SOS Specifications for Compositional ST-Semantics *

J. V. Echagüe† S. Pinchinat ⋆

† Instituto de Computación, Facultad de Ingenieria,
Universidad de la República, Montevideo, Uruguay.
⋆ Institut de Recherche en Informatique et en Systèmes aléatoires,
Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France.

Abstract

Syntactic formats of SOS specifications as a criterion to obtain compositional semantics have widely been investigated, almost always in the interleaving setting.

In this paper, we study the case of truly concurrent semantics. We show how to derive Asynchronous Transition Systems with invisible actions from SOS specifications, and we exhibit simple syntactic conditions over the specifications to get compositionality for (rooted) ST-semantics. Classical process description languages like CCS fit the conditions.

1 Introduction

SOS specifications techniques [Plo81] is a general method to define process description languages, e.g. CCS [Mil89]. It naturally induces operational semantics for the language. Since this last decade, we observe considerable progresses in this field. In particular, strong results relate syntactic aspects of the specifications, as rules formats, with semantic properties of the underlying models [Sim85], [BIM88], [GV89] (see also [GV92]), [Gro89] (see also [Gro93]), [Gla93], [Blo95]. Results of this kind are powerful since using a particular format for defining your favorite language ensures semantic properties. Basically, all the results show that we get compositional semantics for free.

All the previously mentioned works only apply to non-deterministic sequential aspects of concurrent programs. This not surprising since SOS specifications naturally define Transition Systems. However, compositionality results have been established for non-interleaving models, but they all focus on a particular language. Since then [BD92] went ahead by giving a way to "read" true concurrent semantics in any specifications provided they fulfill syntactic conditions. [BD92] showed how to derive a trace automata [Sta89] from an SOS specification (Trace automata are Transition Systems enriched with an independence relation over transitions). But no semantic equivalence was considered.

Among the truly concurrent equivalence, one of the most relevant is *ST-bisimulation* equivalence [GV87] : it does not reduce actions to atomic ones, and by expressing an elementary form of duration (each action has a *beginning* and an *ending*), it constitutes the right notion of behavior compatible with the *refinement of actions*, see [Gla90] but also [Vog91]. The main notion in ST-bisimulation is the one of *ST-states* : an ST-state represents the state of the process when concurrent activities are executing.

In this paper, we address the following question : is it possible to determine syntactic criterions over SOS specifications in order to get a compositional ST-bisimulation semantics?

We partially answered the question in [EHP95]. Strongly inspired from [BD92], we have shown how Asynchronous Transition Systems (of [Shi85] and [Bed87]) underlay SOS specifications for languages

*Work partially supported by a French Government's Scholarship, by the "Plan de Recursos Humanos BID-CONICYT", PEDECIBA - Computer Science (Uruguay), and the project "Formal models for Concurrency" of the French Government ECOS program. E-mails :echague@fing.edu.uy, Sophie.Pinchinat@irisa.fr

with no invisible actions. Asynchronous Transition Systems offer a very easy way to define ST-states as well as ST-bisimulation. The main contribution of [EHP95] is the determination of a simple syntactic condition over the axioms of the specification in order to guarantee semantic compositionality, not only for ST-bisimulation, but also for other truly concurrent bisimulations.

In this paper, we fully answer the question. Namely, we consider SOS specifications containing invisible actions. In this framework, the approach of [BD92] has to be strictly extended to deal with rule concerning non observable actions. in order to built an adequate Asynchronous Transition System. The conditions over SOS specifications regarding weak ST-bisimulation has to be closely related to the conditions proposed in [Blo95] for weak bisimulation and branching bisimulation for non-interleaving models.We show that specifications describing so-called *tame* combinators have compositional weak ST-bisimulation semantics.
Existing operators such as the deterministic choice of CCS (and many others) do not behave properly with weak bisimulations, but does with rooted ones. To cope with these combinators, we exhibit syntactic conditions (delivering *tamable* combinators) to get compositionality for rooted weak ST-bisimulation.

The paper is organised as follows : Section 2 introduces the semantic framework, namely Asynchronous Transition Systems, ST-states, ST-bisimulation and its rooted variant. Section 3 is devoted to the construction of the Asynchronous Transition System, illustrated with an extension of CCS language. Finally in Section 4, we show the congruence theorem for weak ST-bisimulation (resp. rooted weak ST-bisimulation) with respect to all tame (resp. tamable) combinators. We end up by concluding remarks and future works.

2 Truly Concurrent Semantic Models

2.1 Asynchronous Transition Systems

We assume given a set of action names $Act = \{\alpha, \beta, \gamma \ldots\}$ and a new action name $\tau \notin Act$. We denote by $Act_\tau = Act \bigcup \{\tau\}$.

Definition 1 *A* Labelled Asynchronous Transition System (ATS for short) over *Act is a structure* $G = \langle S, E, I, \longrightarrow, l\rangle$ *where*

- $S = \{s, u, s_1, s_2 \ldots\}$ *is a set of* states*.*
- $E = \{a, b, e \ldots\}$ *is a set of* events*.*
- $I \subseteq E \times E$ *is an irreflexive symmetric relation of* independence *between events. We shall write* aIb *instead of* $(a, b) \in I$*.*
- $\longrightarrow \subseteq S \times E \times S$ *is the* transition relation *; let us we write* $s \xrightarrow{e} s'$ *instead of* $(s, e, s') \in \longrightarrow$ *and* $s \xrightarrow{e}$ *if* $s \xrightarrow{e} s'$ *for some* $s' \in S$*. The transition relation has to fulfill the following three properties :*

  Determinism *If* $s \xrightarrow{e} s_1$ *and* $s \xrightarrow{e} s_2$ *then* $s_1 = s_2$*.*

  Forward stability *If* aIb *and* $s \xrightarrow{a} s_1$ *and* $s \xrightarrow{b} s_2$ *then* $\exists u, s_1 \xrightarrow{b} u \wedge s_2 \xrightarrow{a} u$*.*

  Commutativity *If* aIb *and* $s \xrightarrow{a} s_1 \xrightarrow{b} u$ *then* $\exists s_2, s \xrightarrow{b} s_2 \xrightarrow{a} u$*.*

- $l : E \mapsto Act$ *is a* labelling function *(which associates to e its visible part).*

We write $\mathbf{ATS} = \{G, G_1, G_2 \ldots\}$ *for the class of Labelled Asynchronous Transition Systems over Act.*

By means of resources sharing, events have to be understood as tasks. Two tasks are independent if they do not share the same resources. Now the three properties required in Definition 1 can easily be understood. As events are tasks, it is then very natural to ask for a deterministic model. The forward stability property expresses that if two independent tasks, namely a and b, can be performed, the execution of one them would not disturb the possibility of executing the other one, since they do not share the same resource. The commutativity property tells that if a task b can be performed after the execution of a task a independent with b, then the resource needed for b were already avaible before starting a otherwise a would have produced the resource needed for b, which is not what we intend by being independent. So b could have been performed before a and for the same reasons, a can still be executed after b. Moreover,

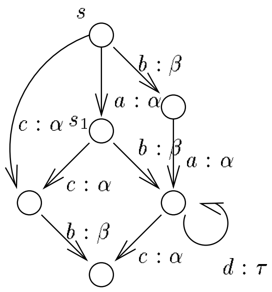

Figure 1: An example of an ATS, with aIb

in both case of forward stability and commutativity, the state of the system after the execution of the two tasks (in any order) is unique.

In graphical representations, the independence relation is represented aside. Transitions between states are drawn using labelled arrows. A label of the form $a:\alpha$ means that the transition gives rise to event a with $l(a)=\alpha$. Figure 1 is an example of an ATS with 6 states where $E=\{a,b,c,d\}$ and aIb.

With models like ATS', states representing ongoing activities of the system can be derived. Consider for example the leftmost ATS of Figure 2. The dashed area, or surface, precisely denotes simultaneous activity of independent tasks a and b. This can also be applied to more than two tasks, leading to higher dimensional surfaces. With three tasks, one would draw a cube, with 4 or more of them, one would not draw anything. At some point in the paper, we use this geometrical intuition for ST-states. The definition below is strongly inspired from [Ech93].

Definition 2 (ST-states) *Let $G=\langle S,E,I,\rightarrow,l\rangle\in$ **ATS**. An* ST-state *of G is a couple $(s,\{e_j\}_{j\in J})\in S\times\mathcal{P}(E)$, where*

1. *for all $j\in J$, $l(e_j)\neq\tau$,*
2. *for all $j\in J$, $s\xrightarrow{e_j}$,*
3. *for all $k,l\in J$, $k\neq l$, e_kIe_l. Then, all the e_j's are distinct.*

From now on, $(s,\{e_j\}_{j\in J})$ will be written $s+\sum_J e_j$ to emphasize that we not pay attention to any order between events e_j. Writing $s+\sum_J e_j$ will implicitly guarantee Properties 1, 2 *and* 3 *of Definition 2. A ST-state of the form $s+\sum_J e_j$ is said to* start from state s *; s is also called the* source *of the ST-state. We call the* dimension *of the ST-state $s+\sum_J e_j$ the cardinal of set J.*

We write $ST(G)$ for the set of ST-states of G.

Example 1 *In the leftmost ATS of Figure 2, $s+a+b$ is an ST-state with dimension* 2, *whereas s_1+b+c is not (defined) as b and c are not independent.*

Notice that any state s is a particular ST-state with dimension 0, starting from s where no event is executing. A transition $s\xrightarrow{e}s'$ is also an ST-state, namely $s+e$, with dimension 1. In the following, *primitive ST-states* are those with dimension 0 or 1.

In ATS', only primitive ST-states are represented. However, thanks to the independence, we get a way to combine primitive ST-states in order to get higher dimensional ones.

2.2 ST-bisimulation and rooted ST-bisimulation

The kind of behavior underlying ST-bisimulation equivalence is based on the notion of ongoing activity of processes : the existence of an ST-bisimulation equivalence between two processes, i.e. ATS in our framework, is simply defined as a classical bisimulation but not only between primitive ST-states (of dimension 0 and 1), as classical bisimulations do, but also between any possible ST-states of the system. Additionally, the ST-bisimulation relation has to be enriched : instead of defining couples in the relation, we consider triples where the first two components are (equivalent) ST-states and the third component is a mapping which a link between ongoing actions of these ST-states, as it is necessary in the presence of auto-concurrency, i.e. concurrent events with same labels, to relate unambiguously every ending phase of each action with its corresponding beginning. From a geometrical point of view, the mapping is defined between the sets of edges of each object, which can be consistently extended as the ST-states grow.

Originally, ST-bisimulation was proposed over Petri nets [GV87] and over Event Structures [Gla90]. Its definition in ATS', as proposed here, definitely improves its simplicity, mostly because there is no need here, comparing to previous approaches, to take into account the past of the computation.

Definition 3 (ST-bisimulation)
Let $G = \langle S, E, I, \rightarrow, l\rangle$ be an ATS.
A relation $\mathcal{R} \subseteq ST(G) \times ST(G) \times \mathcal{P}(E \times E)$ is an ST-bisimulation of G, *iff*

1. *if $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$ then*
   (a) *$h : \{a_j\}_{j\in J} \rightarrow \{b_k\}_{k\in K}$ has to be a* label-preserving *bijection, i.e. $l(h(a)) = l(a)$.*
   (b) *$(r + \sum_K b_k, s + \sum_J a_j, h^{-1}) \in \mathcal{R}$, where h^{-1} is the inverse function of h.*
2. *for all $(s + \sum_J a_j, r + \sum_K b_k, h) \in \mathcal{R}$,*
   (a) *if $s + \sum_J a_j + a \in ST(()G)$ then exists a path with 0 or more τ, $r \stackrel{e^1:\tau}{\rightarrow}\stackrel{e^2:\tau}{\rightarrow} \cdots \stackrel{e^m:\tau}{\rightarrow} r'$, such that $\forall i = 1 \ldots m, j \in J.\ (e^i I b_j)$, $r' + \sum_K b_k + b \in ST(()G)$ and $(s + \sum_J a_j + a, r' + \sum_K b_k + b, h \cup \{(a, b)\}) \in \mathcal{R}$,*
   (b) *for all $k \in J$, let $s', r' \in S$ respectively be s.t. are $s \stackrel{a_k}{\rightarrow} s'$ and $r \stackrel{h(a_k)}{\rightarrow} r'$, then it is the case that $(s' + \sum_{J\setminus\{k\}} a_j, r' + \sum_{K\setminus\{h(a)\}} b_k, h|_{\{a_j\}_{j\in J}\setminus\{a_k\}}) \in \mathcal{R}$,*
   (c) *if $s \stackrel{e:\tau}{\rightarrow} s'$, and $\forall j \in J.\ (e I a_j)$ then exists a path with 0 or more τ, $r \stackrel{e^1:\tau}{\rightarrow}\stackrel{e^2:\tau}{\rightarrow} \cdots \stackrel{e^m:\tau}{\rightarrow} r'$, such that $\forall i = 1 \ldots m, j \in J.\ (e^i I b_j)$ and $(s' + \sum_J a_j, r' + \sum_K b_k, h) \in \mathcal{R}$,*

For all $s, r \in S$, we write $\mathcal{R} : s \equiv_{ST} r$ whenever $\mathcal{R}$ is an ST-bisimulation of G and $(s, r, \emptyset) \in \mathcal{R}$, and we extend this notation to n-tuples $(s_1, ..., s_n)$ and $(r_1, ..., r_n)$ of S^n by writing $\mathcal{R} : (s_1, ..., s_n) \equiv_{ST} (r_1, ..., r_n)$ (or $\vec{s} \equiv_{ST} \vec{r}$) whenever $\mathcal{R} : s_j \equiv_{ST} r_j$ for all $j = 1, ..., n$. We write $s \equiv_{ST} r$ whenever there exists $\mathcal{R}$ s.t. $\mathcal{R} : s \equiv_{ST} r$.

The definition above has to be red as follows : Clauses 1 is clear. Clauses 2a is the classical transfer property of bisimulation for growing ST-states, or in other words when the beginning of an action is observed. Similarly Clause 2b correspond to an ending phase.

Note that when no concurrency is considered weak ST-bisimulation corresponds to *delay bisimulation* [Wei89], [Wal88].

Definition 4 (rooted ST-bisimulation) *We say that two states s and r are rooted ST-bisimilar ($s \equiv_{rST} r$) iff for all $s \stackrel{a:\mu}{\rightarrow} s'$ there exists a path $r \stackrel{e^1:\tau}{\rightarrow} \stackrel{e^2:\tau}{\rightarrow} \cdots \stackrel{e^m:\tau}{\rightarrow} \stackrel{b:\mu}{\rightarrow} r'$, such that $s' \equiv_{ST} r'$, and vice versa.*

Rooted ST-bisimulation should be called "interleaving" rooted ST-bisimulation, because The first move consists of an interleaving rooted delta step before considering classical ST moves.

Consider Examples given in 2.

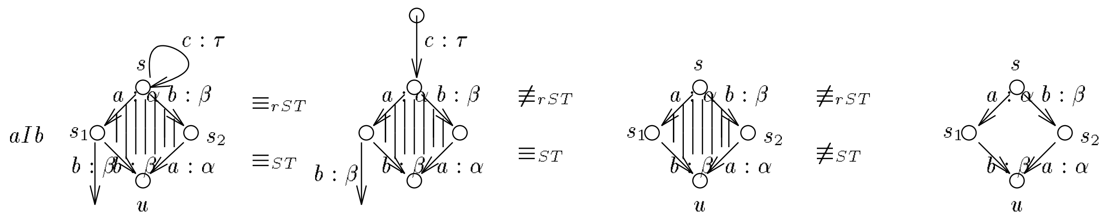

Figure 2: Example for $\equiv_{ST}$ and $\equiv_{rST}$

Remark 1 *Classically, note that if $\mathcal{R}_1$ and $\mathcal{R}_2$ are two (rooted) ST-bisimulations of G so is $\mathcal{R}_1 \cup \mathcal{R}_2$, and consequently, there exists a greatest (rooted) ST-bisimulation of G. Moreover, this (rooted) ST-bisimulation of G, when projected onto its first two components is an equivalence relation over S, the set of states.*

3 SOS-specifications of Asynchronous Transition Systems

In the sequel, we assume given a set $V = \{u, v, u_1, v_1, ...\}$ of *variables*.

Definition 5 *Let $\Sigma = (F, ari)$ be a single sorted signature, where F is a set of* function names *(or* combinators*) different from V and $ari : F \rightarrow N$ gives the arity of the combinators.*

A term on the signature Σ, or simply a Σ-term, *is a labelled tree over Σ, i.e. a partial function $t : (N_+)^* \mapsto \Sigma$. We write T_Σ for the set of* closed *terms over Σ, that is Σ-terms containing no variables.*

In the following, if $\Sigma = (F, ari)$ is a signature, we write simply "$f \in \Sigma^n$" instead of "$f \in F$ and $ari(f) = n$", and "$f \in \Sigma$" if $f \in \Sigma^n$ for some natural n.

3.1 Basic SOS specifications

Definition 6 *Given a set of actions $Act = \{\alpha, \beta \ldots\}$ and $Act_\tau = Act \cup \{\tau\}$, with typical elements $\mu, \eta \ldots$. A* Basic Structural Operational Semantics specification *(or specification for short) is a structure $P = (\Sigma, R)$ where $\Sigma = (F, ari)$ is a signature and R is a set of rules s.t. for all $r \in R$, r has the form*

$$\frac{u_{i_1} \stackrel{\mu_{i_1}}{\rightarrow} v_{i_1} \ \ldots \ u_{i_m} \stackrel{\mu_{i_m}}{\rightarrow} v_{i_m}}{f(u_1, u_2 \ldots, u_n) \stackrel{\mu}{\rightarrow} g(v_1, v_2 \ldots, v_n)} \tag{1}$$

where u_i, v_i are distinct variables, $u_i = v_i$ iff $i \notin \{i_1, \ldots i_m\}$, $\mu, \mu_i \in Act_\tau$, $f, g \in \Sigma^n$.

- *Expressions $u_{i_j} \stackrel{\mu_i}{\rightarrow} v_{i_j}$ are the* premises *of rule r and $f(u_1 \ldots u_n) \stackrel{\mu}{\rightarrow} g(v_1 \ldots v_n)$ its* conclusion.
- *Combinator f in the conclusion of the rule will be called the* source combinator *of r, and g the* target combinator.

  Notice that as long as two combinators occur in the same conclusion of a basic SOS rule, then $ari(f) = ari(g)$.
- *A rule r is an* axiom *if it has no premises.*

Definition 7 (Basic definitions and properties in SOS specifications) *Let P be a basic SOS specification.*

- *An axiom is said to be* reflexive *if the source and the target combinators are identical.*
  A specification P is irreflexive *whenever none of its axioms are reflexive.*
- *Given a specification P, the* patience rule for position i in f *is the (possible derived) rule*

$$\frac{u_i \stackrel{\tau}{\rightarrow} v_i}{f(u_1, \ldots u_i, \ldots u_n) \stackrel{\tau}{\rightarrow} f(u_1, \ldots v_i, \ldots u_n)}$$

- *A combinator f is* cool in P *(n.b. we follow the Bloom's terminology) whenever*
  1. *no rule for f contains premises mentioning τ's, except the patience rules required by the previous clause.*
  2. *for all rule r with source combinator f, and each premises $u_i \stackrel{\mu}{\rightarrow} v_i$ of r, there exists in P a patience rule for position i in f.*

3.2 An example

Although the format of the rules seems very restrictive, SOS-specifications are expressive enough to encode for instance all the CCS combinators as noticed by [BD92] : we extend the syntax of CCS with *id* (the identity combinator), π_1 and π_2 (projections). Example 2 shows a subset of CCS rules (without the rules for restriction and renaming).
We also introduce another combinator, here called *angelic choice*, closely related to [BHR84]. This combinator will be used to illustrate particular results of the paper. Intuitively, the angelic choice chooses between two processes, like combinator + of CCS does, but only when a visible action is performed by one of two components.
Classically, for combinators like $+$, $|$, $\oplus$ we use infixed notations, whereas for less classical ones we use prefixed notations, e.g. π_1, π_2,...

Example 2 *Additionally to the set of combinators occurring in the rules below, we add a combinator with arity* 0 *classically named nil to denote the process that does nothing.*

$$(r_\mu)\ \frac{}{\mu.p \xrightarrow{\mu} id(p)} \qquad (r_{id})\ \frac{p \xrightarrow{\mu} p}{id(p) \xrightarrow{\mu} id(p)}$$

$$(r_{+_1})\ \frac{p \xrightarrow{\mu} p'}{p + q \xrightarrow{\mu} \pi_1(p', q)} \qquad (r_{+_2})\ \frac{q \xrightarrow{\mu} q'}{p + q \xrightarrow{\mu} \pi_2(p, q')}$$

$$(r_{\pi_1})\ \frac{p \xrightarrow{\mu} p'}{\pi_1(p, q) \xrightarrow{\mu} \pi_1(p', q)} \qquad (r_{\pi_2})\ \frac{q \xrightarrow{\mu} q'}{\pi_2(p, q) \xrightarrow{\mu} \pi_2(p, q')}$$

$$(r_{|_1})\ \frac{p \xrightarrow{\mu} p'}{p \mid q \xrightarrow{\mu} p' \mid q} \qquad (r_{|})\ \frac{p \xrightarrow{\alpha} p' \quad q \xrightarrow{\bar{\alpha}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'} \qquad (r_{|_2})\ \frac{q \xrightarrow{\mu} q'}{p \mid q \xrightarrow{\mu} p \mid q'}$$

$$(r_{\oplus^1_\tau})\ \frac{p \xrightarrow{\tau} p'}{p \oplus q \xrightarrow{\tau} p' \oplus q} \qquad (r_{\oplus^2_\tau})\ \frac{q \xrightarrow{\tau} q'}{p \oplus q \xrightarrow{\tau} p \oplus q'} \qquad (r_{\oplus_1})\ \frac{p \xrightarrow{\alpha} p'}{p \oplus q \xrightarrow{\alpha} p'} \qquad (r_{\oplus_2})\ \frac{q \xrightarrow{\alpha} q'}{p \oplus q \xrightarrow{\alpha} q'}$$

The specification above provides us with an irreflexive basic SOS- specification. Language CCS is defined when we remove the rules for the angelic choice. Notice that combinator + is not cool since there is no patience rule for the first argument, nor for the second one. This is not the case for $\oplus$ thanks to rules $(r_{\oplus^1_\tau})$and $(r_{\oplus^2_\tau})$. In rules (r_id), (r_{π_1}), (r_{π_2}), $(r_{|_1}$ and $(r_{|_2})$ when instantiating μ as τ gives the the patience rules for the corresponding combinator.
We shall see in Section 4 that coolness hypothesis is necessary regarding congruence properties for ST-bisimulation.

3.3 Coding proofs in transitions

For the construction of an ATS derived from the specification, we follow [BD92] : the idea is to keep track in the label in the rules' conclusion of informations about that rule. This information (called *schematic rule* in [BD92]) enables us to define event-labeled transitions : the events are simply trees/terms of schematic rules. These terms (called *schematic proofs* in [BD92]) are derived from the proof trees of the transitions in which the rules are replaced by their schematic rule.
Transitions have the form $p \xrightarrow{a:\mu} p'$ where a is a schematic proof and $\mu \in Act_\tau$.

In this paper, we use a fairly handful convention to write schematic rules. Before all, and for technical reasons, we introduce a new symbol ϵ to denote idleness of processes. ϵ will be the label of idle step. Now, if r is a rule of the form

$$\frac{u_{i_1} \xrightarrow{\mu_{i_1}} v_{i_1} \ \ldots \ u_{i_m} \xrightarrow{\mu_{i_m}} v_{i_m}}{f(u_1, u_2 \ldots, u_n) \xrightarrow{\mu} g(v_1, v_2 \ldots, v_n)}$$

its schematic rule will be written $g_\mu^{\eta_1\ \eta_2 \cdots \eta_n}$ where μ is precisely the label of the conclusion, and η_i is the label of the premise $x_i \xrightarrow{\eta_i} y_i$ if it exists, ϵ otherwise.
However, schematic rules of patience rules as well as patience rules obtained from general rules (like the patience rule we obtain from $(r_{|_1})$ of CCS when $\mu = \tau$) are coded differently : consider a patience rule at position i in f, then its schematic rule will be written $pat^{ari(f),i}$. Notice that the target combinator (equal to f as well) does not appear.
In the following, we use Greek letters ρ, σ, ... to denote schematic rules.

From this point in the paper, we propose a new presentation of the specification, where the schematic proofs are inductively represented in the rules themselves. The reader has to be persuaded that the specification we obtain is fully equivalent to the original one.

Let r be a rule which is not a patience rule, with schematic rule ρ and of the form

$$\frac{u_{i_1} \stackrel{\mu_{i_1}}{\rightarrow} v_{i_1} \ \ldots \ u_{i_m} \stackrel{\mu_{i_m}}{\rightarrow} v_{i_m}}{f(u_1, u_2 \ldots, u_n) \stackrel{\mu}{\rightarrow} g(v_1, v_2 \ldots, v_n)}$$

will be coded

$$\frac{u_{i_1} \stackrel{a_{i_1}:\mu_{i_1}}{\rightarrow} v_{i_1} \ \ldots \ u_{i_m} \stackrel{a_{i_m}:\mu_{i_m}}{\rightarrow} v_{i_m}}{f(u_1, u_2 \ldots, u_n) \stackrel{\rho(e_1,\ldots,e_n):\mu}{\rightarrow} g(v_1, v_2 \ldots, v_n)}$$

where $e_i = \epsilon$ whenever $i \notin \{i_1, ..., i_m\}$. Notice that in case r is an axiom, $e_i = \epsilon$ for all i.

The following example shows how Example 2 will henceforth be written.

Example 3

$$(r_\mu) \quad \frac{}{\mu.p \stackrel{id^{\epsilon}_{\mu}(\epsilon):\mu}{\rightarrow} id(p)} \qquad (r_{id}) \quad \frac{p \stackrel{a:\alpha}{\rightarrow} p'}{id(p) \stackrel{id^{\alpha}_{\alpha}(a):\alpha}{\rightarrow} id(p')}$$

$$(pat_{id}) \quad \frac{p \stackrel{a:\tau}{\rightarrow} p'}{id(p) \stackrel{pat^{1,1}(a):\tau}{\rightarrow} id(p')}$$

$$(r_{+_1}) \quad \frac{p \stackrel{a:\mu}{\rightarrow} p'}{p + q \stackrel{\pi^{\mu\epsilon}_{1\mu}(a,\epsilon):\mu}{\rightarrow} \pi_1(p', q)} \qquad (r_{+_2}) \quad \frac{q \stackrel{a:\mu}{\rightarrow} q'}{p + q \stackrel{\pi^{\epsilon\mu}_{2\mu}(\epsilon,a):\mu}{\rightarrow} \pi_2(p, q')}$$

$$(r_{\pi_1}) \quad \frac{p \stackrel{a:\alpha}{\rightarrow} p'}{\pi_1(p, q) \stackrel{\pi^{\alpha\epsilon}_{1\alpha}(a,\epsilon):\alpha}{\rightarrow} \pi_1(p', q)} \qquad (r_{\pi_2}) \quad \frac{q \stackrel{a:\alpha}{\rightarrow} q'}{\pi_2(p, q) \stackrel{\pi^{\epsilon\alpha}_{2\alpha}(\epsilon,a):\alpha}{\rightarrow} \pi_2(p, q')}$$

$$(pat_{\pi_1}) \quad \frac{p \stackrel{a:\tau}{\rightarrow} p'}{\pi_1(p, q) \stackrel{pat^{2,1}(a,\epsilon):\tau}{\rightarrow} \pi_1(p', q)} \qquad (pat_{\pi_2}) \quad \frac{q \stackrel{a:\tau}{\rightarrow} q'}{\pi_2(p, q) \stackrel{pat^{2,2}(\epsilon,a):\tau}{\rightarrow} \pi_2(p, q')}$$

$$(r_{|_1}) \quad \frac{p \stackrel{a:\alpha}{\rightarrow} p'}{p \mid q \stackrel{|^{\alpha\epsilon}_{\alpha}(a,\epsilon):\alpha}{\rightarrow} p' \mid q} \qquad (r_{|}) \quad \frac{p \stackrel{a:\alpha}{\rightarrow} p' \ q \stackrel{b:\bar{\alpha}}{\rightarrow} q'}{p \mid q \stackrel{|^{\alpha\bar{\alpha}}_{\tau}(a,b):\tau}{\rightarrow} p' \mid q'} \qquad (r_{|_2}) \quad \frac{q \stackrel{a:\alpha}{\rightarrow} q'}{p \mid q \stackrel{|^{\epsilon\alpha}_{\alpha}(\epsilon,a):\alpha}{\rightarrow} p \mid q'}$$

$$(pat_{|_1}) \quad \frac{p \stackrel{a:\tau}{\rightarrow} p'}{p \mid q \stackrel{pat^{2,1}(a,\epsilon):\tau}{\rightarrow} p' \mid q} \qquad (pat_{|_2}) \quad \frac{q \stackrel{a:\tau}{\rightarrow} q'}{p \mid q \stackrel{pat^{2,2}(\epsilon,a):\tau}{\rightarrow} p \mid q'}$$

$$(pat_{\oplus_1}) \quad \frac{p \stackrel{a:\tau}{\rightarrow} p'}{p \oplus q \stackrel{pat^{2,1}(a,\epsilon):\tau}{\rightarrow} p' \oplus q} \qquad (pat_{\oplus_2}) \quad \frac{q \stackrel{a:\tau}{\rightarrow} q'}{p \oplus q \stackrel{pat^{2,2}(\epsilon,a):\tau}{\rightarrow} p \oplus q'}$$

$$(r_{\oplus_1}) \quad \frac{p \stackrel{a:\alpha}{\rightarrow} p'}{p \oplus q \stackrel{\oplus^{\alpha\epsilon}_{\alpha}(a,\epsilon):\alpha}{\rightarrow} p'} \qquad (r_{\oplus_2}) \quad \frac{q \stackrel{a:\alpha}{\rightarrow} q'}{p \oplus q \stackrel{\oplus^{\epsilon\alpha}_{\alpha}:\alpha}{\rightarrow} q'}$$

Apart from the way we treat patience rules, the coding we propose is a (hopefully) comfortable encoding of the proposal in [BD92]. From now on, we consider SOS specifications with this new coding.

The use of schematic rules makes loose information about the set of rules: we abstract from the source combinator of the rule. Following [BD92], the information can be completed in a directed graph where

- combinators are vertices, and
- arrows $f \stackrel{\rho}{\dashrightarrow} g$ representing each rule with conclusion of the form $f \stackrel{\rho(a_1 \ldots, a_n):\mu}{\rightarrow} g$.

Definition 8 *As in [EHP95] we write $\rho \diamond \sigma$ whenever $\forall f \in \Sigma$*

- $f \stackrel{\rho}{\dashrightarrow} \stackrel{\sigma}{\dashrightarrow} \iff f \stackrel{\sigma}{\dashrightarrow} \stackrel{\rho}{\dashrightarrow}$ *and*

- *if* $f \overset{\rho}{\dashrightarrow} g_1$ *and* $f \overset{\sigma}{\dashrightarrow} g_2$ *then there exist* g_3 *s.t.* $g_1 \overset{\rho}{\dashrightarrow} g_3$ *and* $g_2 \overset{\sigma}{\dashrightarrow} g_3$.

Notice that in the second point of Definition 8, because of point 1 we have either $g_3 = g_1 = g_2$ ((a) equal or (b) not to f), either (c) ρ or σ is a patience rule, say ρ, and it must be the case that $g_1 = f$ and $g_2 = g_3 \neq f$.

Figure 3 shows the situation for CCS combinators $\alpha.$, id, $+$, π_1, π_2, $|$ enriched with combinator $\oplus$ for the angelic choice. Situation (a) happens for rules $|_{\alpha}^{\alpha\epsilon}$ and $|_{\alpha}^{\epsilon\alpha}$ starting from $|$ combinator. Situation (c) is given from combinator $\oplus$ for rules $pat^{1,1}$ and $\oplus_{\alpha}^{\alpha\epsilon}$.

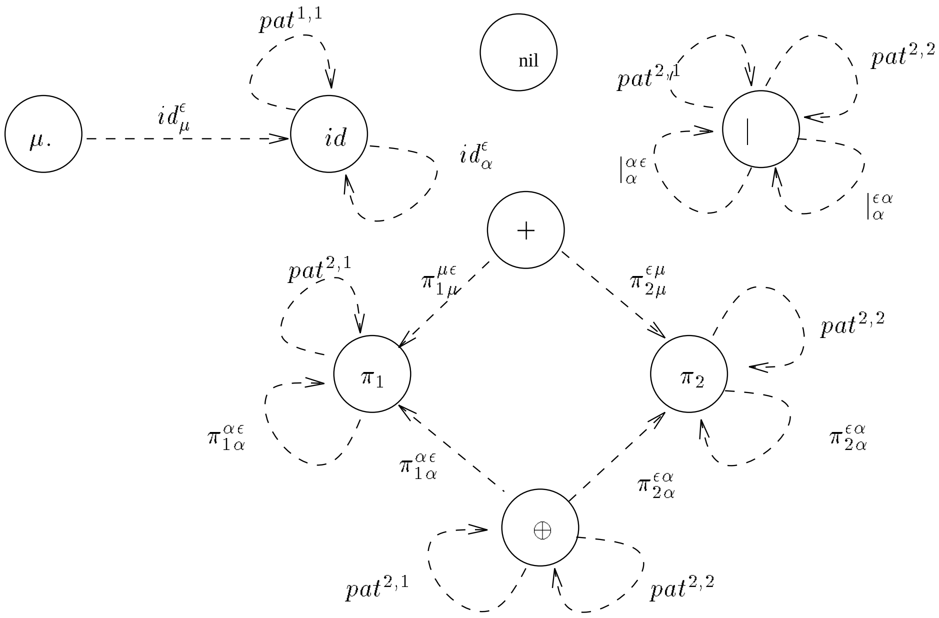

Figure 3: Another presentation for specifications

Example 4 *Starting from the CCS example, here is a proof* $(\mathcal{P})$ *of the CCS transition* $\alpha.\alpha.nil \mid \beta.nil \overset{a:\alpha}{\rightarrow} id(\alpha.nil) \mid \beta.nil$ *where* $a = |_{\alpha}^{\alpha\epsilon} (id_{\epsilon}^{\alpha}(\epsilon), \epsilon)$.

$$(\mathcal{P}) \qquad \frac{\dfrac{\alpha.nil \overset{\epsilon:\epsilon}{\rightarrow} \alpha.nil}{\alpha.\alpha.nil \overset{id_{\alpha}^{\epsilon}(\epsilon):\alpha}{\longrightarrow} id(\alpha.nil)} \quad \beta.nil \overset{\epsilon:\epsilon}{\rightarrow} \beta.nil}{\alpha.\alpha.nil \mid \beta.nil \overset{|_{\alpha}^{\alpha\epsilon}(id_{\alpha}^{\epsilon}(\epsilon),\epsilon):\alpha}{\longrightarrow} id(\alpha.nil) \mid \beta.nil}$$

3.4 Specifications of Asynchronous Transition Systems

The following proposition shows a canonical construction of an ATS from a specification P.

Proposition 1 *For each basic SOS specification* P*, the structure* G_P *is an ATS, where* $G_P \overset{\text{def}}{=} \langle T_\Sigma, E_P, I_P, \rightarrow_P, l\rangle$*, with*

- T_Σ *is the set of closed terms over* Σ*,*
- $\rightarrow_P \subseteq T_\Sigma \times E_P \times T_\Sigma$ *is the set of transitions than can be proved in* R*.*
- $E_P \overset{\text{def}}{=} \{(a : \mu) : p \overset{a:\alpha}{\rightarrow}_P p'\}$.
- $l(a : \mu) \overset{\text{def}}{=} \mu$,
- I_P *is the greatest binary relation over* E_P*, such that for all* aI_Pb*,*

$$\begin{array}{ll} \forall i \in Dom(a) \cap Dom(b).\ ((a(i) \diamond b(i) \vee (a(i) = \epsilon) \vee (b(i) = \epsilon)) & (1) \\ a \neq b & (2) \end{array}$$

Proof The proof is routine : it consists in showing that relation $\rightarrow_P$ is deterministic, forward stable and commutative. We leave it to the reader. $\square$

In Definition 1 the requirement (2) that $a \neq b$ is redundant whenever P is a irreflexive specification, i.e. for all axiom $r \in R$, source and target combinators are distinct..

Lemma 1 *Condition (2) in Proposition 1 is redundant if and only if the specification P is irreflexive.*

Proof If P is irreflexive, then $\forall e \in E_P$ there exists a position p (corresponding to the application of an axiom), where $e(p) \neq \epsilon$ and $e(p) \not\diamond e(p)$. So condition (2) is redundant.

If P is not irreflexive then there is a reflexive axiom with schematic rule ρ, s.t. $\rho \diamond \rho$. The schematic of the proof which uses only one application of this axiom is $\rho(\epsilon \ldots \epsilon)$, and we need the condition (2) in order to preserve I_P irreflexive. $\square$

4 General Congruence Theorem

4.1 Motivations for coolness and irreflexivness hypothesis

Coolness hypothesis[Blo95] is clearly necessary if one wants to abstract from τ's. Indeed, take the combinator + of CCS which does not fulfill Clause 1. in Definition 7: we have $\tau.\alpha.nil \equiv_{ST} \alpha.nil$ but $\tau.\alpha.nil + \beta.nil \not\equiv_{ST} \alpha.nil + \beta.nil$, because when the first process only makes its τ move, there is no possible equivalent move for the second one. + combinator is not the worst case one can think of, since it is manageable if we consider a rooted version of ST-bisimulation. A really bad combinator which does not fulfill coolness can be, for example, a combinator which corresponds to the identity for any visible action, and stops the process when a τ occurs. Then process $\alpha.\beta.nil$ and $\alpha.\tau.\beta.nil$, are no more ST-bisimulation equivalent when put in this context.

Clause 2. of Definition 7 is also necessary : take a combinator which simply renames τ into a visible action α, call it *alarm*. Then, $\tau.nil \equiv_{ST} \tau.\tau.nil$, but $alarm(\tau) \not\equiv_{ST} alarm(\tau.\tau.nil)$ as they respectively correspond to $\alpha.nil$ and $\alpha.\alpha.nil$, which are not even trace equivalent.

Regarding *irreflexivness hypothesis*, Table 1 describes a non-irreflexive specification, because of axiom $(r_{\surd})$.

$$(r_{\surd})\ \frac{}{\surd \overset{\surd^{\alpha}:\alpha}{\longrightarrow} \surd} \qquad (r_{\uparrow})\ \frac{}{\uparrow \overset{\downarrow^{\alpha}:\alpha}{\longrightarrow} \downarrow} \qquad (r_{\downarrow})\ \frac{}{\downarrow \overset{\uparrow^{\alpha}:\alpha}{\longrightarrow} \uparrow}$$

$$(r_1)\ \frac{y \overset{b:\alpha}{\longrightarrow} y'}{f(x,y) \overset{f^{\alpha}_{\epsilon\alpha}(\epsilon,b):\alpha}{\longrightarrow} f(x,y')} \qquad (r_2)\ \frac{x \overset{a:\alpha}{\longrightarrow} x' \quad y \overset{b:\alpha}{\longrightarrow} y'}{f(x,y) \overset{f^{\alpha}_{\alpha\alpha}(a,b):\alpha}{\longrightarrow} f(x',y')}$$

Table 1:

Although $\surd$ and $\uparrow$ are ST-bisimulation equivalent - they both can only perform the infinite sequence α^{ω}, with no branching point and no concurrent events -, when put in the same context, namely $f(\surd, .)$, they turn out to have a different features regarding concurrency and therefore are no more ST-bisimulation equivalent : notice that $f^{\epsilon\alpha}_{\alpha} \diamond f^{\alpha\alpha}_{\alpha}$ and $\surd_{\alpha} \diamond \surd_{\alpha}$ but $\downarrow_{\alpha} \not\diamond \downarrow_{\alpha}$. Indeed, from $f(\surd, \surd)$ it is possible to perform two independent events :

$$f(\surd, \surd) \overset{f^{\epsilon\alpha}_{\alpha}(\epsilon, \surd_{\alpha})}{\longrightarrow} \qquad \text{and} \qquad f(\surd, \surd) \overset{f^{\alpha\alpha}_{\alpha}(\surd_{\alpha}, \surd_{\alpha})}{\longrightarrow}$$

whereas, starting from $f(\surd, \uparrow)$, the only possible transitions are

$$f(\surd, \uparrow) \overset{f^{\epsilon\alpha}_{\alpha}(\epsilon, \downarrow_{\alpha})}{\longrightarrow} \qquad \text{and} \qquad f(\surd, \uparrow) \overset{f^{\alpha\alpha}_{\alpha}(\surd_{\alpha}, \downarrow_{\alpha})}{\longrightarrow}$$

but the two involved events $f^{\epsilon\alpha}_{\alpha}(\epsilon, \downarrow_{\alpha})$ and $f^{\alpha\alpha}_{\alpha}(\surd_{\alpha}, \downarrow_{\alpha})$ are not independent.

4.2 General congruence theorems

Hypothesis for coolness and irreflexivity are the right notions since we prove here that they are sufficient hypothesis to ensure the congruence results for ST-bisimulation. Moreover, we show that coolness can be slightly relaxed when dealing with rooted ST-bisimulation

Let us first fix some vocabulary, inspired from Bloom's terminology.

Definition 9 *A combinator f of P is said to be* tame (in P) *if it is cool and for all other combinator g of P, if f $\dashrightarrow$ g then g is tame (in P). In other words, tame combinators in P form the greatest set of cool combinators which is closed under $\dashrightarrow$ relation.*

A combinator f is tamable (in P) *if for each rule*

$$\frac{u_{i_1} \xrightarrow{\mu_{i_1}} v_{i_1} \; \ldots \; u_{i_m} \xrightarrow{\mu_{i_m}} v_{i_m}}{f(u_1, u_2 \ldots, u_n) \xrightarrow{\mu} g(v_1, v_2 \ldots, v_n)}$$

then g is tame in P and if the rule above is not an axiom then there exist also rules

$$\frac{u_i \xrightarrow{\tau} v_i}{f(u_1, u_2 \ldots, u_n) \xrightarrow{\tau} g(v_1, v_2 \ldots, v_n)} \qquad \forall i = i_1, \ldots i_m \qquad \frac{u_{i_1} \xrightarrow{\mu_{i_1}} v_{i_1} \; \ldots \; u_{i_m} \xrightarrow{\mu_{i_m}} v_{i_m}}{g(u_1, u_2 \ldots, u_n) \xrightarrow{\mu} g(v_1, v_2 \ldots, v_n)}$$

In CCS specification, the set of tamed combinators is $\{\alpha.\}_{\alpha \in Act} \bigcup \{id, \pi_1, \pi_2, |\}$. The remaining combinator $+$ is tamable. Notice that all the CCS combinators are tamable, but that in the general case being tame does not imply being tamable.

Definition 9 can be interpreted as follows : in the context of a tamable combinator f, any possible transition leads to a tame context. Although patience rules are missing, it is possible through a gentle τ move to enter the tame portion of the specification at a place g where similar rules as for f can apply.

Under certain conditions, tame combinators preserve patience rules : the following lemma shows in which case patience rules propagate among cool combinators.

Lemma 2 *If f and g are cool s.t. $f \overset{\rho}{\dashrightarrow} g$, $f \overset{\sigma}{\dashrightarrow} g$ and $\rho \diamond \sigma$, with $\rho = g_{\mu}^{\eta_1 \eta_2 \ldots \eta_n}, \sigma = g_{\nu}^{\zeta_1 \zeta_2 \ldots \zeta_n}$ then for all k s.t. $\eta_k \neq \epsilon$ (we also have $\zeta_k \neq \epsilon$) we have $g \overset{pat^{n,k}}{\dashrightarrow}$, $pat^{n,k} \diamond \rho$ and $pat^{n,k} \diamond \sigma$.*

Proof Omitted. $\square$

Theorem 1 *Let G_P be the ATS associated to a given irreflexive specification P and let $\mathcal{R}_P$ be the greatest ST-bisimulation of G_P. Then for all $t_1, ..., t_n, u_1, ..., u_n \in T_\Sigma$ and $f \in \Sigma^n$, if f is tame in P, then*

$$\mathcal{R}_P : \vec{t} \equiv_{ST} \vec{u} \text{ implies } \mathcal{R}_P : f(\vec{t}) \equiv_{ST} f(\vec{u}) \tag{2}$$

Proof Let $\mathcal{R}$ bet the smallest subset of $ST(G_P) \times ST(G_P) \times \mathcal{P}(E_P \times E_P)$ containing $\mathcal{R}_P$ and all the trios

$$(f(\vec{t}) + \sum_J \rho^j(\vec{a^j}), f(\vec{u}) + \sum_J \rho^j(\vec{b^j}), \{(\rho^j(\vec{a^j}), \rho^j(\vec{b^j})) | j \in J\})$$

such that $J_k \overset{\text{def}}{=} \{j \in J | a_k^j \neq \epsilon\}$ (the indices of the events where the sub-term k is active). Notice that $\forall j, k.\ a_k^j = \epsilon \iff b_k^j = \epsilon$ (this is due that ρ_j ask for ϵ always in the same place).

$$\forall k = 1, \ldots n, (t_k + \sum_{J_k} a_k^j, u_k + \sum_{J_k} b_k^j, \{(a_k^j, b_k^j) | j \in J_k\}) \in \mathcal{R}_P$$

The proof that $\mathcal{R}$ is an ST-bisimulation of G_P is easy. For lack of space, this proof will appear in a full version of the paper (see appendix).

$\square$

Theorem 2 *Let G_P be the ATS associated to a given irreflexive specification P. Then for all $t_1, ..., t_n, u_1, ..., u_n \in T_\Sigma$ and $f \in \Sigma^n$, if f is tamable in P, then*

$$\vec{t} \equiv_{rST} \vec{u} \text{ implies } f(\vec{t}) \equiv_{rST} f(\vec{u}) \tag{3}$$

Proof Easy. $\square$

5 Conclusion

In this paper, we have presented general results for compositionality: we have proved that compatibility of process description language combinators w.r.t. ST-semantics can be decided syntactically. ST-semantics is considered in the setting of Asynchronous Transition Systems (ATS), where ST-states are easily representable.

Concerning the SOS specifications, we have considered so-called *basic* SOS specifications and shown how to derive from them an Asynchronous Transition System. This process has its own interest as we propose a way to deal with invisible actions (τ's). To our knowledge, this is a new contribution. Then, we established that when (and only when) the specification fulfills the syntactic *irreflexivness* hypothesis, i.e. no axiom is independent with itself, the contexts built up from so-called *tame* combinators of this specification preserve ST-bisimulation equivalence. We also considered a super family of previous contexts by defining *tamable* combinators (enabling us to deal with the non-deterministic choice of CCS, among many others) and showed a similar result but for a variant of ST-bisimulation. Notice that what we call Rooted ST-bisimulation in this paper, is not a fully ST-based equivalence, since the first transition has no duration. We will it discuss later in the conclusion.

Obviously, we could have investigated many other truly concurrent equivalences. Actually, in the absence of τ's, the results can be found in [EHP95] : we obtained congruence theorems for all equivalences.

In the presence of τ's, equivalences which are not ST-based equivalences, e.g. those based on partial orders, are not fully interesting : indeed, no equivalence from *weak interleaving bisimulation* to *branching history preserving bisimulation* is compatible with the refinement of actions, as showed R. Van Glabbeek and the second author (see [Pin93]). Nevertheless, we conjecture a positive result for any known concurrent equivalence, because, as we will discuss it later, the refinement of actions is not definable in basic SOS specifications.

Analysing our approach, one might ask for few choices. First of all, about ST-states, we did not consider ongoing τ's executions, although it might be very useful and relevant to study certain properties. But there is no influence here as ST-bisimulation allows free τ's execution.

Secondly, it must be emphasised that the coding of τ rules can be handled differently. However, nice properties of Lemma 4 encourages us to follow this approach. Nevertheless, a more complex coding for τ rules would enable us to work with finer equivalences than the proposed rooted ST-bisimulation. A natural proposal can be

Definition 10 (rooted ST-bisimulation) *We say that two states s and r are rooted ST-bisimilar ($s \equiv_{rST} r$) iff $s \equiv_{ST} r$ and*

- *if $s \xrightarrow{e:\tau} s'$ then exists a path with 1 or more τ, $r \xrightarrow{e^1:\tau}\xrightarrow{e^2:\tau} \cdots \xrightarrow{e^m:\tau} r'$, such that $s' \equiv_{ST} r'$.*
- *vice-versa.*

With the definition above, Theorem 2 does not hold anymore. This problem deserves being studied carefully.

We discuss now the framework of the paper, to see somehow if the constraints over the specification formats can be overtaken. Basic SOS specifications are reasonably powerful since they contain CCS and many others. However, it is not sufficient to describe anything! For example, [Ech93] showed that the refinement of actions cannot be treated. Enlarging the kind of specifications is at the moment an open question, but we can straightaway rule out SOS specifications with negative premises : concurrent objects underlying such specifications would not in general be an Asynchronous Transition Systems.

It is then natural to question the choice for ATS. As these models are a kind of Transition Systems, they permit an extension of usual operational interleaving semantics, while providing a way to define ST-states. But ATS suffer from a lack of expressivness: it is not possible to represent the leftmerge operation, or the priority one. On the other hand, priority combinator is naturally representable in models for ST-states (see [DER95]). As in our context, the relevant objects are ST-states, we are tempted to get rid of ATS (as well as Transition Systems with Independence [SNW93], and Trace Automata [Sta89]). In other words, a perspective for improving the present work, would consist in searching for a semantic model with ST-states as primitive objects (see [Gou94]) s.t. it can be described in SOS specifications.

References

[BD92] E. Badouel and Ph. Darondeau. Structural operational specifications and trace automata. In *Proc. CONCUR'92, Stony Brook, NY, LNCS 630*, pages 302–316. Springer-Verlag, August 1992.

[Bed87] M. A. Bednarczyk. *Categories of Asynchronous Systems.* PhD thesis, Univ. Sussex, October 1987. Available as CS R 1/88.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BIM88] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: preliminary report. In *Proc. 15th ACM Symp. Principles of Programming Languages, San Diego, CA*, pages 229–239, January 1988.

[Blo95] B. Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science*, 146:25–68, 1995.

[DER95] P. R. D'Argenio, J. V. Echagüe, and L. Ramos. Priorities in a truly concurrent model (in spanish). In *Proc. XXIV Jornadas Argentinas de Informática e Investigación Operativa*, 1995.

[Ech93] J. V. Echagüe. *Sémantique des Systèmes Réactifs: Raffinement, Bisimulations et Sémantique Opérationnelle Structurée dans les Systèmes de Transitions Asynchrones.* Thèse de Doctorat, I.N.P. de Grenoble, France, January 1993.

[EHP95] J. V. Echagüe, Z. Habbas, and S. Pinchinat. *Structural Operational Semantics Specifications for True Concurrency.* In *Proc. 15th International Conference of the Chilean Society in Information Science, Chili*, November 1995.

[Gla90] R. J. van Glabbeek. The refinement theorem for ST-bisimulation semantics. Research Report CS-R9002, CWI, January 1990.

[Gla93] R. J. van Glabbeek. Full abstraction in structural operational semantics (extended abstract). In *Proc. AMAST'93, Enschede, NL*, pages 77–84. Springer-Verlag, June 1993.

[Gou94] E. Goubault. *The Geometry of Concurrency.* Thèse de Doctorat, École Normale Supérieure, France, December 1994.

[Gro89] J. F. Groote. Transition system specifications with negative premisses. Research Report CS-R8950, CWI, December 1989.

[Gro93] J. F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.

[GV87] R. J. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 224–242. Springer-Verlag, June 1987.

[GV89] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence (extended abstract). In *Proc. 16th ICALP, Stresa, LNCS 372*, pages 423–438. Springer-Verlag, July 1989.

[GV92] J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

[Mil89] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(2):227–247, 1989.

[Pin93] S. Pinchinat. *Des Bisimulations pour la Sémantique des Systèmes Réactifs.* Thèse de Doctorat, I.N.P. de Grenoble, France, January 1993.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Lect. Notes, Aarhus University, Aarhus, DK, 1981.

[Shi85] M. W. Shields. Deterministic asynchronous automata. In *Formal Methods in Programming.* North-Holland, 1985.

[Sim85] R. De Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[SNW93] V. Sassone, M. Nielsen, and G. Winskel. A classification of models for concurrency. In *Proc. CONCUR'93, Hildesheim, Germany, LNCS 715*, pages 82–96. Springer-Verlag, August 1993.

[Sta89] E. W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 64:221–269, 1989.

[Vog91] W. Vogler. Bisimulation and action refinement. In *Proc. STACS'91, Hamburg, LNCS 480*, pages 309–321. Springer-Verlag, February 1991.

[Wal88] D. J. Walker. Bisimulations and divergence. In *Proc. 3rd IEEE Symp. Logic in Computer Science, Edinburgh*, July 1988.

[Wei89] W. P. Weijland. *Synchrony and Asynchrony in Process Algebra.* PhD thesis, Univ. Amsterdam, June 1989.

APPENDIX

The following lemma present a characterization of $ST(G_P)$

Lemma 3 *The set $ST(G_P)$ is the least set $\mathcal{D} \subseteq T_\Sigma \times \mathcal{P}(E_P)$ containing*

$$\mathcal{D}_0 \stackrel{\text{def}}{=} \{f + \sum_J \rho^j \ \mid\ f \in \Sigma^0 \wedge \exists g \in \Sigma^0 \forall j \in J.(\rho^j = g_{\alpha_j} \wedge f \overset{g_{\alpha_j}}{\dashrightarrow}) \wedge \forall j \neq l \in J.(g_{\alpha_j} \diamond g_{\alpha_l})\}$$

and s.t. $f(t_1, ...t_n) + \sum_J \rho^j(a_1^j, ..., a_n^j) \in \mathcal{D}$ whenever

- *exists g such that for all $j \in J$, $\rho^j = g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}$*
- *for all $j \in J$, $f \overset{g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}}{\dashrightarrow}$*
- *for all $j \neq l \in J$, $g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j} \diamond g_{\alpha_l}^{\eta_1^l \ldots \eta_n^l}$*
- *for all $k = 1, ..., n$, $t_k + \sum_{J_k} a_k^j \in \mathcal{D}$ where $J_k \stackrel{\text{def}}{=} \{j \in J | a_k^j \neq \epsilon\}$*

  P is irreflexive iff for all $f + \sum_J g_{\alpha_j} \in \mathcal{D}_0$ $f \neq g$.
  If P is irreflexive then for all $f + \sum_J g_{\alpha_j} \in \mathcal{D}_0$ $|J| \leq 1$.

The following lemma show that patience rules sometimes propagate between cool combinators (in this sense, between tame ones).

Lemma 4 *If f and g are cool, $f \overset{\rho}{\dashrightarrow} g$, $f \overset{\sigma}{\dashrightarrow} g$ and $\rho \diamond \sigma$, with $\rho = g_\mu^{\eta_1 \eta_2 \ldots \eta_n}, \sigma = g_\nu^{\zeta_1 \zeta_2 \ldots \zeta_n}$ then for all k s.t. $\eta_k \neq \epsilon$ or $\zeta_k \neq \epsilon$, $g \overset{pat^{n,k}}{\dashrightarrow}$, $pat^{n,k} \diamond \rho$ and $pat^{n,k} \diamond \sigma$.*

Theorem 1 *Let G_P be the ATS associated to a given irreflexive specification P and let $\mathcal{R}_P$ be the greatest ST-bisimulation of G_P. Then for all $t_1, ..., t_n, u_1, ..., u_n \in T_\Sigma$ and $f \in \Sigma^n$, if f is tame in P, then*

$$\mathcal{R}_P : \vec{t} \equiv_{ST} \vec{u} \textit{ implies } \mathcal{R}_P : f(\vec{t}) \equiv_{ST} f(\vec{u})$$

Proof We $\mathcal{R}$ to be the smallest subset of $ST(G_P) \times ST(G_P) \times \mathcal{P}(E_P \times E_P)$ containing $\mathcal{R}_P$ and all the trios

$$(f(\vec{t}) + \sum_J \rho^j(\vec{a^j}), f(\vec{u}) + \sum_J \rho^j(\vec{b^j}), \{(\rho^j(\vec{a^j}), \rho^j(\vec{b^j}))| j \in J\})$$

whenever

$$\forall k = 1, \ldots n, (t_k + \sum_{J_k} a_k^j, u_k + \sum_{J_k} b_k^j, \{(a_k^j, b_k^j)| j \in J_k\}) \in \mathcal{R}_P$$

where $J_k \stackrel{\text{def}}{=} \{j \in J | a_k^j \neq \epsilon\}$.
Notice that for each trio exists g s.t. for all $j \in J$, $\rho^j = g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}$ and $\forall j, k.\ a_k^j = \epsilon \iff b_k^j = \epsilon$, since both a_k^j and b_k^j are sub-terms of ρ^j at the same position k.

We claim that $\mathcal{R}$ is an ST-bisimulation on G_P. As a description of the proof techniques we have chosen the less easy case for ST-bisimulation, namely the case 2a of Definition 3 where the process engages in a new action concurrently with already ongoing ones :

Assume $(f(\vec{t}) + \sum_J \rho^j(\vec{a^j}), f(\vec{u}) + \sum_J \rho^j(\vec{b^j}), \{(\rho^j(\vec{a^j}), \rho^j(\vec{b^j}))| j \in J\}) \in \mathcal{R}$, and $f(\vec{t}) + \sum_J \rho^j(\vec{a^j}) + \rho(\vec{a}) \in ST(G_P)$. By lemma 3 this state can be written $f(\vec{t}) + \sum_J g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}(\vec{a^j}) + g_\alpha^{\eta_1 \ldots \eta_n}(\vec{a})$ where

- $f \overset{g_\alpha^{\eta_1 \ldots \eta_n}}{\dashrightarrow}$ and $\forall j.\ (f \overset{g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}}{\dashrightarrow})$,
- $\forall j \in J.\ (\rho^j \diamond \rho)$,

- $\forall k.\, \forall j \in J_k.\, (a_k \neq \epsilon \Rightarrow a_k^j I_P a_k)$.

Because f is cool, $\forall k = 1 \ldots n.\, (a_k \neq \epsilon \Rightarrow l(a_k) \neq \tau)$, then, for all $a_k \neq \epsilon$ there exists $t_k + \sum_{J_k} a_k^j + a_k \in ST(G_P)$ and by definition of $\mathcal{R}_P$ there exists a path

$$u_k \xrightarrow{e_k^1:\tau} \xrightarrow{e_k^2:\tau} \ldots \xrightarrow{e_k^{m_k}:\tau} u_k' \xrightarrow{b_k}$$

s.t.

- $\forall i = 1 \ldots m_k, j \in J_k (e_k^i I b_k^j)$,
- $u_k' + \sum_{J_k} b_k^j + b_k \in ST(G_P)$, and
- $(t_k + \sum_{J_k} a_k^j + a_k, u_k' + \sum_{J_k} b_k^j + b_k, \{(a_k^j, b_k^j) | j \in J_k\} \cup \{(a_k, b_k)\}) \in \mathcal{R}$.

As f is cool, there exists a patience rule $pat^{n,k}$ for position k in f. We can construct a sequence of events,

$$pat^{n,k}(\epsilon, \ldots e_k^1, \ldots \epsilon), \;\; pat^{n,k}(\epsilon, \ldots e_k^2, \ldots \epsilon), \;\; \ldots \;\; pat^{n,k}(\epsilon, \ldots e_k^{m_k}, \ldots \epsilon)$$

which can be applied to $f(\vec{u})$ (and in general to any other term of the form $f(\ldots u_k, \ldots)$). Concatenating all such sequences for each k, we build a path starting in $f(\vec{u})$ and end up $f(\vec{u'})$ from which $\rho(\vec{b})$ can be executed.

By lemma 4 and the definition of $\mathcal{R}_P$ for all $k = 1, ..., n$, for all $j = 1, ..., m_k$, $pat^{n,k}(\epsilon, ..., e_k^j, ..., \epsilon) I_P g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}(\vec{b^j})$ and $g_\alpha^{\eta_1 \ldots \eta_n}(\vec{b}) I_P g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}(\vec{b^j})$, so the good candidate is

$$f(\vec{u'}) + \sum_J g_{\alpha_j}^{\eta_1^j \ldots \eta_n^j}(\vec{b^j}) + g_\alpha^{\eta_1 \ldots \eta_n}(\vec{b})$$

$\square$