages 167–184. Elsevier, 2004.

[14] B. Jacobs. A bialgebraic review of regular expressions, deterministic automata and languages. Technical Report NIII-R050003, Nijmegen Institute for Computing and Information Sciences, 2005.

[15] B. Jacobs. *Introduction to Coalgebra. Towards Mathematics of States and Observations.* In preparation. Available from `www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf`.

[16] B. Klin. A coalgebraic approach to process equivalence and a coinduction principle for traces. In J. Adámek and S. Milius, editors, *Coalgebraic Methods in Computer Science*, volume 106 of *Electr. Notes in Theor. Comp. Sci.*, pages 201–218. Elsevier, 2004.

[17] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[18] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. Part I and II. *Information and Computation*, 100:1–77, 1992.

[19] J.-E. Pin. Tropical semirings. In J. Gunawardena, editor, *Idempotency*, pages 50–69. Cambridge University Press, 1998.

[20] J. Power and D. Turi. A coalgebraic foundation for linear time semantics. In M. Hofmann, G. Rosolini, and D. Pavlovic, editors, *Category Theory and Computer Science*, volume 29 of *Electr. Notes in Theor. Comp. Sci.* Elsevier Science, 1999.

[21] J.J.M.M. Rutten. Automata, power series and coinduction: taking derivatives seriously (extended abstract). In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Automata, Languages and Programming*, volume 1664 of *Lect. Notes in Comp. Sci.*, pages 645–654. Springer, 1999.

[22] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comp. Sci.*, 308:1–53, 2003.

[23] D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8:447–480, 1998.

[24] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Logic in Computer Science*, pages 280–291. IEEE Computer Society Press, 1997.

[25] U. Wolter. CSP, partial automata and coalgebras. *Theor. Comp. Sci.*, 280:3–34, 2002.

A Proof of Lemma 5.5

The main result of the section is Proposition A.6, from which the wanted lemma follows as a corollary. For its proof, we need a few additional definitions and technical results. In what follows, unless otherwise stated, we consider $\mathcal{P}^{\mathrm{e}}$ processes and contexts. We also consider a new label $\alpha \notin A \cup \{\tau\}$, which will serve as a fresh visible action, for use in the definition below; we abbreviate $\alpha.\mathsf{nil}$ as α. We recall the (somehow standard) definitions of guarded context and convergence in k steps.

Definition A.1. *Let $C[\cdot]$ be a context, P be a process and $k \geq 0$. We say that:*

- *$C[\cdot]$ is* k-guarded *if whenever $C[\alpha] \xrightarrow{s\alpha}$, with $s \in (A \cup \{\tau\})^*$, then $|s| \geq k$;*
- *P* converges within k steps, *written $P \Downarrow_k$, if whenever $P \xrightarrow{\tau^i}$ then $i \leq k$;*
- *P* converges along a within k steps *($a \in A$), written $P \Downarrow_k a$, if whenever $P \xrightarrow{\tau^i a \tau^j}$ then $i + j + 1 \leq k$.*

Note that $P \Downarrow$ if and only if $P \Downarrow_k$ for some k, by virtue of the finite-branching-ness of the LTS (Lemma 2.1) and of König's Lemma. A similar remark applies to the other pair of predicates, $P \Downarrow a$ and $P \Downarrow_k a$. The next two lemmas establish some basic properties of guarded contexts. In particular, Lemma A.3 asserts roughly that, when $C[\cdot]$ is "sufficiently guarded", then whatever P is plugged into $C[\cdot]$, P plays no role in the next-step behaviour of $C[P]$.

Lemma A.2. *Let $C[\cdot]$ be k-guarded and suppose $C[P] \xrightarrow{s} R$, with $s \in (A \cup \{\tau\})^*$ and $|s| \leq k$. Then there is a context $C'[\cdot]$ such that $R = C'[P]$ and, for each Q, $C[Q] \xrightarrow{s} C'[Q]$.*

PROOF: An easy induction on $|s|$ proves the stronger statement additionally requiring that $C'[\cdot]$ be $(k - |s|)$-guarded. The base case $s = \mu$ is in turn a transition induction on $C[P] \xrightarrow{\mu} R$. □

Lemma A.3. *Let $C[\cdot]$ be $k+1$-guarded. Then*

1. *there is $F \in \mathcal{K}_T$ such that for each P with $C[P] \Downarrow_k$, $o(C[P]) = F$;*
2. *let $a \in A$; there is $C'[\cdot]$ such that for each P with $C[P] \Downarrow_k a$, $(C[P])_a \sim C'[P]$ (possibly, $C'[\cdot] = \underline{\mathbf{0}}$).*

PROOF: Both assertions are easy consequences of Lemma A.2. As an example, we check 2. Consider the set of contexts

$$\mathfrak{C} \stackrel{\text{def}}{=} \{C_i[\cdot] \mid \exists s \in (A \cup \{\tau\})^*, |s| \leq k \text{ such that } \forall Q : C[Q] \xrightarrow{s} C_i[Q]\}.$$

The set $\mathfrak{C}$ is finite, by the finite-branching-ness of the LTS. Take any P such that $C[P] \Downarrow_k a$ and suppose $C[P] \stackrel{a}{\Longrightarrow} P'$, i.e. $C[P] \xrightarrow{s} P'$ for some $s = \tau^l a \tau^m$. By $C[P] \Downarrow_k$

a, it must be $|s| \leq k$. Hence, by Lemma A.2, $P' = C''[P]$, for some $C''[\cdot]$; moreover $C''[\cdot]$ must be in $\mathfrak{C}$, again by Lemma A.2 and by definition of $\mathfrak{C}$. On the other hand, for any $C_i[\cdot] \in \mathfrak{C}$ it holds $C[P] \stackrel{a}{\Longrightarrow} C_i[P]$, again by definition. So we have shown that $\{P' | C[P] \stackrel{a}{\Longrightarrow} P'\} = \{C_i[P] \,|\, C_i[\cdot] \in \mathfrak{C}\}$. Then, take $C'[\cdot] \stackrel{\text{def}}{=} \bigoplus_{C_i[\cdot]\in\mathfrak{C}} C_i[\cdot]$. □

The next definition introduces another ingredient for the proof, a useful up-to technique. The only difference from the definition of simulation is that the condition on the derivatives "$\langle \delta_S(s,a), \delta_S(t,a)\rangle \in R$" is replaced here by the weaker "$\langle \delta_S(s,a), \delta_S(t,a)\rangle \in \sim R \sim$" (recall that we denote composition of binary relations by juxtaposition).

Definition A.4 (simulation up to bisimulation)**.** *Let S an automaton with inputs in A and outputs in $\mathcal{K}$. A* simulation up to bisimulation *on S is a relation $R \subseteq S \times S$ such that if $\langle s,t\rangle \in R$ then (a) $o_S(s) \downarrow$ implies $o_S(s) = o_S(t)$, and (b) $\delta_S(s,a) \downarrow$ implies $\langle \delta_S(s,a), \delta_S(t,a)\rangle \in \sim R \sim$, for each $a \in A$.*

The following lemma establishes correctness of the above up-to technique.

Lemma A.5. *If R is a simulation up to bisimulation on S then $R \subseteq \preceq$.*

PROOF: Show that the relation $\overline{R} \stackrel{\text{def}}{=} \sim R \sim$ is a simulation on S, which is immediate by transitivity of $\sim$. Since $R \subseteq \overline{R}$ the thesis follows. □

We need two more ingredients for the proof, that is unfoldings of terms and strong bisimulation. Given a context $P[\cdot]$ and $k \geq 0$, the k^{th} *unfolding of* $P[\cdot]$ is the context defined by induction on k as follows: $P^{(0)}[\cdot] \stackrel{\text{def}}{=} [\cdot]$ (the empty context), $P^{(k+1)}[\cdot] \stackrel{\text{def}}{=} \tau.P[P^{(k)}[\cdot]]$. In other words, $P^{(k)}[\cdot]$ is the k-guarded context $\tau.P[\tau.P[\cdots\tau.P[\cdot]\cdots]]$, with k nested τ's.

Let $\sim_{\text{sb}}$ denote ordinary *strong bisimulation* [17] over processes. It is immediate to check, using the fact that $rec_x.P \sim_{\text{sb}} \tau.P[rec_x.P]$ and the congruence properties of strong bisimulation, that for each k, $rec_x.P \sim_{\text{sb}} P^{(k)}[rec_x.P]$. Note that $\sim_{\text{sb}}$ is (strictly) finer than $\simeq$ (hence than $\sim$ and $\preceq$). Finally note that, by definition, $P \stackrel{s}{\longrightarrow}$ and $P \sim_{\text{sb}} Q$ imply $Q \stackrel{s}{\longrightarrow}$, for any $s \in (A \cup \{\tau\})^*$; thus, in particular, $\sim_{\text{sb}}$ preserves all the convergence predicates considered here.

Proposition A.6. *Let $C[\cdot]$ and $P[x]$ be contexts, and Q a process. In $\mathcal{A}ut$, we have that if $Q \sim P[Q]$ then $C[rec_x.P] \preceq C[Q]$.*

PROOF: Fix a generic open term $P[x]$ and fix a generic Q such that $Q \sim P[Q]$. We show that the relation

$$R \stackrel{\text{def}}{=} \{\langle C[rec_x.P], C[Q]\rangle \,|\, C[\cdot] \text{ is a context } \}$$

(where we allow $C[\cdot]$ to possibly be $\underline{\mathbf{0}}$) is a simulation up to bisimulation, thus proving the thesis. Thus, take a generic $C[\cdot]$ (different from the trivial $\underline{\mathbf{0}}$), and suppose $o(C[rec_x.P]) \downarrow$, i.e. $C[rec_x.P] \Downarrow$.

We first check requirement (a) of simulation, that is, $o(C[rec_x.P]) = o(C[Q])$. To see this, we first prove that $C[Q] \Downarrow$ as well. By contradiction, assume $C[Q] \Uparrow$ and take h such that $C[rec_x.P] \Downarrow_h$ (as already noted, this h must exist). Since $C[Q] \sim C[P^{(h+1)}[Q]]$ (by repeating the unfolding $Q \sim P[Q] \sim \tau.P[Q]$ and by congruence), $C[P^{(h+1)}[Q]] \Uparrow$ as well, hence there is a sequence of τ-transitions of length $h+1$, say $C[P^{(h+1)}[Q]] \xrightarrow{\tau^{h+1}}$. Since $C[P^{(h+1)}[\cdot]]$ is $h+1$-guarded, by Lemma A.2 we have $C[P^{(h+1)}[rec_x.P]] \xrightarrow{\tau^{h+1}}$ as well; but $C[P^{(h+1)}[rec_x.P]] \sim_{\mathrm{sb}} C[rec_x.P]$ (by congruence and repeated unfolding), hence we would have $C[rec_x.P] \xrightarrow{\tau^{h+1}}$, which contradicts $C[rec_x.P] \Downarrow_h$. This proves that $C[Q] \Downarrow$, hence $C[Q] \Downarrow_{h'}$ for some h'. Take $k = \max\{h, h'\}$. Note that $C[rec_x.P] \sim_{\mathrm{sb}} C[P^{(k+1)}[rec_x.P]] \Downarrow_k$ and $C[Q] \sim_{\mathrm{sb}} C[P^{(k+1)}[Q]] \Downarrow_k$. Since $C[P^{(k+1)}[\cdot]]$ is $k+1$-guarded, by Lemma A.3(1) we obtain that: $o(C[P^{(k+1)}[rec_x.P]]) = o(C[P^{(k+1)}[Q]])$. But this implies (a), because $C[rec_x.P] \sim C[P^{(k+1)}[rec_x.P]]$ and $C[Q] \sim C[P^{(k+1)}[Q]]$.

We now assume for a generic a that $C[rec_x.P]_a \downarrow$, i.e. that $C[rec_x.P] \Downarrow a$, and check requirement (b) of simulation up to, that is, $C[rec_x.P]_a \sim R \sim C[Q]_a$. In the first place, note that $C[Q] \Downarrow a$ as well: the argument mimics that given above to show that $C[Q] \Downarrow$, so we are not going to repeat it. Now, take k such that both $C[rec_x.P] \Downarrow_k a$ and $C[Q] \Downarrow_k a$. Again, note that $C[rec_x.P] \sim_{\mathrm{sb}} C[P^{(k+1)}[rec_x.P]] \Downarrow_k a$ and that $C[Q] \sim_{\mathrm{sb}} C[P^{(k+1)}[Q]] \Downarrow_k a$. Since $C[P^{(k+1)}[\cdot]$ is $k+1$-guarded, by Lemma A.3(2) we obtain that there is a context $C'[\cdot]$ such that $(C[P^{k+1}[rec_x.P])_a \sim C'[rec_x.P] \stackrel{\mathrm{def}}{=} A$ and $(C[P^{(k+1)}[Q]])_a \sim C'[Q] \stackrel{\mathrm{def}}{=} B$. Clearly $\langle A, B\rangle \in R$ by definition. Now, from $C[rec_x.P] \sim C[P^{(k+1)}[rec_x.P]]$ and $C[Q] \sim C[P^{(k+1)}[Q]]$ we get $(C[rec_x.P])_a \sim A$ and $(C[Q])_a \sim B$, respectively. In the end, we have obtained: $(C[rec_x.P])_a \sim A\,R\,B \sim (C[Q])_a$, that is (b). □

To obtain Lemma 5.5, take $C[\cdot] = [\cdot]$, the empty context, in the previous proposition.