CPO Models for a Class of GSOS Languages

Luca Aceto* Anna Ingólfsdóttir†

BRICS‡
Department of Mathematics and Computer Science
Aalborg University
Fredrik Bajersvej 7E
9220 Aalborg Ø, Denmark

Abstract

In this paper, we present a general way of giving denotational semantics to a class of languages equipped with an operational semantics that fits the GSOS format of Bloom, Istrail and Meyer. The canonical model used for this purpose will be Abramsky's domain of synchronization trees, and the denotational semantics automatically generated by our methods will be guaranteed to be fully abstract with respect to the finitely observable part of the bisimulation preorder. In the process of establishing the full abstraction result, we also obtain several general results on the bisimulation preorder (including a complete axiomatization for it), and give a novel operational interpretation of GSOS languages.

1 Introduction

In this paper, we present a general way of giving denotational semantics to a class of languages equipped with an operational semantics that fits the GSOS format of Bloom, Istrail and Meyer [6]. The canonical model used for this purpose will be Abramsky's domain of synchronization trees $\mathcal{D}$ presented in [1], and the denotational semantics automatically generated by our methods will be guaranteed to be fully abstract with respect to the finitely observable part of the bisimulation preorder studied in, e.g., [9, 11]. Moreover, in the process of establishing the full abstraction result, we also give an algorithm, along the lines of those given in [2], to generate a complete axiomatization of the bisimulation preorder. As a byproduct of our denotational semantics, we shall be able to establish very general results about the bisimulation preorder that would be hard to prove using purely operational definitions. (For an example, cf. Thm 4.10.)

---

*On leave from School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, UK. Partially supported by HCM project EXPRESS. Email: luca@iesd.auc.dk.

†Partially supported by a grant from the Danish Research Council. Email: annai@iesd.auc.dk.

‡Basic Research in Computer Science, Centre of the Danish National Research Foundation.

The class of GSOS systems we shall give denotational semantics to will have the structure of most standard process algebras (see, e.g., [12, 5]). They will consist of a set of operations to construct finite, acyclic process graphs, and a facility for the recursive definition of behaviours. Borrowing a terminology introduced in [10] in the context of denotational semantics, we shall refer to these languages as *compact* GSOS languages. Their operational semantics will be given in terms of a variation on the standard model of labelled transition systems that takes divergence information into account. This will be done in such a way that the bisimulation preorder is a precongruence with respect to all the operators in the language. In order to obtain this substitutivity result, special care must be taken in interpreting negative premises in GSOS rules; in particular, negative premises will only be interpreted over convergent (or fully specified) processes. Intuitively, this is because, in order to find out what a process can*not* do, we need to know precisely what its capabilities are, and the initial behaviour of a divergent process is only partially specified. A consequence of our choice is that, for example, the rule

$$\frac{x \overset{a}{\nrightarrow}}{odd(x) \overset{a}{\rightarrow} odd(x)} \tag{1}$$

cannot be used to derive that $odd(\Omega)$ has an a-labelled transition to itself, where Ω denote the typical totally divergent process with no transitions. Our first main result is that, with our choice of operational semantics for GSOS languages, the bisimulation preorder is substitutive with respect to all language contexts. Moreover, as a consequence of general results established by Abramsky in [1], we are able to give a characterization of the finitely observable (or finitary) part of the bisimulation preorder for every GSOS language. Intuitively, this is the preorder obtained by restricting the bisimulation preorder to observations of finite depth.

We then show how to automatically give a denotational semantics for a GSOS language in terms of Abramsky's domain of synchronization trees $\mathcal{D}$. To this end, it is sufficient to endow Abramsky's model with an appropriate continuous algebra structure in the sense of [8]. This we do by showing how the GSOS rules defining the operational semantics of an operation symbol f of a compact GSOS language can be used to define a continuous function $\mathbf{f}_{\mathcal{D}}$ of the appropriate arity over the domain of synchronization trees $\mathcal{D}$. In defining the semantic counterparts of the operations in a compact GSOS language, we shall rely on a description of the domain $\mathcal{D}$ presented in [10], where it is shown how to reconstruct $\mathcal{D}$ from a suitable preorder over finite synchronization trees. This view of $\mathcal{D}$ will allow us to define each semantic operation $\mathbf{f}_{\mathcal{D}}$ in stepwise fashion from monotonic operations over finite synchronization trees. We hope that this choice will make the presentation more accessible to readers who are unfamiliar with domain theory [13].

As a result of our general framework, we shall then show that the denotational semantics so obtained is guaranteed to be in complete agreement with the chosen behavioural semantics. More precisely, for every compact GSOS language, the denotational semantics produced by the general approach presented in this paper is always fully abstract with respect to the finitary part of the bisimulation preorder.

Due to strict space limitations, this paper contains no proofs, very few technical details, no comparison with related work and no concluding remarks. These may be found in the full version of this paper [3].

2 Preliminaries

We begin by reviewing the basic notions on transition systems and domain theory that will be needed in this study.

2.1 Labelled Transition Systems with Divergence

The operational semantics of the languages considered in this paper will be given in terms of a variation on the model of labelled transition systems that takes divergence information into account. We refer the interested readers to, e.g., [7, 11, 15] for motivation and more information on (variations on) this semantic model for reactive systems.

Definition 2.1 *A* labelled transition system with divergence (lts) *is a quadruple*

$$(\mathsf{P}, \mathsf{Lab}, \rightarrow, \uparrow)$$

where: P *is a set of* processes*, ranged over by* s, t*;* Lab *is a set of* labels*, ranged over by* ℓ*;* $\rightarrow \subseteq \mathsf{P} \times \mathsf{Lab} \times \mathsf{P}$ *is a* transition relation*;* $\uparrow \subseteq \mathsf{P}$ *is a* divergence predicate*, notation* $s \uparrow$*. As usual, we shall write* $s \xrightarrow{\ell} t$ *in lieu of* $(s, \ell, t) \in \rightarrow$*.*

We write $s \downarrow$, read "s definitely converges", iff it is not the case that $s \uparrow$. A useful source of examples for labelled transition systems with divergence is the set of finite synchronization trees over a set of labels Lab, denoted by $\mathsf{ST}(\mathsf{Lab})$. These are the sets generated by the following inductive definition:

$$\frac{\{\ell_i \in \mathsf{Lab}, t_i \in \mathsf{ST}(\mathsf{Lab})\}_{i \in I}}{\{\langle \ell_i, t_i \rangle \mid i \in I\}[\cup\{\bot\}] \in \mathsf{ST}(\mathsf{Lab})}$$

where I is a finite index set, and the notation $[\cup\{\bot\}]$ means optional inclusion of $\bot$. The set of finite synchronization trees $\mathsf{ST}(\mathsf{Lab})$ can be turned into a labelled transition system with divergence by stipulating that, for $t \in \mathsf{ST}(\mathsf{Lab})$: $t \uparrow$ iff $\bot$ is in t, and $t \xrightarrow{\ell_i} t_i$ iff $\langle \ell_i, t_i \rangle$ is in t. The behavioural relation over processes that we shall study in this paper is that of *prebisimulation* [11, 7, 15] (also known as *partial bisimulation* [1]).

Definition 2.2 *Let* $(\mathsf{P}, \mathsf{Lab}, \rightarrow, \uparrow)$ *be an lts. Let* $\mathsf{Rel}(\mathsf{P})$ *denote the set of binary relations over* P*. Define the functional* $F : \mathsf{Rel}(\mathsf{P}) \rightarrow \mathsf{Rel}(\mathsf{P})$ *by:*

$$\begin{array}{rcl} F(\mathcal{R}) & = & \{(s,t) \mid \forall \ell \in \mathsf{Lab} \\ & & \quad \bullet\ s \xrightarrow{\ell} s' \Rightarrow \exists t' :\ t \xrightarrow{\ell} t' \textit{ and } s' \,\mathcal{R}\, t' \\ & & \quad \bullet\ s \downarrow \Rightarrow t \downarrow \textit{ and } [t \xrightarrow{\ell} t' \Rightarrow \exists s' :\ s \xrightarrow{\ell} s' \textit{ and } s' \,\mathcal{R}\, t']\} \end{array}$$

A relation $\mathcal{R}$ is a prebisimulation iff $\mathcal{R} \subseteq F(\mathcal{R})$. We write $s_1 \lesssim s_2$ iff there exists a prebisimulation $\mathcal{R}$ such that $s_1 \mathrel{\mathcal{R}} s_2$.

The relation $\lesssim$ is a preorder over P based on a variation on bisimulation equivalence [12]. Its kernel will be denoted by $\sim$, i.e., $\sim = \lesssim \cap \lesssim^{-1}$. In this study, we shall be interested in relating the notion of prebisimulation to a denotationally defined preorder on processes. As the preorders induced by a denotational semantics are completely determined by how they act on *finite processes*, we shall be interested in comparing them with the "finitely observable", or *finitary*, part of the bisimulation preorder in the sense of, e.g., [7]. The following definition is from [1].

Definition 2.3 *The* finitary preorder $\lesssim^F$ *is defined on any lts by*

$$s \lesssim^F s' \Leftrightarrow \forall t \in \mathsf{ST}(\mathsf{Lab}).\ t \lesssim s \Rightarrow t \lesssim s' \ .$$

An alternative method for using the functional F to obtain a behavioural preorder is to apply it inductively as in [1, Page 165]. The ω-iterate of this inductive construction will be denoted by $\lesssim_\omega$. The preorders $\lesssim$, $\lesssim_\omega$ and $\lesssim^F$ are related thus: $\lesssim \ \subseteq \ \lesssim_\omega \ \subseteq \ \lesssim^F$. Moreover the inclusions are, in general, strict. The interested reader is referred to [1] for more information on these preorders.

2.2 A Domain Equation for Synchronization Trees

In this section we recall Abramsky's domain equation for synchronization trees, and introduce the background in domain theory that is necessary to understand the paper. The interested reader is referred to, e.g., [13] for more general information on the theory of domains.

The canonical domain we shall use to give a denotational semantics to a class of GSOS languages is the domain of synchronization trees over a countable set of labels Lab considered by Abramsky in his seminal paper [1]. This is defined to be the initial solution $\mathcal{D}(\mathsf{Lab})$ of the domain equation

$$\mathcal{D}(\mathsf{Lab}) \cong (\mathbf{1})_\perp \oplus P[\sum_{\ell \in \mathsf{Lab}} \mathcal{D}(\mathsf{Lab})] \tag{2}$$

where $\mathbf{1}$ is the one point domain, $(\cdot)_\perp$ is lifting, $\oplus$ is coalesced sum, $\sum$ is separated sum, and $P[D]$ denotes the Plotkin powerdomain of D (cf. [13] for details on these domain-theoretic operations). We henceforth omit the parameter Lab as it will always be clear from the context.

To streamline the presentation and make our results more accessible to uninitiated readers, in this study we shall abstract completely from the domain-theoretic description of $\mathcal{D}$ given by (2). Our description of the domain of synchronization trees $\mathcal{D}$ will follow the one given in [10], and we shall rely on results presented in that reference that show how to construct $\mathcal{D}$ starting from a suitable preorder on the set of finite synchronization trees $\mathsf{ST}(\mathsf{Lab})$. Our reconstruction of $\mathcal{D}$ will be given in three steps: First of all, we shall define a preorder $\sqsubseteq$ on the set of synchronization trees $\mathsf{ST}(\mathsf{Lab})$. This preorder will be a reformulation of the Egli-Milner preorder over $\mathsf{ST}(\mathsf{Lab})$ presented in [10]. Secondly, we shall relate the poset of compact elements of $\mathcal{D}$ to the poset of equivalence classes induced by $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$. Finally, we shall use the fact that $\mathcal{D}$ is the ideal completion of its poset of compact elements to relate it to $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$. This approach will allow us to factor the definition of the continuous algebra structure [8] on $\mathcal{D}$ given in Sect. 4 in three similar steps, hopefully making it simpler to understand.

Definition 2.4 *We define $\sqsubseteq$ as the least binary relation over $\mathsf{ST}(\mathsf{Lab})$ satisfying:*

$$\begin{array}{lll} t \sqsubseteq u \quad \textit{if} & \textit{(1)} & \langle \ell, t' \rangle \in t \Rightarrow \exists \langle \ell, u' \rangle \in u : \; t' \sqsubseteq u' \quad \textit{and} \\ & \textit{(2)} & \bot \in u \Rightarrow \bot \in t \quad \textit{and} \\ & \textit{(3)} & \langle \ell, u' \rangle \in u \Rightarrow \big(\bot \in t \;\textit{or}\; \exists \langle \ell, t' \rangle \in t : \; t' \sqsubseteq u'\big) \end{array}$$

The relation $\sqsubseteq$ so defined is easily seen to be a preorder over $\mathsf{ST}(\mathsf{Lab})$, whose kernel will be denoted by $\simeq$. We can now relate the preorder of synchronization trees $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ with the poset of compact elements of $\mathcal{D}$ in a way that will allow us to define, in a canonical way, continuous operations on $\mathcal{D}$ from monotonic ones on $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$. First of all, we recall here that $\mathcal{D}$ is, up to isomorphism, the algebraic complete partial order (cpo) whose poset of compact elements $(\mathcal{K}(\mathcal{D}), \sqsubseteq_{\mathcal{K}(\mathcal{D})})$ is defined in [1, Def. 3.9]. $\mathcal{K}(\mathcal{D})$ is a subset of the set of finite synchronization trees, and the relations $\sqsubseteq$ and $\sqsubseteq_{\mathcal{K}(\mathcal{D})}$ may be shown to coincide over it. As a consequence of this observation, from now on we shall always use $\sqsubseteq$ as our notion of preorder on $\mathcal{K}(\mathcal{D})$.

For a synchronization tree $t = \{\langle \ell_i, t_i \rangle \mid 1 \leq i \leq n\}[\cup\{\bot\}]$, its recursive convex-closure t^c is inductively defined as follows:

$$t^c \quad \triangleq \quad \mathsf{Con}(\{\langle \ell_i, t_i^c \rangle \mid 1 \leq i \leq n\}[\cup\{\bot\}]) \tag{3}$$

where Con denotes the convex-closure operation given in [1, Def. 3.2]. It is not difficult to see that, for every $t \in \mathsf{ST}(\mathsf{Lab})$, t^c is a compact element of $\mathcal{D}$. Moreover, the function $(\cdot)^c : (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq) \to (\mathcal{K}(\mathcal{D}), \sqsubseteq)$ is monotonic and maps trees that are related by $\simeq$ to the same compact element of $\mathcal{D}$. Assume now that $\mathbf{f}_{\mathsf{ST}} : (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)^l \to (\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ is a monotonic function. Using the fact that each element of $\mathcal{K}(\mathcal{D})$ is a synchronization tree, we may naturally use $\mathbf{f}_{\mathsf{ST}}$ to define a function $\mathbf{f}_{\mathcal{K}(\mathcal{D})} : (\mathcal{K}(\mathcal{D}), \sqsubseteq)^l \to (\mathcal{K}(\mathcal{D}), \sqsubseteq)$ as follows:

$$\mathbf{f}_{\mathcal{K}(\mathcal{D})}(\vec{d}) \quad \triangleq \quad \Big(\mathbf{f}_{\mathsf{ST}}(\vec{d})\Big)^c \; . \tag{4}$$

It is easy to see that the function so defined is guaranteed to be monotonic. This implies that we can use (4) to lift any Σ-preorder structure [10] on $(\mathsf{ST}(\mathsf{Lab}), \sqsubseteq)$ to a Σ-poset structure on $(\mathcal{K}(\mathcal{D}), \sqsubseteq)$ in the sense of [8]. In Sect. 4.2 we shall take advantage of this fact. Finally, from the theory of powerdomains [13], we know that the domain of synchronization trees $\mathcal{D}$ is, up to isomorphism, the ideal completion of the poset of compact elements $\mathcal{K}(\mathcal{D})$. As a result of this observation, we can extend any monotonic function $\mathbf{f}_{\mathcal{K}(\mathcal{D})} : (\mathcal{K}(\mathcal{D}), \sqsubseteq)^l \to (\mathcal{K}(\mathcal{D}), \sqsubseteq)$ to a continuous function $\mathbf{f}_{\mathcal{D}} : (\mathcal{D}, \sqsubseteq_{\mathcal{D}})^l \to (\mathcal{D}, \sqsubseteq_{\mathcal{D}})$ by:

$$\mathbf{f}_{\mathcal{D}}(\vec{k}) \triangleq \bigsqcup \Big\{ \mathbf{f}_{\mathcal{K}(\mathcal{D})}(\vec{d}) \mid \vec{d} \in \mathcal{K}(\mathcal{D})^l \text{ and } \vec{d} \sqsubseteq_{\mathcal{D}} \vec{k} \Big\} \; . \tag{5}$$

The interested reader is invited to consult, e.g., [8, Sect. 3.3] for a discussion of the properties afforded by this canonical extension. In what follows, all we shall need to know is that $\mathbf{f}_{\mathcal{D}}$ is guaranteed to be continuous. Thus (5) can be used to conservatively extend any Σ-po algebra structure on $(\mathcal{K}(\mathcal{D}), \sqsubseteq)$ to a continuous algebra structure on $\mathcal{D}$.

3 GSOS Languages

Let Var be a denumerable set of *meta-variables* ranged over by x, y. A *signature* Σ consists of a set of *operation symbols*, disjoint from Var, together with a function *arity* that assigns a natural number to each operation symbol. Throughout this paper, following the standard lines of algebraic semantics (see, e.g., [8]), we shall assume that signatures contain a distinguished function symbol Ω of arity zero to denote the totally unspecified, or divergent, process, i.e., a process about whose behaviour we have no information. The set $\mathbb{T}(\Sigma, \mathsf{Var})$ of *terms* over Σ and Var (abbreviated to $\mathbb{T}(\Sigma)$ when the set of variables is clear from the context or immaterial) is defined as usual. We shall use $P, Q, \ldots$ to range over terms and the symbol $\equiv$ for the relation of syntactic equality on terms. $\mathrm{T}(\Sigma)$ is the set of *closed* terms over Σ, i.e., terms that do not contain variables. Constants, i.e. terms of the form $f()$, will be abbreviated as f. A Σ-*context* $C[\vec{x}]$ is a term in which at most the variables $\vec{x}$ appear. $C[\vec{P}]$ is $C[\vec{x}]$ with x_i replaced by P_i wherever it occurs. Besides terms we have *actions*, elements of some given finite set Act, which is ranged over by a, b, c.

Definition 3.1 (GSOS Rules) *Suppose Σ is a signature. A* GSOS rule ρ *over Σ is an inference rule of the form:*

$$\frac{\bigcup_{i=1}^{l}\left\{x_i \stackrel{a_{ij}}{\rightarrow} y_{ij} | 1 \leq j \leq m_i\right\} \quad \cup \quad \bigcup_{i=1}^{l}\left\{x_i \stackrel{b_{ik}}{\not\rightarrow} | 1 \leq k \leq n_i\right\}}{f(x_1, \ldots, x_l) \stackrel{c}{\rightarrow} C[\vec{x}, \vec{y}]} \tag{6}$$

where all the variables are distinct, $m_i, n_i \geq 0$, f is an operation symbol from Σ with arity l, $C[\vec{x}, \vec{y}]$ is a Σ-context, and the a_{ij}, b_{ik}, and c are actions in Act.

The operation symbol f in rules of the form (6) will be referred to as the *principal operation* of the rule. If, for some i, $m_i > 0$ then we say that ρ *tests its i-th argument positively*. Similarly if $n_i > 0$ then we say that ρ *tests its i-th argument negatively*. An operation f *tests its i-th argument positively (resp. negatively)* if it occurs as principal operation of a rule that tests its i-th argument positively (resp. negatively). We say that an operation f *tests its i-th argument* if it tests it either positively or negatively.

Definition 3.2 (GSOS Systems) *A* GSOS system *is a pair $G = (\Sigma_G, R_G)$, where Σ_G is a finite signature and R_G is a finite set of GSOS rules over Σ_G containing no rules with Ω as principal operation.*

An example of GSOS system, the language $\mathcal{L}$, is presented in Fig. 1. We shall use this concrete language as a running example throughout the paper to illustrate our definitions and results. The language $\mathcal{L}$ is a subset of ACP [5] with action prefixing in lieu of general sequential composition. Its parallel composition operator, denoted by $||$, is parameterized with respect to a partial, commutative and associative communication function $\gamma : \mathsf{Act} \times \mathsf{Act} \rightharpoonup \mathsf{Act}$. An operation in $\mathcal{L}$ that uses the power of negative premises, at least in the presence of a non-trivial priority structure on actions, is the priority operation θ of Baeten, Bergstra and Klop [4]. In order to define this operation, we assume a given partial ordering relation $>$ on Act. Intuitively, $b > a$ is interpreted as "action b has priority over action a". The sub-language of $\mathcal{L}$ consisting only of the operations Ω, δ, $a.\_$ and $+$ will be denoted by $\mathsf{FINTREE}_\Omega$.

Signature	*Arity*	*Rules*
δ	0	no rules
Ω	0	no rules
$a.\_ \quad (a \in \mathsf{Act})$	1	$a.x \stackrel{a}{\rightarrow} x$
$+$	2	$\frac{x \stackrel{a}{\rightarrow} x'}{x+y \stackrel{a}{\rightarrow} x'} \quad \frac{y \stackrel{a}{\rightarrow} y'}{x+y \stackrel{a}{\rightarrow} y'}$
$\|\|$	2	$\frac{x \stackrel{a}{\rightarrow} x'}{x\|\|y \stackrel{a}{\rightarrow} x'\|\|y} \quad \frac{y \stackrel{a}{\rightarrow} y'}{x\|\|y \stackrel{a}{\rightarrow} x\|\|y'}$ $\frac{x \stackrel{a}{\rightarrow} x', y \stackrel{b}{\rightarrow} y'}{x\|\|y \stackrel{c}{\rightarrow} x'\|\|y'} \quad \gamma(a,b) \simeq c$
θ	1	$\frac{x \stackrel{a}{\rightarrow} x', (\forall b > a) x \stackrel{b}{\not\rightarrow}}{\theta(x) \stackrel{a}{\rightarrow} \theta(x')}$

Figure 1: The language $\mathcal{L}$

GSOS systems have been introduced and studied in depth in [6]. Intuitively, a GSOS system gives a language, whose constructs are the operations in the signature Σ_G, together with a Plotkin-style structural operational semantics [14] for it defined by the set of conditional rules R_G. In this study, the operational semantics of a GSOS system will be given in terms of labelled transition systems with divergence. In order to obtain this novel interpretation, we aim at using the rules in a GSOS system G to define a divergence predicate over terms and a transition relation in such a way that our definitions: (1) specialize to those originally given by Bloom, Istrail and Meyer in their seminal study [6] when divergence is not taken into account; (2) give results that are in agreement with those already presented in the literature when applied to known process description languages; and (3) produce operators that are well-behaved with respect to the notion of prebisimulation, i.e., operations for which prebisimulation is a precongruence.

First of all, we shall use the rules in a GSOS systems to define a divergence (or underspecification) predicate on the set of closed terms over Σ_G. In fact, as is common practice in the literature on process algebras, we shall define the notion of convergence, and use it to define the divergence predicate we are after. Intuitively, a term P is convergent if the set of its initial transitions is fully specified. The basic divergent term is Ω, the totally unspecified process. A term of the form $f(\vec{P})$ is convergent iff the set of its initial transitions only depends on those arguments P_is whose initial behaviour is completely known. This informal discussion motivates the following definition.

Definition 3.3 (Convergence) *Let $G = (\Sigma_G, R_G)$ be a GSOS system. The convergence predicate $\downarrow_G$ (abbreviated to $\downarrow$ when the GSOS system G is clear from the context) is the least predicate over $\mathrm{T}(\Sigma_G)$ that satisfies the following clause:*

$$f(P_1, \ldots, P_l) \downarrow_G \text{ if } f \neq \Omega \text{ and, for every argument } i \text{ of } f, \text{ if } f \text{ tests } i \text{ then } P_i \downarrow_G \quad .$$

We write $P \uparrow_G$ iff it is not the case that $P \downarrow_G$.

When applied to the language $\mathcal{L}$, Def. 3.3 gives the following convergence predicate: (1) $\delta \downarrow$; (2) if $P \downarrow$ and $Q \downarrow$, then $P + Q \downarrow$ and $P \| Q \downarrow$; and (3) if $P \downarrow$ then $\theta(P) \downarrow$. The reader familiar with the literature on prebisimulation over CCS-like languages will have noted the similarity of this definition with those given in, e.g., [9, 7, 1].

We shall now present our non-standard operational semantics for GSOS languages. As stated above, we take as our starting point the original theory developed by Bloom, Istrail and Meyer. However, in the presence of divergence information, we shall interpret negative transition formulae over *convergent processes* only. Intuitively, to know that a process cannot initially perform a given action, we need to find out precisely all the actions that it can perform. If a process is divergent, its set of initial actions is not fully specified; thus we cannot be sure whether such a process satisfies a negative transition formula or not. Informally, the intent of a GSOS rule in our setting is as follows. Suppose that we are wondering whether $f(\vec{P})$ is capable of taking a c-step. We look at each rule of the form (6) with principal operation f and action c in turn. We inspect each positive premise $x_i \stackrel{a_{ij}}{\to} y_{ij}$, checking if P_i is capable of taking an a_{ij}-step for each j and if so calling the a_{ij}-children Q_{ij}. We also check the negative premises; this involves finding out whether P_i is convergent, and, in that case, if P_i is *in*capable of taking a b_{ik}-step for each k. If so, then the rule *fires* and $f(\vec{P}) \stackrel{c}{\to} C[\vec{P}, \vec{Q}]$. We write $\to_G$ for the transition relation for G defined in this way. The lts with divergence specified by a GSOS system G is then given by $\mathsf{lts}(G) = (\mathrm{T}(\Sigma_G), \to_G, \uparrow_G)$. The largest prebisimulation over $\mathsf{lts}(G)$ will be denoted by $\lesssim_G$, and its kernel by $\sim_G$. (The subscript G will be omitted from these relations when this causes no confusion).

Example: We exemplify our approach using our running example, the language $\mathcal{L}$, by considering some identities involving simple terms that use the priority operation θ.

The term $\theta(\Omega)$ is divergent, as Ω is. Moreover it does not have any transition because Ω has none. We thus have that $\theta(\Omega) \sim \Omega$.

Consider a term of the form $P \equiv a.\delta + \Omega$, with a a maximal element in the poset $(\mathsf{Act}, >)$, i.e., with a an action with maximal priority. Then the rule for θ with action a has no negative premises, and it can be used to establish the transition $\theta(P) \stackrel{a}{\to} \theta(\delta)$. Indeed, this is the only transition that is possible from $\theta(P)$. As $\theta(P)$ is divergent, as P is, it is easy to see that $\theta(P) \sim a.\delta + \Omega$.

On the other hand, if a is *not* maximal in the poset $(\mathsf{Act}, >)$, the rule for θ with action a will have at least one negative premise. As P is divergent, that rule cannot be used to derive a transition from the term $\theta(P)$. It thus follows that $\theta(P) \sim \Omega$, if a is not maximal in the poset $(\mathsf{Act}, >)$. □

We are now ready to establish the first main result of this paper. Namely, we shall prove that the operations of a GSOS system preserve the semantic notion of prebisimulation.

Theorem 3.4 *Let G be a GSOS system. Then $\lesssim_G$ is a precongruence for all operation symbols f of G,* i.e., $(\forall i : P_i \lesssim_G Q_i) \;\Rightarrow\; f(\vec{P}) \lesssim_G f(\vec{Q})$.

3.1 GSOS Languages with Recursion

In this section we consider GSOS languages that may include recursion. Let $G = (\Sigma_G, R_G)$ be a GSOS system, and let PVar be a fresh denumerable set of *process variables* $(X, Y \in \mathsf{PVar})$. The set of recursive terms over Σ_G and PVar, denoted by $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$, is given by the following BNF syntax:

$$P ::= X \mid f(P_1, \ldots, P_l) \mid \mathrm{fix}(X = P)$$

where $X \in \mathsf{PVar}$, f is an operation symbol in Σ_G of arity l and fix is a binding construct. This gives rise to the usual notions of free and bound variables in terms. The set of closed recursive terms (or *programs*) will be denoted by $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. We shall assume a standard notion of substitution of terms for free process variables, and use $P\{Q/X\}$ to denote term P in which each free occurrence of X has been replaced by Q, after possibly renaming bound variables in P. We shall now define an operational semantics for the set of programs $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ in terms of an lts with divergence, following the techniques presented in Sect. 3. In this study, the operational semantics of recursion will be given by the standard unfolding rule:

$$\frac{x\{\mathrm{fix}(X = x)/X\} \xrightarrow{a} y}{\mathrm{fix}(X = x) \xrightarrow{a} y} \tag{7}$$

This will ensure that recursive terms are interpreted as fixed-points, and that the following equation holds:

$$\mathrm{fix}(X = P) = P\{\mathrm{fix}(X = P)/X\} \; . \tag{8}$$

In order to define the operational semantics of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, we need, first of all, to extend the convergence predicate to $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$.

Definition 3.5 *The convergence predicate $\downarrow_{Grec}$ (abbreviated to $\downarrow$ when the GSOS system G is clear from the context) is the least predicate over $\mathsf{CREC}(\Sigma_G, \mathsf{Var})$ that satisfies the following clauses:*

1. $f(P_1, \ldots, P_l) \downarrow_{Grec}$ *if $f \neq \Omega$, and for every argument i of f, if f tests i then $P_i \downarrow_{Grec}$;*
2. $\mathrm{fix}(X = P) \downarrow_{Grec}$ *if $P\{\mathrm{fix}(X = P)/X\} \downarrow_{Grec}$.*

Again, we write $P \uparrow_{Grec}$ iff it is not the case that $P \downarrow_{Grec}$.

The motivation for the above definition is the following: a term P is divergent if its initial transitions are not fully specified. This occurs either when the initial behaviour of term P depends on underspecified arguments like Ω or in the presence of unguarded recursive definitions. For example, the terms $\mathrm{fix}(X = X)$ and $\mathrm{fix}(X = \mathit{odd}(X))$, where *odd* is the operation given by rule (1), are *not* convergent as the initial behaviour of these processes depends on itself. It is immediate to see that the predicates $\downarrow_G$ and $\downarrow_{Grec}$ coincide over $\mathrm{T}(\Sigma_G)$, the set of recursion-free terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$.

We shall now show how to associate a transition relation with a GSOS language with recursion by putting the extra structure given by the convergence predicate to good use.

We construct the relation $\rightarrow_{Grec}$ over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ in two steps. In the first step of the construction, we derive the transitions emanating from *convergent terms* by induction on the convergence predicate following the approach outlined in the previous section, and using (7) to derive the transitions of recursive terms. In the second, we use the information about the transitions that are possible for convergent terms to determine the outgoing transitions for all the terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. For a divergent term P, a transition $P \xrightarrow{a}_{Grec} Q$ holds iff it is provable from the rules in R_G and those for recursion in the standard sense, with the proviso that every time we need to establish the validity of a negative premise we appeal to the information on the transitions that are possible from convergent terms discovered in the first step of our construction. To exemplify this construction on a pathological case, let us consider the term $\mathrm{fix}(X = \mathit{odd}(X))$, where the operation *odd* is given by rule (1). The operation *odd* is standardly used in the literature to show that negative premises and unguarded recursive definitions can lead to inconsistent specifications. Note, first of all, that $\mathrm{fix}(X = \mathit{odd}(X))$ is a divergent term. It is then easy to see that, because of our requirement that negative premises in rules be interpreted over convergent terms only, rule (1) can never be applied to derive a transition for $\mathrm{fix}(X = \mathit{odd}(X))$. Thus we have that this term has *no* transition and is divergent, i.e., that $\mathrm{fix}(X = \mathit{odd}(X)) \sim \Omega$.

With the above definitions, the operational semantics of a GSOS language with recursion $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is given by the lts with divergence

$$\mathsf{lts}(G_{rec}) = (\mathsf{CREC}(\Sigma_G, \mathsf{PVar}), \mathsf{Act}, \rightarrow_{Grec}, \uparrow_{Grec}) \ .$$

As a corollary of general results by Abramsky, we can give the following general characterization of the finitary bisimulation preorder over $\mathsf{lts}(G_{rec})$ for arbitrary G:

Proposition 3.6 *The preorders $\lesssim^F$ and $\lesssim_\omega$ coincide over $\mathsf{lts}(G_{rec})$.*

4 Denotational Semantics

In this section we shall present a general technique to give denotational semantics in terms of the Plotkin powerdomain of synchronization trees (see Sect. 2.2) for a class of GSOS languages with recursion. The denotational semantics will be guaranteed to be *fully abstract*, in the sense of Milner and Plotkin, with respect to the finitary part of the prebisimulation relation $\lesssim_\omega$. The languages that we shall consider have the structure of most standard process calculi (see, e.g., [12, 5]); they will consist of a set of operations to build finite, acyclic labelled transition systems and a facility for recursive definitions of behaviours. Thus we shall consider GSOS languages with recursion in which infinite behaviours can only be defined by means of recursive definitions.

We assume that the reader is familiar with the basic notions of ordered and continuous algebras (see, e.g., [8, 10]); however, in what follows we give a quick overview of the way a denotational semantics can be given to a recursive language like $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$ following the standard lines of algebraic semantics [8]. The interested reader is invited to consult [8] for an explanation of the theory.

Let $\mathcal{A}$ be any Σ_G-cpo, i.e., a Σ_G-algebra whose carrier is an algebraic cpo and whose operations are interpreted as continuous functions. A denotational semantics for the language $\mathsf{REC}(\Sigma_G, \mathsf{PVar})$ is given by the mapping $\mathcal{A}[\![\cdot]\!] : \mathsf{REC}(\Sigma_G, \mathsf{PVar}) \to [\mathrm{ENV}_\mathcal{A} \to \mathcal{A}]$ defined in, e.g., [8, Def. 4.2.3], where $\mathrm{ENV}_\mathcal{A} = [\mathsf{PVar} \to \mathcal{A}]$ is the set of $\mathcal{A}$-environments, ranged over by the meta-variable ρ. For each $P \in \mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, $\mathcal{A}[\![P]\!]\rho$ does not depend on the environment ρ. Hence the denotation of a closed term P will be denoted by $\mathcal{A}[\![P]\!]$. A general result about the semantic mapping $\mathcal{A}[\![\cdot]\!]$ states that, for any $P \in \mathsf{REC}(\Sigma_G, \mathsf{PVar})$, there exists an infinite sequence of *finite approximations* $P^n \in \mathbb{T}(\Sigma_G, \mathsf{PVar})$ $(n \in \mathbb{N})$ such that

$$\mathcal{A}[\![P]\!] = \bigsqcup\nolimits_{n\geq 0} \mathcal{A}[\![P^n]\!] \ . \tag{9}$$

For any binary relation $\mathcal{R}$ over $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$, the algebraic part of $\mathcal{R}$, denoted by $\mathcal{R}^A$ is defined as follows [8]:

$$P \ \mathcal{R}^A \ Q \quad \Leftrightarrow \quad \text{for every } P^n\text{, there exists } Q^m \text{ such that } P^n \ \mathcal{R} \ Q^m \ .$$

We say that $\mathcal{R}$ is *algebraic* iff $\mathcal{R}$=$\mathcal{R}^A$. Intuitively, a relation is algebraic if it is completely determined by how it behaves on recursion-free terms. Because of Eqn. 9, the relations over terms induced by a denotational semantics are always algebraic. We shall make use of this fact in the technical developments to follow.

4.1 Compact GSOS Systems

The following notion from [2] will allow us to pin down precisely a class of GSOS operations that map finite processes to finite processes. The semantic counterparts of these operations will have the property of being *compact* in the sense of [10], i.e., of mapping compact elements in the Plotkin powerdomain of synchronization trees to compact elements.

Definition 4.1 *A GSOS rule of the general form (6) is* linear *if each variable occurs at most once in the target and, for each argument i that is tested positively, x_i does not occur in the target and at most one of the y_{ij}'s does. An* *operation from a GSOS system G is* linear *iff all rules for it are linear. Finally, G itself is* linear *iff it only contains linear rules.*

The format of linear rules is a restriction of the general GSOS format in that no copying of arguments is allowed and no argument for which there is a positive premise may appear in the target of a rule. Moreover, there may be possibly many positive premises for an argument x_i in a rule, but at most one of the y_{ij}'s may appear in its target. As far as we know, all the operations occurring in the standard process algebras are linear.

Definition 4.2 ([2]) *A GSOS system G is* syntactically well-founded *iff there exists a function w from operation symbols in Σ_G to natural numbers such that, for each rule $\rho \in R_G$ with principal operation symbol f and target $C[\vec{x}, \vec{y}]$ the following conditions hold:*

- *if ρ has no positive premises then $W(C[\vec{x}, \vec{y}]) < w(f)$, and*
- *$W(C[\vec{x}, \vec{y}]) \leq w(f)$ otherwise,*

where $W : \mathbb{T}(\Sigma_G) \to \mathbb{N}$ is given by

$$\begin{array}{rcl} W(x) & \triangleq & 0 \\ W(f(P_1, \ldots, P_l)) & \triangleq & w(f) + W(P_1) + \cdots + W(P_l). \end{array}$$

For example, the GSOS system in Fig. 1 is linear and syntactically well-founded. In fact, it is sufficient to assign weight 1 to the action prefixing operations and weight 0 to all the other operations. Syntactic well-foundedness is decidable over GSOS systems (cf. [2, Thm. 6.8]), and, for linear GSOS systems, it is sufficient to guarantee that terms are semantically well-founded in the sense of [2].

Definition 4.3 (Compact GSOS Systems) *A GSOS system is said to be* compact *iff it is linear and syntactically well-founded.*

4.2 Full Abstraction for Compact GSOS Systems

Let G be a compact GSOS language. We shall now give a way of defining, for each Σ_G-context $C[\vec{x}]$, a function $\mathbf{C}_{\mathsf{ST}}$ over $\mathsf{ST}(\mathsf{Act})$ of the appropriate arity. The definition of $\mathbf{C}_{\mathsf{ST}}$ will be given using the rules in R_G as a guideline. First of all, note that it is sufficient to define semantic operations $\mathbf{f}_{\mathsf{ST}}$ for each $f \in \Sigma_G$, as derived semantic operations can then be obtained by function composition. The definition of the functions $\mathbf{f}_{\mathsf{ST}}$ is given by the inductive construction in Def. 4.4. Intuitively, the inductive construction of the synchronization tree $\mathbf{f}_{\mathsf{ST}}(\vec{t})$ given below is well-founded because, by the compactness of G, whenever the premises of a rule of the form (6) can be met by $\vec{t}$, then either the weight of $C[\vec{x}, \vec{y}]$ is strictly smaller than that of f, or the weight of $C[\vec{x}, \vec{y}]$ is the same as that of f, and the sum of the sizes of the arguments of $\mathbf{C}_{\mathsf{ST}}$ has decreased.

Definition 4.4 *Let $G = (\Sigma_G, R_G)$ be a compact GSOS system, and let f be an l-ary operation in Σ_G. We define the operation $\mathbf{f}_{\mathsf{ST}} : \mathsf{ST}(\mathsf{Act})^l \to \mathsf{ST}(\mathsf{Act})$ inductively by stipulating that, for every $t_1, \ldots, t_l \in \mathsf{ST}(\mathsf{Act})$:*

- $\perp \in \mathbf{f}_{\mathsf{ST}}(t_1, \ldots, t_l)$ *iff* $f = \Omega$ *or there is an argument* i *for* f *such that* f *tests its* i*-th argument and* $\perp \in t_i$*;*

- $\langle c, t\rangle \in \mathbf{f}_{\mathsf{ST}}(t_1, \ldots, t_l)$ *iff there exist a rule for* f *of the form (6) and a vector of trees* $\vec{u} = u_{11} \ldots u_{1m_1} \ldots u_{l1} \ldots u_{lm_l}$ *such that:*

  1. *for all* $1 \leq i \leq l$, $1 \leq j \leq m_i$, $\langle a_{ij}, u_{ij}\rangle \in t_i$,
  2. *for all* $1 \leq i \leq l$, $1 \leq k \leq n_i$, $\perp \notin t_i$ *and, for no* $u \in \mathsf{ST}(\mathsf{Act})$, $\langle b_{ik}, u\rangle \in t_i$, *and*
  3. $\mathbf{C}_{\mathsf{ST}}[\vec{t}, \vec{u}] = t$, *where* $\mathbf{C}_{\mathsf{ST}}$ *denotes the derived semantic operation associated with the* Σ_G*-context* $C[\vec{x}, \vec{y}]$*. If* $C[\vec{x}]$ *is a variable* x_i*, then* $\mathbf{C}_{\mathsf{ST}}[\vec{t}] = t_i$.

When applied to the language $\mathcal{L}$, the construction in Def. 4.4 produces the following functions:

- $\delta_{\mathsf{ST}} = \emptyset$,
- $\Omega_{\mathsf{ST}} = \{\perp\}$,
- for every $t \in \mathsf{ST}(\mathsf{Act})$, $\mathbf{a}_{\mathsf{ST}}(t) = \{\langle a, t\rangle\}$,
- for every $t_1, t_2 \in \mathsf{ST}(\mathsf{Act})$, $t_1 +_{\mathsf{ST}} t_2 = t_1 \cup t_2$,
- for every $t_1, t_2 \in \mathsf{ST}(\mathsf{Act})$, $t_1 \|_{\mathsf{ST}} t_2$ is given by:

  1. $\perp \in t_1 \|_{\mathsf{ST}} t_2$ iff $\perp \in t_1$ or $\perp \in t_2$;
  2. $\langle c, t\rangle \in t_1 \|_{\mathsf{ST}} t_2$ iff one of the following holds:
     (a) there exists $\langle c, t_1'\rangle \in t_1$ such that $t = t_1' \|_{\mathsf{ST}} t_2$, or
     (b) there exists $\langle c, t_2'\rangle \in t_2$ such that $t = t_1 \|_{\mathsf{ST}} t_2'$, or
     (c) there exist $\langle a, t_1'\rangle \in t_1$ and $\langle b, t_2'\rangle \in t_2$ such that $c = \gamma(a, b)$ and $t = t_1' \|_{\mathsf{ST}} t_2'$.

- for every $t \in \mathsf{ST}(\mathsf{Act})$, $\theta_{\mathsf{ST}}(t)$ is given by:

  1. $\perp \in \theta_{\mathsf{ST}}(t)$ iff $\perp \in t$,
  2. $\langle c, t_1\rangle \in \theta_{\mathsf{ST}}(t)$ iff there exists $\langle c, t'\rangle \in t$ such that:
     (a) either c is maximal in $(\mathsf{Act}, >)$ and $\theta_{\mathsf{ST}}(t') = t_1$,
     (b) or c is not maximal in $(\mathsf{Act}, >)$, for no action $b > c$ and t_2 $\langle b, t_2\rangle \in d$, $\perp \notin t$, and $\theta_{\mathsf{ST}}(t') = t_1$.

We shall now prove that the definition of the operations given in Def. 4.4 endows the preorder of synchronization trees $\mathsf{ST}(\mathsf{Act})$ with a Σ_G-preorder structure in the sense of [10]. To this end, it is sufficient to prove that each operation $\mathbf{f}_{\mathsf{ST}}$ is monotonic with respect to the preorder $\sqsubseteq$.

Theorem 4.5 (Monotonicity) *Let* G *be a compact GSOS system, and* $f \in \Sigma_G$*. Then the function* $\mathbf{f}_{\mathsf{ST}}$ *given by the construction in Def. 4.4 is monotonic with respect to* $\sqsubseteq$*.*

Because of the above result, the construction of Def. 4.4 allows us to define, for each operation symbol f in the signature of a compact GSOS system, a monotonic function $\mathbf{f}_{\mathsf{ST}}$ over $\mathsf{ST}(\mathsf{Act})$ of the appropriate arity. This is exactly what is needed to endow the preorder $(\mathsf{ST}(\mathsf{Act}), \sqsubseteq)$ with the structure of a Σ_G-preorder in the sense of [10]. Using Eqn. 4, we then automatically obtain a Σ_G-poset structure on the poset of compact elements of $\mathcal{D}$. The unique meaning maps from $\mathrm{T}(\Sigma_G)$ to the resulting algebraic structures, denoted by $\mathsf{ST}[\![\cdot]\!]$ and $\mathcal{K}(\mathcal{D})[\![\cdot]\!]$, are related thus:

Lemma 4.6 *For every $P \in \mathrm{T}(\Sigma_G)$, $(\mathsf{ST}[\![P]\!])^c = \mathcal{K}(\mathcal{D})[\![P]\!]$.*

We now proceed to show that the denotational semantics $\mathcal{K}(\mathcal{D})[\![\cdot]\!]$ for recursion-free terms in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is fully abstract with respect to the bisimulation preorder.

Theorem 4.7 *Let G be a compact GSOS system. Then, for all $P, Q \in \mathrm{T}(\Sigma_G)$, $P \lesssim Q$ iff $\mathcal{K}(\mathcal{D})[\![P]\!] \sqsubseteq \mathcal{K}(\mathcal{D})[\![Q]\!]$.*

Our aim in the remainder of this section will be to extend the above full abstraction result to the whole of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. In the process of proving this result, we shall establish very general results for compact GSOS languages that have independent interest. First of all, in order to define an interpretation of programs in $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ as elements of $\mathcal{D}$, we need to define a continuous Σ_G-algebra structure on $\mathcal{D}$. As $(\mathcal{D}, \sqsubseteq_{\mathcal{D}})$ is, up to isomorphism, the unique algebraic cpo with $(\mathcal{K}(\mathcal{D}), \sqsubseteq)$ as poset of compact elements, this is easily done by using Eqn. (5) to define a continuous function $\mathbf{f}_{\mathcal{D}}$ for each $f \in \Sigma_G$. By the general theory of algebraic semantics (see, e.g., [8]), we then have that, for all $P, Q \in \mathrm{T}(\Sigma_G)$,

$$\mathcal{D}[\![P]\!] \sqsubseteq_{\mathcal{D}} \mathcal{D}[\![Q]\!] \;\Leftrightarrow\; \mathcal{K}(\mathcal{D})[\![P]\!] \sqsubseteq \mathcal{K}(\mathcal{D})[\![Q]\!] \ . \tag{10}$$

In view of Thm. 4.7, our desired full abstraction result will follow if we prove that the behavioural preorder $\lesssim_\omega$ is algebraic. This is because, from general properties of denotational interpretations, the relation $\sqsubseteq_{\mathcal{D}}$ is algebraic, and two algebraic relations that coincide over $\mathrm{T}(\Sigma_G)$ do, in fact, coincide over the whole of $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$. The key to the proof of the algebraicity of $\lesssim_\omega$ is the following general theorem providing a partial completeness result for $\lesssim$ in the sense of Hennessy [7] for arbitrary compact GSOS systems.

Theorem 4.8 (Partial Completeness) *Let G be a compact GSOS system. Then there exist a compact GSOS system G' and a set of $\Sigma_{G'}$-inequations T such that:*

- *G' disjointly extends G and $\mathsf{FINTREE}_\Omega$ in the sense of [2], and*
- *for all $P \in \mathrm{T}(G')$, $Q \in \mathsf{CREC}(\Sigma_{G'}, \mathsf{Var})$, $P \lesssim Q$ iff $T \cup \{\text{Eqn. } 8\} \vdash P \leq Q$.*

The proof of this theorem is rather involved. It relies on an algorithm for generating the compact GSOS system G' and the inequational theory T over it following the lines of [2]. Apart from its intrinsic interest, the main consequence of Thm. 4.8 is the following key result, whose proof uses properties of initial continuous algebras in inequational varieties, that essentially states that finite trees are compact elements with respect to the preorder $\lesssim$.

Theorem 4.9 *Suppose that G is a compact GSOS system. Assume that $t \in \mathsf{ST}(\mathsf{Act})$ and $P \in \mathsf{CREC}(\Sigma_G, \mathsf{Var})$. Then $t \lesssim_{Grec} P$ iff there exists a finite approximation P^n of P such that $t \lesssim_{Grec} P^n$.*

The above result, in conjunction with Propn. 3.6, allows us to prove that $\lesssim_\omega$ is indeed algebraic.

Theorem 4.10 *Let G be a compact GSOS system. Then the relation $\lesssim_\omega$ over $\mathsf{lts}(Grec)$ is algebraic.*

In light of the above results, we can now show that, for any compact GSOS system G, the denotational semantics for $\mathsf{CREC}(\Sigma_G, \mathsf{PVar})$ is fully abstract with respect to $\lesssim_\omega$.

Theorem 4.11 *Let G be a compact GSOS system. Then, for all $P, Q \in \mathsf{CREC}(\Sigma_G, \mathsf{Var})$, $P \lesssim_\omega Q$ iff $\mathcal{D}[\![P]\!] \sqsubseteq_{\mathcal{D}} \mathcal{D}[\![Q]\!]$.*

References

[1] S. Abramsky, *A domain equation for bisimulation*, Information and Computation, 92 (1991), pp. 161–218.

[2] L. Aceto, B. Bloom, and F. Vaandrager, *Turning SOS rules into equations*, Information and Computation, 111 (1994), pp. 1–52.

[3] L. Aceto and A. Ingólfsdóttir, *CPO models for GSOS languages — Part I: Compact GSOS languages*, Research Report RS-94-40, BRICS (Basic Research in Computer Science, Centre of the Danish National Research Foundation), Department of Mathematics and Computer Science, Aalborg University, December 1994. An extended abstract of the paper will appear in: *Proceedings of CAAP '95*, LNCS, 1995.

[4] J. Baeten, J. Bergstra, and J. Klop, *Syntax and defining equations for an interrupt mechanism in process algebra*, Fundamenta Informaticae, IX (1986), pp. 127–168.

[5] J. Baeten and W. Weijland, *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[6] B. Bloom, S. Istrail, and A. Meyer, *Bisimulation can't be traced: preliminary report*, in Conference Record of the Fifteenth Annual ACM Symposium on Principles of Programming Languages, 1988, pp. 229–239. Full version available as Technical Report 90-1150, Department of Computer Science, Cornell University, Ithaca, New York, August 1990. To appear in the *Journal of the ACM*.

[7] M. Hennessy, *A term model for synchronous processes*, Information and Control, 51 (1981), pp. 58–75.

[8] ——, *Algebraic Theory of Processes*, MIT Press, Cambridge, Massachusetts, 1988.

[9] M. Hennessy and G. Plotkin, *A term model for CCS*, in 9^{th} Symposium on Mathematical Foundations of Computer Science, P. Dembiński, ed., vol. 88 of Lecture Notes in Computer Science, Springer-Verlag, 1980, pp. 261–274.

[10] A. Ingólfsdóttir, *Semantic Models for Communicating Process with Value-Passing*, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, June 1994. Computer Science Report 8/94. Also available as Report R–94–2044, Department of Mathematics and Computer Science, Aalborg University.

[11] R. Milner, *A modal characterisation of observable machine behaviour*, in Proceedings CAAP 81, G. Astesiano and C. Bohm, eds., vol. 112 of Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 25–34.

[12] ——, *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, 1989.

[13] G. Plotkin, *Lecture notes in domain theory*, 1981. University of Edinburgh.

[14] ——, *A structural approach to operational semantics*, Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[15] D. Walker, *Bisimulation and divergence*, Information and Computation, 85 (1990), pp. 202–241.