Synchronous Modeling and Asynchronous deployment of Mobile Processes

Jean-Pierre Talpin

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Synchronous modeling and asynchronous deployment of mobile processes

Jean-Pierre Talpin

N°3893

March 2000

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--3893--FR+ENG

INRIA
RENNES

Synchronous modeling and asynchronous deployment of mobile processes

Jean-Pierre Talpin

Thème 1 — Réseaux et systèmes
Projet Ep-Atr

Rapport de recherche n ° 3893 — March 2000 — 60 pages

Abstract: We introduce the programming model consisting of pre-ordered transition systems (SPOTS) for modeling synchronous and asynchronous concurrent systems in the presence of mobility and of dynamicity.

The model of SPOTS provides an operational and a denotational semantics to an expressive calculus of mobile processes. Encodings of related asynchronous, synchronous and functional calculi (JOIN, FUSION and λ) are provided.

The data-structure of SPOTS is shown to have a canonical, hierarchic, representation. This representation is the basis for the decisions of key properties for a correct deployment of synchronous system specifications on asynchronous architecture.

The property of *endochrony* (i.e. the equivalence between the synchronous (internal) and asynchronous (external) observation of a system) determines which components of a system can be compiled separately and executed autonomously;

The property of *isochrony* (i.e. the equivalence between the synchronous and asynchronous compositions of endochronous components) determines which components of a system can be distributed over a network without loss of information.

Key-words: concurrency, synchrony and asynchrony, mobile processes

(Résumé : tsvp)

Modélisation synchrone et déploiement asynchrone de processus mobiles

Résumé : Nous proposons un modèle de programmation fondé sur la notion de système de transitions pré-ordonné afin de décrire des systèmes concurrents, synchrones et asynchrones, en présence de mobilité et de dynamicité.

Le modèle proposé comprends la sémantique opérationel et la sémantique dénotationnelle d'un calcul de processus expressif dans lequel des calculs apparentés (FUSION, JOIN, λ) peuvent être exprimés.

La structure de donnée sur laquelle repose ce modèle possède une forme canonique qui permet d'exprimer les propriétés essentielles de désynchronisation:

la propriété d'*endochronie* (c.á.d. l'équivalence entre les observations synchrones et asynchrones d'un système) permet de déterminer les composants d'un système pouvant être compilés séparément.

la propriété d'*isochronie* (c.á.d. l'équivalence entre les compositions synchrones et asynchrones de composants) permet de déterminer les composants d'un système pouvant être déployés sur un réseau asynchrone.

Mots-clé : théorie de la concurrence, modèles synchrones et asynchrones, processus mobiles

Synchronous modeling and asynchronous deployment of mobile processes

Jean-Pierre Talpin

INRIA – IRISA, campus de Beaulieu, F-35042 Rennes

We introduce the programming model consisting of pre-ordered transition systems (SPOTS) for modeling synchronous and asynchronous concurrent systems in the presence of mobility and of dynamicity. The model of SPOTS provides an operational and a denotational semantics to an expressive calculus of mobile processes. Encodings of related asynchronous, synchronous and functional calculi (JOIN, FUSION and λ) are provided. The data-structure of SPOTS is shown to have a canonical, hierarchic, representation. The hierarchic normal form of a SPOTS allows to intuitively express:

- the property of *endochrony* (i.e. the equivalence between the synchronous (internal) and asynchronous (external) observation of a system) to determine which components of a system can be compiled separately and executed autonomously;
- the property of *isochrony* (i.e. the equivalence between the synchronous and asynchronous compositions of endochronous components) to determine which components of a system can be distributed over a network without loss of information.

The decision of these key properties guaranty a correct deployment of synchronous system specifications on asynchronous network architecture.

1. Introduction

Synchronous programming [2,5,12] has been proposed as an efficient approach for the design of reactive and real-time systems. It has been widely publicized, using the idealized picture of "zero time" computation and instantaneous broadcast communication. Distributed architectures do not, however, obey the ideal "zero time" model of perfect synchrony.

Similarities and formal links between synchrony and asynchrony have nonetheless been discussed in the literature already, thus questioning the oversimplified vision of "zero time" computation and instantaneous broadcast communication. Early paper [3] informally discussed the link between perfect synchrony and token-based asynchronous data-flow networks. The first formal study of desynchronization, in [7], establishes precise relations between so-called well-clocked synchronous functional programs and the subset of Kahn networks amenable to "buffer-less" evaluation. Since then, the issue of synchronous program "desynchronization" has been investigated by several authors, e.g. [4].

Taking advantage of the rich background of the synchronous programming technology for the purpose of engineering reactive and distributed systems, we cast the theory

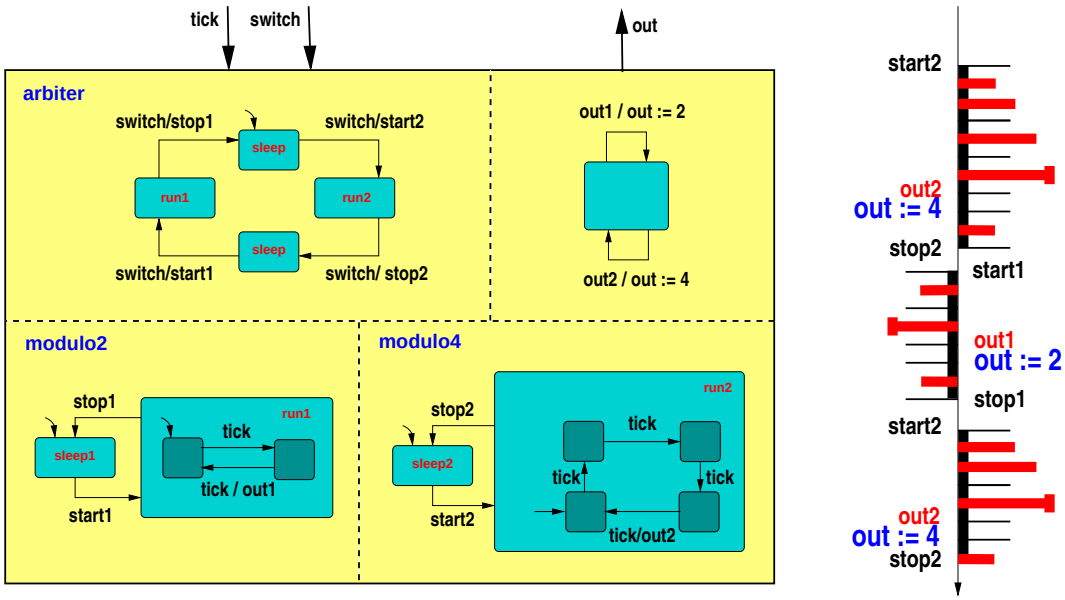
of *synchronous programming* and *desynchronization* in the age of mobility and present a model, consisting of pre-order transition systems, which provides symmetric theories of synchronous and asynchronous communications, related by the formal properties of *endochrony* and *isochrony*, in the presence of mobility and dynamicity.

The property of endochrony (see, e.g., example 14) is defined by the equivalence between the synchronous (internal) and asynchronous (external) observation of a system. The components of a system which satisfy the property of endochrony are robust to an asynchronous environment (e.g. a stream of input values) and can hence be compiled separately and executed autonomously.

The property of isochrony (see, e.g., example 16) is defined by the equivalence between the synchronous and asynchronous compositions of endochronous components. For two endochronous components of a system, the property of isochrony means that their composition can be specified synchronously (to ease verification and compilation) and then safely deployed on an asynchronous network infrastructure without any possible loss of information.

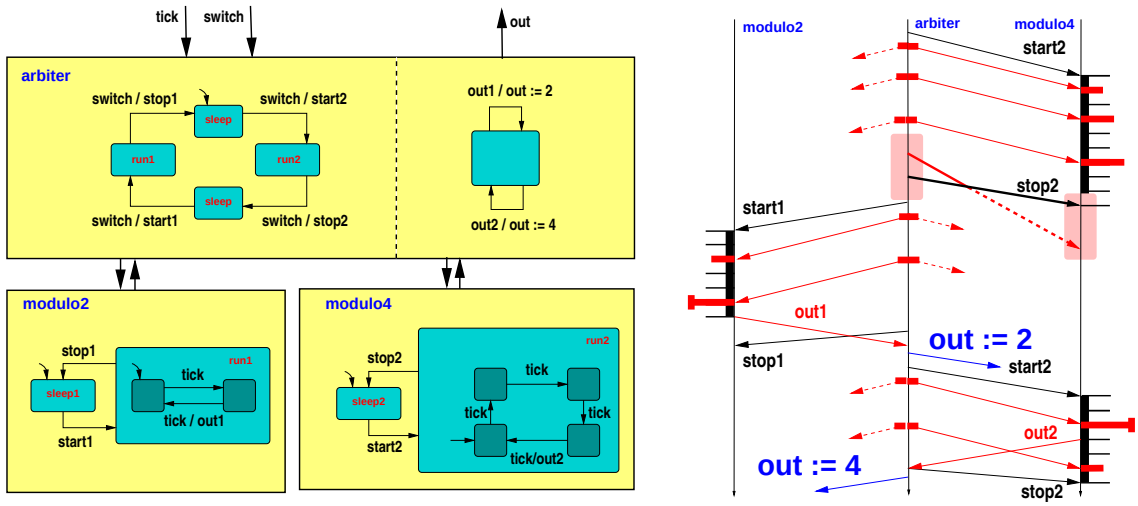
As a result, SPOTS underlies a methodology for correctly deploying of synchronous system designs on asynchronous architectures.

Example 1 (desynchronization) *The following* STATECHART *will support part of our discussion on the issues of desynchronization throughout the paper. It consists of four synchronous automata: an arbiter, two counters (modulo 2 and 4) and an output. The input message* switch *commands the state transitions of the* arbiter. *When the* arbiter *is in either of the* run *state, the* modulo *automata count the occurrences of the input message* tick, *and forward the appropriate* out *count to the* output *automaton. The diagram (on the right) describes the progression of a sequential (synchronous) execution of the* STATECHART *(the tick, vertical, black line). Messages (the red, horizontal lines) are perfectly scheduled one after the other.*

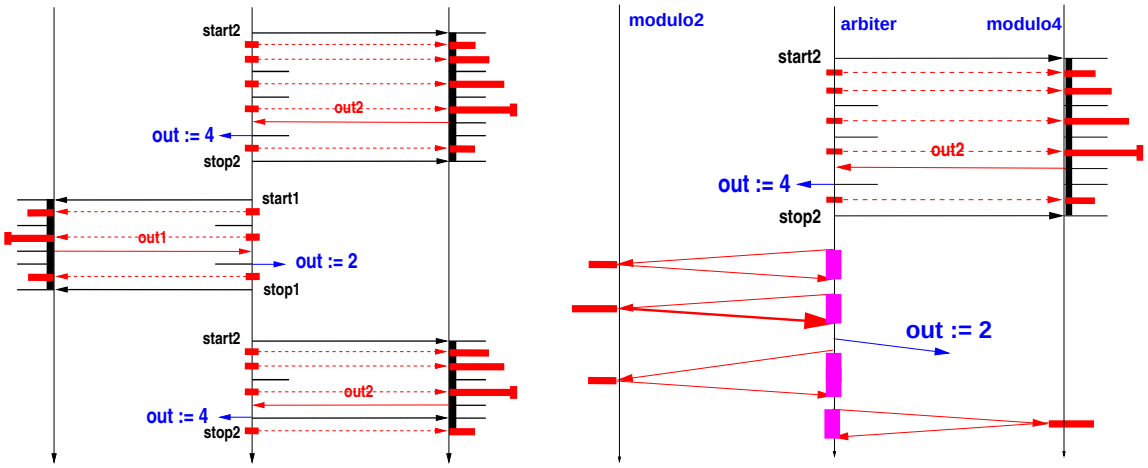


The next STATECHART *positions the problem of desynchronization by depicting a first, brute force and unsuccessful attempt to deploy the components of the* STATECHART *over*

an asynchronous network. In this STATECHART, *we have glued together the* arbiter *and the emitter to form a single, synchronous, component, and we have deployed the* modulo *counters over an asynchronous network. As race conditions prevail in such an environment, we observe in the grey boxes of the message sequence chart (on the right) an interleaving between occurrences of the messages* tick *and* stop2. *The sequence outlines a symptomatic consequence of this interleaving: message* out:=2 *occurs before* out:=4, *as opposed to its sequential execution above. Whence the loss of serial semantics. The cause of this phenomenon can easily be traced from the sequence diagram: there is no causal connection between the events* tick *and* switch. *As a consequence, the activation and suspension of the counters* modulo2 *and* modulo4 *are not coordinated. As a result, the messages* out:=2 *and* out:=4 *cannot be properly scheduled.*



This undesirable effect can be corrected by fully distributing the synchronous semantics, as illustrated (message sequence, left): the step is broadcast and is used to activate each communicating STATECHART. *By doing this we restore the semantics, even globally. However, each step is blocking. This is often unacceptable as a mode of execution. And it turns out that this is unnecessary as is shown next and depicted in the last message sequence (below, right) where* modulo2 *is non-blocking.*



The example indicates the tool needed to make desynchronization systematic and, unlike the last STATECHART*, compositional (message sequence, right). To this end, we introduce the notion of* hierarchic normal form *for synchronous specifications (section 4.2). To prove the correctness of our approach, we introduce a semantical framework in which synchrony and asynchrony can be jointly handled and the preservation of semantics can be proved.*

1.1. Related Work

Early papers [3] informally discussed the link between perfect synchrony and token-based asynchronous data-flow networks. A first formal study [7] established a precise relation between well-clocked synchronous functional programs and the subset of Kahn networks amenable to buffer-less evaluation. Since then, the issue of *desynchronization* has been investigated by several authors. In [4], an extensive analysis of the links between synchrony and asynchrony was provided. The proposed vision of asynchrony encompasses distributed systems, in which no global synchronization state is available and communications are not instantaneous. This extension allows to handle incomplete designs, specifications, properties, architectures, and executable programs, in a unified framework, for both synchronous and asynchronous semantics. We share the vision proposed in [4] around the principles of isochrony and propose decision procedures to implement them.

On one hand, and from a theoretical perspective, correspondences between the synchronous and asynchronous theories of concurrency have been studied in the context of the theory of CCS [16], of the SPROC and ASPROC interaction categories [1], of synchronous structures [21]. On the other hand, numerous models and concepts have been proposed to reason on calculi of mobile processes. One is the notion of action structure [17], which provides a category-theoretic model for the π-calculus. Another is the general concept of concurrent constraints [22,25], which consists of a basic logical framework that finds ad-hoc instances in specific domains of the theory of concurrency (e.g. reactive systems [25], mobile asynchronous systems [22]).

1.2. Contributions

We introduce a structure of pre-order transition systems (SPOTS) which provides a unified view of both synchronous and asynchronous models of concurrency in the presence of mobility (section 2). SPOTS form a semantical framework in which encodings of asynchronous calculi (e.g. JOIN and FUSION calculi) as well as synchronous languages (e.g., synchronous stream functions) are expressed (section 3).

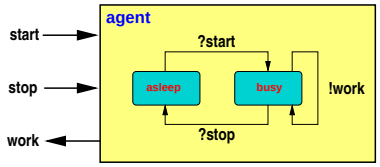
SPOTS establishes formal connections between synchronous and asynchronous models of concurrency in the presence of mobility (section 4) by providing decision procedure for the properties of *endochrony* and *isochrony* (section 4.3) via the construction of a canonical, *hierarchical* representation of transition systems (section 4.2) .

As a result, SPOTS provide a unified formalism for an integrated development of distributed and reactive systems, starting from early requirement specifications down to architecture deployment, in the presence of both synchronous (local) and asynchronous (remote) communications, in the presence of mobility and of dynamicity.

Example 2 (Pre-order transition systems) *The notion of* SPOTS *can informally be presented by considering more familiar formalisms such as automata. Let us for instance*

consider the definition of an agent *that switches between the states* asleep *and* busy *upon receipt of the events* stop *and* start*, and that emits the event* work *when* busy*.*

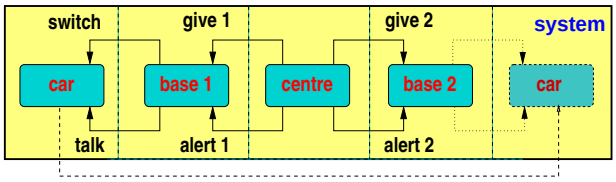
Pre-order transition systems rely on messages, causality relations and choice to describe the behaviour of a system. The SPOTS *of the* agent *consists of a description of its possible reactions to input events and initial conditions. Reactions* (e.g. $s_{1,2,3}$) *are assembled by the choice operator* $+$ *to form the union of its possible behaviours. Each reaction consists of state transitions* (e.g. *from* $\mathsf{agent}^-(\mathsf{asleep})$ *to* $\mathsf{agent}(\mathsf{busy})$) *and of ordered events* (e.g. *the input event* start *causes (or is scheduled or serialized before) the event* $\mathsf{agent}(\mathsf{busy})$*). Note that input messages are the minima of the relation ".;". Also note that, in the examples, we will often write* x *in place of* $x()$ *for an event* (i.e. *a message with no arguments).*



$$\begin{aligned} f_{\mathsf{agent}} = 1 \;& +(\mathsf{agent}^-(\mathsf{asleep}) \,\|\, \mathsf{start});\mathsf{agent}(\mathsf{busy}) && : s_1 \\ & +\mathsf{agent}^-(\mathsf{busy});(\mathsf{work} \,|\, \mathsf{agent}(\mathsf{busy})) && : s_2 \\ & +(\mathsf{agent}^-(\mathsf{busy}) \,\|\, \mathsf{stop});\mathsf{agent}(\mathsf{asleep}) && : s_3 \end{aligned}$$

As it will now be depicted, in a way akin to the definition of the π-calculus [18] starting from CCS [16], we now seamlessly bring the model of pre-order transition systems in the context of mobile computing.

Example 3 (mobility) *To this end, we now consider the introductory example of [18], which describes the interaction between a mobile phone in a car and its base. The flow-graph of [18] depicts a* car *connected to* base_1*. The* centre *is alerted on the move of the* car *within the area of* base_2 *and handles the appropriate* switch *to maintain the connection.*



In SPOT*, the* car *is either silent (transition* 1*), talking (event* talk*) or switching from one base to another (message* switch*). The receipt of the message* talk *is serialized with a transition of* car *(term* $\mathsf{talk};\mathsf{car}(\mathsf{talk},\mathsf{switch})$*). For all ports* t' *and* s'*, the receipt of the message* switch *with* (t', s') *causes a transition from* $\mathsf{car}(t, s)$ *to* $\mathsf{car}(t', s')$*.*

$$\begin{aligned} f_{\mathsf{car}} = \forall \mathsf{talk}, \mathsf{switch}.1+ & \quad (\mathsf{car}^-(\mathsf{talk},\mathsf{switch}) \,|\, \mathsf{talk});\mathsf{car}(\mathsf{talk},\mathsf{switch}) \\ & +\forall t', s'.(\mathsf{car}^-(\mathsf{talk},\mathsf{switch}) \,|\, \mathsf{switch}(t', s'));\mathsf{car}(t', s') \end{aligned}$$

The centre *switches between the* base_1 *and the* base_2 *by receiving the* alert_i *events and by forwarding the interface* $(\mathsf{talk}_i, \mathsf{switch}_i)$ *to the disconnected base* $j \neq i$ *(message* give*).*

$$\begin{aligned} f_{\mathsf{centre}} = 1 \;\; & + \;\; (\mathsf{centre}^-(\mathit{tt}) \,|\, \mathsf{alert}_2);(\mathsf{give}_1(\mathsf{talk}_2, \mathsf{switch}_2) \,\|\, \mathsf{centre}(\mathit{ff})) \\ & + \;\; (\mathsf{centre}^-(\mathit{ff}) \,\|\, \mathsf{alert}_1);(\mathsf{give}_2(\mathsf{talk}_1, \mathsf{switch}_1) \,|\, \mathsf{centre}(\mathit{tt})) \end{aligned}$$

Each base switches to active mode (state $\mathsf{base}_i(\mathit{tt})$*) by receiving the alert message. When active, it can talk to the car (message* talk*). Upon receipt of the give message, it forwards new addresses via the switch message and goes idle.*

$$\begin{aligned} f_{\mathsf{base}_{i=1,2}} = 1+ & \quad (\mathsf{base}_i^-(\mathit{ff}) \,\|\, \mathsf{alert}_i);\mathsf{base}_i(\mathit{tt}) \\ & + \quad \mathsf{base}_i^-(\mathit{tt});(\mathsf{talk}_i \,|\, \mathsf{base}_i(\mathit{tt})) \\ & +\forall t', s'.(\mathsf{base}_i^-(\mathit{tt}) \,\|\, \mathsf{give}_i(t', s'));(\mathsf{switch}_i(t', s') \,\|\, \mathsf{base}_i(\mathit{ff})) \end{aligned}$$

The telephone switching system p_{system} is modularly specified as the synchronous composition of autonomous processes $p_{\mathsf{car}}, p_{\mathsf{centre}}, p_{\mathsf{base_i}}$ owning a state and a behaviour.

$$\begin{array}{rcl} p_{\mathsf{car}} & = & \exists \mathsf{car}.(\mathsf{do}\ f_{\mathsf{car}}\ \mathsf{in}\ \mathsf{car}(\mathsf{talk}_1, \mathsf{switch}_1)) \\ p_{\mathsf{centre}} & = & \exists \mathsf{centre}.(\mathsf{do}\ f_{\mathsf{centre}}\ \mathsf{in}\ \mathsf{centre}(tt)) \\ \forall i = 1,2, p_{\mathsf{base_i}} & = & \exists \mathsf{base_i}.(\mathsf{do}\ f_{\mathsf{base_i}}\ \mathsf{in}\ \mathsf{base_i}(i = 1)) \\ p_{\mathsf{system}} & = & \exists (\mathsf{alert_i}, \mathsf{give_i}, \mathsf{switch_i}, \mathsf{talk_i})_{i=1}^{2}.(p_{\mathsf{car}} \,|\, p_{\mathsf{centre}} \,\|_{i=1}^{2} p_{\mathsf{base_i}}) \end{array}$$

We will now give the necessary formal definitions in order to observe, as in [18], how the process p_{system} switches from base_1 to base_2 upon a move of the car and the simultaneous choice of a reaction of the centre and the bases.

2. Synchronous pre-order transition systems

Figure 1 defines the domains under consideration, comprising port names $x \in \mathcal{X}$, variables $v \in \mathcal{V}$ (i.e. quantifiers) and constants $c \in \mathcal{C}$. We write $u \in \mathcal{U}$ for a name and $\tilde{u} \in \cup_{0 \leq n < \omega} \mathcal{U}^n$ for a tuple of names. We write $\mathcal{D}_u$ for the domain of values of a name u and assume every u to be associated to a given data-type. We write $l \in \mathcal{L}$ for a label, i.e., a port name x, a variable or a source x^- (i.e. a reference to the previous value of x).

$x \in \mathcal{X}$	port
$v \in \mathcal{V}$	variable
$c \in \mathcal{C} \supset \mathbb{B} + \mathbb{Z}$	constant
$g \in \bigcup_{n>0} \mathcal{C}^n \to \mathcal{C}$	function
$u \in \mathcal{U} = \mathcal{C} + \mathcal{X} + \mathcal{V}$	names
$l \in \mathcal{L} = \mathcal{X} + \mathcal{X}^- + \mathcal{V}$	label

Figure 1. Namespaces for synchronous pre-order transition systems

2.1. Structure

The primitive structure of a synchronous pre-order transition system s (figure 2) consists of messages $l(u)$ and of reflexive and transitive ordering. A term s is either the silent transition 1, a message $l(u)$ s.t. $u \in \mathcal{D}_l$ or a relation $s;s'$, which specifies that s is a cause of s' (causality) or that s preceeds s' (sequence). The term $\mathsf{a}(e)$ is an asserition guarded by the expressin e (details section 2.7). We write $s \,\|\, s'$ for the synchronous composition of the pre-orders s and s', defined by the union $s \cup s'$ of s and s' iff $u = u'$ for all $(l(u), l(u'))$ of (s, s'). Terms s are subject to the equivalence relations detailed in the figure 4 (rules (par) and (seq)).

2.2. Notations

We write $l(u) \subseteq s$ the membership of a message $l(u)$ to a pre-order s (figure 3). We write $\mathrm{fv}_{\mathcal{V}}(.)$, $\mathrm{fv}_{\mathcal{L}}(.)$, $\mathrm{fv}_{\mathcal{X}}(.)$ for the set of free variables, locations, ports of a term (figure 26). We

$$
\begin{array}{lcll}
e & ::= & u \mid g(\tilde{e}) & \text{(guard)} \\
s & ::= & 1 \mid l(u) \mid \mathsf{a}(e) \mid s;s' \mid s \,|\, s' & \text{(pre-order)} \\
p & ::= & s \mid s+s' \mid p \,|\, p \mid \mathsf{do}\, f\, \mathsf{in}\, p \mid \exists x.p & \text{(process)} \\
f & ::= & p \mid f+f' \mid f \,|\, f' \mid \forall v.f \mid \exists x.f & \text{(family)}
\end{array}
$$

Figure 2. Syntax for synchronous pre-order transition systems

write $\cup$ and $\cap$ for the union and intersection of pre-orders by considering an isomorphic presentation of pre-orders s as pairs consisting of a set of messages $\mathcal{M} \subseteq \mathcal{L} \times \mathcal{U}$ and a relation $; \subseteq \mathcal{M}^2$. We write $\mathrm{pred}_{l(u)}s$ and $\mathrm{succ}_{l(u)}s$ for the set of immediate predecessors and predecessors of $l(u)$ in a pre-order s. We write $\min s$ and $\max s$ for the minimal and maximal elements of a pre-order s.

$$
l(u) \subseteq l(u) \qquad (l(u) \subseteq s \,|\, s') \wedge (l(u) \subseteq s;s') \Leftrightarrow (l(u) \subseteq s) \vee (l(u) \subseteq s')
$$

$$
\begin{array}{rcl}
\min s & = & \{l(u) \subseteq s \mid \mathrm{pred}_{l(u)}s = \emptyset\} \\
\max s & = & \{l(u) \subseteq s \mid \mathrm{succ}_{l(u)}s = \emptyset\} \\
\mathrm{pred}_{l(u)}s & = & \{l'(u') \subseteq s \mid l'(u');l(u) \subseteq s\} \setminus \{l(u)\} \\
\mathrm{succ}_{l(u)}s & = & \{l'(u') \subseteq s \mid l(u);l'(u') \subseteq s\} \setminus \{l(u)\}
\end{array}
$$

Figure 3. Notations for synchronous pre-order transition systems

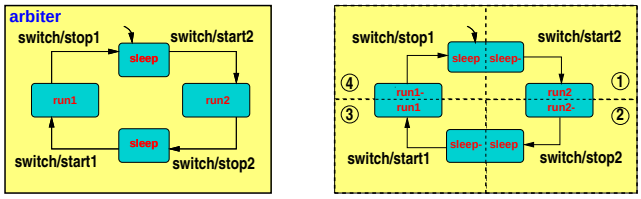
2.3. Conventions

By construction, pre-orders s satisfy the well-formedness criterion $\mathrm{wf}(s)$ which requires that, for all l, there exists a unique $u \in \mathcal{D}_l$ s.t. $l(u) \subseteq s$. Rather than directly manipulating pre-orders, we merely consider the structure of directed graphs which generate them. Hence, reflexive closures are implicit (c.f. figure 4, rules (seq)) and transitive closures explicit as s^* wherever required (e.g. figure 21). In particular, hiding a message $l(u)$ from a pre-order s, written $s \backslash l(u)$, consists of removing $l(u)$ from s and replacing all relations $l(u);l'(u')$ and $l'(u');l(u)$ in s by $l'(u');l''(u'')$.

2.4. Processes

A process p (figure 2) is either a pre-order s (which describes a state), a synchronous composition $p \,\|\, p'$ or a definition $\mathsf{do}\, f\, \mathsf{in}\, p$ of initial state p and of transitions f. By inductive extension of the membership notation to processes, we write $l(u) \subseteq (\mathsf{do}\, f\, \mathsf{in}\, p)$ iff $l(u) \subseteq p$. A family f is a non-empty set of pre-orders p (i.e. $1 \sqsubseteq f, \forall f$) assembled with the choice operator $+$ (figure 4, rules (or)). The choice of a reaction p in f, is written $p \sqsubseteq f$ iff $\exists f', f = p+f'$.

Example 4 (family) *Let us reconsider the* arbiter *defined in section 1 and observe the correspondence between automata and pre-orders.*



In the arbiter*, four state-transitions are specified in reaction to the input message* switch*. Each of them incurs a state-transition (from a* run *mode to a* sleep *mode) and the occurrence of an output message (a* start *or a* stop *command to the* modulo *counters).*

$$f_{\mathsf{arbiter}} = \left(\begin{array}{ll} 1 + (\mathsf{arbiter}^-(\mathsf{sleep2}) \,\|\, \mathsf{switch});(\mathsf{start2} \,\|\, \mathsf{arbiter}(\mathsf{run2})) & : s_1 \\ \quad + (\mathsf{arbiter}^-(\mathsf{run2}) \,\|\, \mathsf{switch});(\mathsf{stop2} \,\|\, \mathsf{arbiter}(\mathsf{sleep1})) & : s_2 \\ \quad + (\mathsf{arbiter}^-(\mathsf{sleep1}) \,\|\, \mathsf{switch});(\mathsf{start1} \,\|\, \mathsf{arbiter}(\mathsf{run1})) & : s_3 \\ \quad + (\mathsf{arbiter}^-(\mathsf{run1}) \,\|\, \mathsf{switch});(\mathsf{stop1} \,\|\, \mathsf{arbiter}(\mathsf{sleep2})) & : s_4 \end{array}\right)$$

A pre-order transition system represents this structure by decomposing it into atomic reactions s, and by rendering the behaviour of the system as a choice of a reaction s within a set f, depending on the state (the $\mathsf{arbiter}^-$*) and the inputs. For instance, the transition from* sleep2 *to* run2 *state requires the input* switch *and produces the output* stop2*. The term* 1 *stands for the silent transition.*

2.5. Synchronous composition

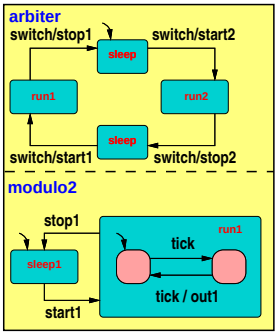
Synchronous composition extends to families by considering synchronously composable pre-orders. We say that $p \sqsubseteq f$ and $p' \sqsubseteq f'$ are *composable*, written $p \bowtie p'$, iff all labels l of the interface $\mathrm{fv}_{\mathcal{L}}(f) \cap \mathrm{fv}_{\mathcal{L}}(f')$ between f and f' are simultaneously present in (resp. or absent from) both p and p' with the same value.

$$\forall p \sqsubseteq f, \forall p' \sqsubseteq f', p \bowtie p' \Leftrightarrow \forall l \in \mathrm{fv}_{\mathcal{L}}(f) \cap \mathrm{fv}_{\mathcal{L}}(f'), l(u) \subseteq p \Leftrightarrow l(u) \subseteq p'$$

The synchronous composition of two processes p and p' belonging to the families $f \,|\, f'$ requires a mutual agreement $p \bowtie p'$ on the presence of ports x belonging to the interface $\mathrm{fv}_{\mathcal{L}}(f) \cap \mathrm{fv}_{\mathcal{L}}(f')$. We write $\sum$ and $\prod$ for enumerated choices and compositions (e.g. $\sum_{i=1}^{n} p_i$ for $p_1 + \ldots + p_n$).

$$f \,|\, f' \;=\; 1 + \sum_{p \sqsubseteq f}^{p' \sqsubseteq f' \,|\, p \bowtie p'} (p \,|\, p')$$

Example 5 (composition) *The synchronous composition of the* arbiter *(example 4) with the* modulo *counter amounts to modeling the choice of a reaction in which the ports of the interface (*i.e. start1 *and* stop1*) are simultaneously present or absent.*



For instance, s_a and s_3 shares the event start1*, s_b and s_4 the event* stop1*. In contrast, s_1 and s_2 interleave with s_c.*

$$f_{\mathsf{modulo2}} = \left(\begin{array}{l} 1+(\mathsf{modulo2}^-(\mathsf{sleep1})\,\|\,\mathsf{start1});\mathsf{modulo2}(\mathsf{run1}) \\ \quad +(\mathsf{modulo2}^-(\mathsf{run1})\,\|\,\mathsf{stop1});\mathsf{modulo2}(\mathsf{sleep1}) \\ \quad +f_{\mathsf{run1}}\,\|\,((\mathsf{modulo2}^-(\mathsf{run1})\,\|\,\mathsf{tick});\mathsf{modulo2}(\mathsf{run1})) \end{array}\right) \begin{array}{l} : s_a \\ : s_b \\ : s_c \end{array}$$

$$f_{\mathsf{arbiter}}\,\|\,f_{\mathsf{modulo2}} = 1+(s_3\,\|\,s_a)+(s_4\,|\,s_b)+(s_1+s_2+s_c+(s_1\,|\,s_c)+(s_2\,|\,s_c))$$

Example 6 (composition and mobility) *Let us for instance consider the definition of the* car *expanded for* (talk, switch) *ranging over $(t_i, s_i)_{i=1}^2$. Four reactions $s_{1,..4}$ are possible. Let us now consider a similar expansion of the definition of* base_1 *for a system comprising two of such components. The synchronous composition of* car *and* base_1 *processes amounts to reconstructing the choice of a reaction in which the ports shared by both processes,* i.e. *t_1 and s_1, are either simultaneously present or simultaneously absent in both processes. In the example, $s_b\,\|\,s_1$ share t_1 and $s_c\,\|\,s_3$ share s_1.*

$$f_{\mathsf{car}} = \left(\begin{array}{l} 1+(\mathsf{car}^-(t_1,s_1)\,\|\,t_1);\mathsf{car}(t_1,s_1) \\ \quad +(\mathsf{car}^-(t_2,s_2)\,\|\,t_2);\mathsf{car}(t_2,s_2) \\ \quad +(\mathsf{car}^-(t_1,s_1)\,\|\,s_1(t_2,s_2));\mathsf{car}(t_2,s_2) \\ \quad +(\mathsf{car}^-(t_2,s_2)\,\|\,s_2(t_1,s_1));\mathsf{car}(t_1,s_1) \end{array}\right) \begin{array}{l} : s_1 \\ : s_2 \\ : s_3 \\ : s_4 \end{array}$$

$$f_{\mathsf{base}_1} = \left(\begin{array}{l} 1+(\mathsf{base}_1{}^-(\mathit{ff})\,\|\,\mathsf{alert}_1);\mathsf{base}_1(\mathit{tt}) \\ \quad +\mathsf{base}_1{}^-(\mathit{tt});(t_1\,\|\,\mathsf{base}_1(\mathit{tt})) \\ \quad +(\mathsf{base}_1^-(\mathit{tt})\,|\,\mathsf{give}_1(t_2,s_2));(s_1(t_2,s_2)\,\|\,\mathsf{base}_1(\mathit{ff})) \end{array}\right) \begin{array}{l} : s_a \\ : s_b \\ : s_c \end{array}$$

$$f_{\mathsf{car}}\,\|\,f_{\mathsf{base}_1} = 1+(s_b\,|\,s_1)+(s_c\,\|\,s_3)+(s_a+s_2+s_4+(s_2\,\|\,s_a)+(s_4\,\|\,s_a))$$

2.6. Quantifiers

The term $\forall v.f$ is a family which has the behaviour of f for all possible value $u \in \mathcal{D}_v$ of the variable $v \in \mathcal{V}$. We write σ for a substitution of names (e.g. $(.)[u/u']$ for replacing u' by u) or messages (e.g. $(.)[l(u)/l'(u')]$). We write $\mathrm{dom}\,\sigma = \{u \,|\, (u)\sigma \neq u\}$ the domain of a substitution, $\mathrm{im}\,\sigma = (\mathrm{dom}\,\sigma)\sigma$ its range.

$$\forall v.f = 1+\sum_{u\in\mathcal{D}_v} f[u/v]$$

Restricting the scope of a port name x to a family f is written $\exists x.f$ and gives a process the ability to define fresh names (as formally specified in definition 2). Quantifiers are subject to equivalence rules (all) and (any) of the figure 4. (notice that the quantifiers of a f cannot flow out of a $\mathsf{do}\, f \,\mathsf{in}\, p$). In the remainder, we consider universally quantified pre-orders p consisting of pre-orders which adhere to an extension of the well-formedness criterion $\mathrm{wf}(p) : \forall v \in \mathrm{fv}_{\mathcal{V}}(p), l(v) \subseteq p \wedge l \neq v$ (i.e. each variable v is related to a node $l(v)$).

$$
\begin{array}{rcl}
l(u);l(u) = l(u) & & (seq) \\
s;1 = 1;s = s & & \\
s;(s' \,|\, s'') = (s;s') \,|\, (s;s'') & & \\
(s' \,|\, s'');s = (s';s) \,|\, (s'';s) & &
\end{array}
\qquad
\begin{array}{rcl}
\mathsf{do}\, 1 \,\mathsf{in}\, p = p & & (def) \\
(\mathsf{do}\, f \,\mathsf{in}\, p) \,|\, (\mathsf{do}\, f' \,\mathsf{in}\, p') = \mathsf{do}\, f \,\|\, f' \,\mathsf{in}\, p \,\|\, p' & & \\
\mathsf{do}\, f \,\mathsf{in}\, (\exists x.p) = \exists y.(\mathsf{do}\, f \,\mathsf{in}\, (p[y/x])) & &
\end{array}
$$

$$
\begin{array}{rcl}
f+f = f & & (or) \\
f+f' = f'+f & & \\
f+(f'+f'') = (f+f')+f'' & &
\end{array}
\qquad
\begin{array}{rcl}
\forall y \notin \mathrm{fv}_{\mathcal{X}}(f) \cup \mathrm{fv}_{\mathcal{X}}(f'), & & (any) \\
f+(\exists x.f') = \exists y.(f+(f'[y/x])) & & \\
f \,\|\, (\exists x.f') = \exists y.(f \,\|\, (f'[y/x])) & &
\end{array}
$$

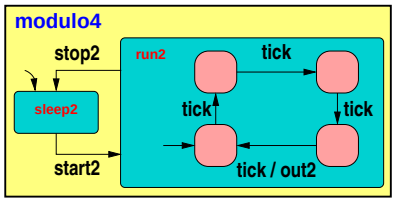
$$
\begin{array}{rcl}
f \,\|\, 1 = f \,\|\, f = f & & (par) \\
f \,\|\, f' = f' \,\|\, f & & \\
f \,\|\, (f' \,\|\, f'') = (f \,\|\, f') \,\|\, f'' & &
\end{array}
\qquad
\begin{array}{rcl}
\forall w \notin \mathrm{fv}_{\mathcal{V}}(f) \cup \mathrm{fv}_{\mathcal{V}}(f'), & & (all) \\
f+(\forall v.f') = \forall w.(f+(f'[w/v])) & & \\
f \,|\, (\forall v.f') = \forall w.(f \,|\, (f'[w/v])) & &
\end{array}
$$

Figure 4. Equivalence rules for pre-order transition systems

2.7. Guards

A guard $\mathsf{a}(e)$ consists of a pseudo-port a (to mean "assert") whose domain is restricted to $\mathcal{D}_{\mathsf{a}} = \{\sharp\}$ and of a boolean expression e. Guards are a key ingredient for the compilation of SPOTS (i.e. for determining their *hierarchic normal form*, def. 10) as they allow to make control explicit. Given e a boolean expression on names, a guard $\mathsf{a}(e) \subseteq p$ means that p can only be triggered if $e = \sharp$. In the examples, we often write $l(e)$ in place of $l(v) \,\|\, \mathsf{a}(v = e)$ for a message of guarded value and, in particular, $l(u_1, \ldots u_n)$ for a message $l(v) \,\|\, \mathsf{a}(v = (u_1, \ldots u_n))$ carrying a tuple of n arguments.

Example 7 (quantifiers and guards) *In the* modulo4 *counter (example 1), enumeration can be used to generically define the default transition.*



In f_{run2}*, this is implemented by stepping from* $\mathsf{run2}^-(n)$ *to* $\mathsf{run2}((n+1) \bmod 4)$ *upon receipt of a* tick*. A guards is used to make control explicit and emit* out2 *when the count* run2 *reaches* 4 ticks*.*

$$f_{\mathsf{run2}} = 1 + \forall m, n. \left(\begin{array}{ll} & (\mathsf{a}(n = (m+1) \bmod 4) \| \mathsf{tick} \| \mathsf{run2}^-(m); \mathsf{run2}(n)) \\ \| & ((\mathsf{a}(n = 0) \| \mathsf{tick}; \mathsf{out2}) + (\mathsf{a}(n \neq 0) \| \mathsf{tick})) \end{array} \right)$$

Example 8 (quantifiers and guards) *Quantifiers and guards allow to give generic definitions to enumerative processes, such as the synchronous composition* $f_{\mathsf{switch}} = f_{\mathsf{nat}} \,|\, f_{\mathsf{even}}$ *of the stream of natural numbers* f_{nat} *with the filter of even numbers* f_{even}*.*

$$\begin{array}{rcll} f_{\mathsf{nat}} & = & 1 + \quad \forall m, n. & (\mathsf{a}(n = m+1) \quad \| \mathsf{nat}^-(m); \mathsf{nat}(n)) \\ f_{\mathsf{even}} & = & 1 + \quad \forall n. & ((\mathsf{a}(n \bmod 2 = 0) \quad \| \mathsf{nat}(n); \mathsf{even}) + (\mathsf{a}(n \bmod 2 = 1) \| \mathsf{nat}(n)) \end{array}$$

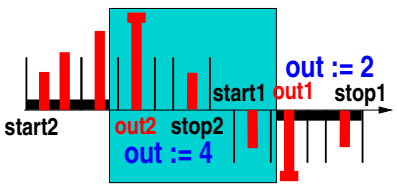
2.8. Synchronous semantics

Just as the meaning of an automaton can be denoted by the set of words it recognizes, the semantics of a SPOTS is defined by the concatenation of reactions. Concatenation consists of merging the maxima $s;x(u)$ of traces and the minima $x^-(u);s'$ of successive reactions as $s;x(u);i_x;s'$ by introducing a term i_x which denotes that x stepped the instant i. In turn, the successive instants $i \in \mathcal{I}$ give a logical model of internal time of a process.

Example 9 (synchronous concatenation) *Let us experience with the stepwise construction of the trace* t *of the process* f_{switch} *defined in example 8. The producer* nat *can either be silent (it performs a* 1 *transition) or choose to increment its state. The consumer* even *successively emits true and false. Concatenation is obtained by the juxtaposition of a pre-order (*e.g. $\mathsf{nat}^-(0);\mathsf{nat}(1)$*) belonging to the family* f_{switch} *whose source (*e.g. $\mathsf{nat}^-(0)$*) matches the current state of the trace* t *(*e.g. $\mathsf{nat}(0)$*) and by inserting the mark* i_{nat} *of an instant. We write* $\rightarrow$ *for causality relations. Repeated application of this procedure yields the construction of the trace(s) of the switch.*

$$\begin{array}{rcl} \left(\begin{array}{l} \mathsf{nat}(0) \\ \|\mathsf{even} \end{array} \right) \bullet f_{\mathsf{switch}} & = & \left(\begin{array}{l} \mathsf{nat}(0) \\ \|\mathsf{even} \end{array} \right) \bullet \left(\mathsf{nat}^-(0);\mathsf{nat}(1) \right) \\ & = & \left(\begin{array}{l} \mathsf{nat}(0); i_{\mathsf{nat}} \cup i_{\mathsf{nat}}; \mathsf{nat}(1) \\ \|\mathsf{even}; \quad i_{\mathsf{even}} \end{array} \right) \\ & = & \left(\begin{array}{l} \mathsf{nat}(0); i_{\mathsf{nat}}; \mathsf{nat}(1) \\ \|\mathsf{even}; \quad i_{\mathsf{even}} \end{array} \right) \end{array}$$

Example 10 (synchronous trace) *Let us now consider the trace of* SPOTS *that corresponds to the grey box of the sequential execution diagram shown in the section 1.*



It consists of the assembly (the concatenation) of small patterns (reactions) that is formed, at every instant, by the unification of sinks $x(c)$ *of the trace with the sources* $x^-(c)$ *of the*

reaction, to form the mark of an instant (an i_x). One can hence observe that tick *and* out2 *are* synchronous, *in the first considered instant, and then, that a* switch *causes a simultaneous* stop2, *etc., as in the many details as in the diagram.*

$$t_{\mathsf{arbiter}} = \begin{pmatrix} \mathsf{tick}\to i^1_{\mathsf{tick}} & \to i^2_{\mathsf{tick}} & \to i^3_{\mathsf{tick}} & \to\mathsf{tick}\to i^4_{\mathsf{tick}} & \to\mathsf{tick}\ldots \\ \downarrow & \mathsf{switch}\to i^2_{\mathsf{switch}} & \to\mathsf{switch}\to i^3_{\mathsf{switch}} & \to i^4_{\mathsf{switch}} & \downarrow\ \ldots \\ \downarrow & \mathsf{stop2}\to i^2_{\mathsf{stop2}} & \downarrow\ \to i^3_{\mathsf{stop2}} & \to i^4_{\mathsf{stop2}} & \downarrow\ \ldots \\ \mathsf{out2}\to i^1_{\mathsf{out2}} & \to i^2_{\mathsf{out2}} & \downarrow\ \to i^3_{\mathsf{out2}} & \to i^4_{\mathsf{out2}} & \downarrow\ \ldots \\ & & \mathsf{start1}\to i^3_{\mathsf{start1}} & \to i^4_{\mathsf{start1}} & \downarrow\ \ldots \\ & & & & \mathsf{out1}\ldots \end{pmatrix}$$

As depicted in example 9, a synchronous trace t (figure 5) consists of messages $x(u)$ and marks $i_x \in \mathcal{S} = \mathcal{I} \times \mathcal{X}$ constructed using an infinite set of instants $i \in \mathcal{I}$. Traces are assembled using causality **;** and composition $\|$. We write r for a run consisting of a trace t and of a set of active definitions f. Notice the containment of terms s (resp. p) in t (resp. r). Equivalence rules for processes (figure 4) extend pointwise to traces and runs.

$$\begin{array}{rcll} t & ::= & x(u) \mid i_x \mid t;t' \mid t \,|\, t' & \text{(trace)} \\ r & ::= & t \mid r \,|\, r' \mid \mathsf{do}\ f\ \mathsf{in}\ r \mid \exists x.r & \text{(run)} \end{array}$$

Figure 5. Synchronous traces and runs of pre-order transition systems

---

We write $t{\uparrow}x$ for the restriction of the trace t to messages of $x \times \mathcal{U}$ and marks of $\mathcal{I} \times x$ (and $(\mathsf{do}\ f\ \mathsf{in}\ r){\uparrow}x = r{\uparrow}x$ by inductive extension to runs). The synchronous composition $t \,|\, t'$ of the traces t and t' is defined by $t \cup t'$ iff $t \bowtie t'$ i.e. $\forall x \in \mathrm{fv}_{\mathcal{X}}(t) \cap \mathrm{fv}_{\mathcal{X}}(t'), t{\uparrow}x = t'{\uparrow}x$.

Definition 1 (synchronous concatenability) *A pre-order s is* concatenable *to a trace t, written $t \bullet^? s$ iff all $x^-(u)$ of s have a matching $x(u)$ maximal in t.*

$$t \bullet^? s \ \Leftrightarrow\ \forall x^-(u) \subseteq s, x(u) \in \max((t{\uparrow}x) \backslash \mathcal{S})$$

By inductive extension to processes,

$$(t \bullet^? \exists x.p) \wedge (t \bullet^? (\mathsf{do}\ f\ \mathsf{in}\ p)) \Leftrightarrow t \bullet^? p$$

The synchronous concatenation $t \bullet s$ of a pre-order s to a trace t is obtained by inserting a mark i_x in place of the sources $x^-(u) \subseteq s$ linked to the sinks $x(u) \in \max(t \backslash \mathcal{S})$ and by removing guards. We write $\mathrm{fv}_{\mathcal{I}}(t)$ for the set of instants i in t.

Definition 2 (synchronous concatenation) *The concatenation of a pre-order s to a trace t satisfying $t \bullet^? s$ is defined by $t \bullet s = (t\sigma) \cup ((s \setminus \mathsf{a}(tt))\sigma)$ where the substitution σ is defined for $i \in \mathcal{I} \setminus \mathrm{fv}_{\mathcal{I}}(t)$ by*

$$\begin{array}{llll} \forall x \in \mathrm{fv}_{\mathcal{X}}(s), & \forall & x^-(u) \subseteq s, & x^-(u)\sigma = i_x, \\ & \forall & x(u) \subseteq s, & x(u)\sigma = i_x;x(u) \\ \forall x \in \mathrm{fv}_{\mathcal{X}}(t), & \forall & x(u) \in \max(t\uparrow x), & x(u)\sigma = x(u);i_x, \\ & \forall & i'_x \in \max(t\uparrow x), & i'_x\sigma = i'_x;i_x \end{array}$$

By inductive extension to processes p and runs r,

$$\begin{array}{rcl} (\mathsf{do}\, f \,\mathsf{in}\, t) \bullet (\mathsf{do}\, f' \,\mathsf{in}\, p) & = & (\mathsf{do}\, f \parallel f' \,\mathsf{in}\, t) \bullet p \\ (\mathsf{do}\, f \,\mathsf{in}\, t) \bullet s & = & \mathsf{do}\, f \,\mathsf{in}\, (t \bullet s) \\ r \bullet \exists x.p & = & \exists x.(r \bullet p) \Leftrightarrow x \notin \mathrm{fv}_{\mathcal{X}}(r) \end{array}$$

2.9. Synchronous semantics

The extension of synchronous concatenation to runs r gives rise to a transition relation (figure 6), written $\bullet r$, which consists of an axiom of concatenation (a) and of inductive rules for making an initial choice (b) and allocating a port name (c).

$$\begin{array}{rcll} \bullet(\mathsf{do}\, f \,\mathsf{in}\, t) & = & \{(\mathsf{do}\, f \,\mathsf{in}\, t) \bullet p \mid p \sqsubseteq f \wedge t \bullet^? p\} & (a) \\ \bullet(\mathsf{do}\, f \,\mathsf{in}\, (s{+}s')) & = & \bullet(\mathsf{do}\, f \,\mathsf{in}\, s) \cup \bullet(\mathsf{do}\, f \,\mathsf{in}\, s') & (b) \\ \bullet(\exists x.r) & = & \bullet r & (c) \end{array}$$

Figure 6. Transition relation for synchronous pre-order transition systems

The operational semantics $r \bullet\!\!\rightarrow r'$ (figure 7) is defined by the choice of a transition $r' \in (\bullet r)$. The denotation $\bullet^\omega p$ of a process p is defined by induction on the transition relation.

$$\begin{array}{rcl} r \bullet\!\!\rightarrow r' & \Leftrightarrow & r' \in (\bullet r) \\ \bullet^0 p & = & \{p\} \\ \forall n \geq 0, \bullet^{n+1} p & = & \cup_{r \in \bullet^n p}(\bullet r), \\ \bullet^\omega p & = & \lim_{n \geq 0} \bullet^n p \end{array}$$

Figure 7. Semantics of synchronous pre-order transition systems

2.10. Synchronous equivalence relations

The equivalence of trace is written $p \simeq p'$ and defined by $\bullet^\omega p = \bullet^\omega p'$ up to isomorphism on $\mathcal{I}$. Terms equality (figure 4) yields the observation that

Property 1 $(p = p') \Rightarrow (p \simeq p')$.

PROOF. Appendix D.

The operational semantics naturally gives rise to a notion of strong barbed (in the sense of [19]) synchronous bisimulation, similar to that of [15], by way of the synchronous observation criterion $\downarrow$.

Definition 3 (synchronous observation) *We say that $x(u)$ is observable in r, written $r\downarrow x(u)$, iff $r \bullet\!\!\rightarrow r'$ and $x(u) \subseteq r'$.*

Definition 4 (synchronous bisimulation) *We say that r_1 and r_2 are synchronously bisimilar, written $r_1 \approx r_2$ iff there exists a strong bisimulation $\mathcal{R}$* s.t. *$r_1 \mathcal{R} r_2$. A strong bisimulation $\mathcal{R}$ is a symetric binary relation* s.t. *$r_1 \mathcal{R} r_2$ implies $r_1 \bullet\!\!\rightarrow r_1' \Rightarrow (r_2 \bullet\!\!\rightarrow r_2' \wedge r_1' \mathcal{R} r_2')$ and $r_1\downarrow x(u) \Rightarrow r_2\downarrow x(u)$.*

Property 2 *Synchronous bisimulation $\approx$ is a congruence relation*

PROOF. Appendix E.

Example 11 (dynamicity) *The combined use of definitions and existential quantifiers makes mobility and dynamicity expressible using the* SPOTS. *Let us for instance reconsider the process f_{base}. The ability to clone as many copies of the* base *as we want can be modeled by embedding its definition into that of f_{clone} as shown.*

$$f_{\mathsf{clone}} = \exists \mathsf{alert}, \mathsf{give}, \mathsf{switch}, \mathsf{talk}, \mathsf{base}.\forall\kappa.(\mathsf{do}\ f_{\mathsf{base}}\ \mathsf{in}\ \overbrace{(\mathsf{base}(\mathit{ff}) \,|\, (\mathsf{clone}(\kappa);\kappa(\mathsf{alert}, \mathsf{give}, \mathsf{switch}, \mathsf{talk})))}^{s_{\mathsf{clone}}})$$

Each occurrence of the message clone *gives rise to a reaction f_{clone} which dynamically creates a copy of the process f'_{base} by:*

1. *performing a transition that selects the reaction f_{clone}, by using the concatenation rule ($\mathsf{do}\, p\, \mathsf{in}\, t$) of the operational semantics and by instantiating the universally quantified term κ to κ' using σ;*
2. *moving the existential quantifiers* alert, give, switch, talk *out of the term by substituting them with fresh names* alert′, give′, switch′, talk′ *using the substitution σ'.*
3. *Eleminating the outer-most existential quantifiers on* alert′, give′, switch′, talk′ *by using the rule for existentially quantified terms ($\exists x.r$);*
4. *activating the behaviour $f_{\mathsf{base}'}$ of the new process (rule for* $\mathsf{do}\, f\, \mathsf{in}\, \mathsf{do}\, f'\, \mathsf{in}\, r$*).*

$$\begin{aligned}
\mathsf{do}\ f_{\mathsf{clone}}\ \mathsf{in}\ 1 \bullet\!\!\rightarrow \quad & \mathsf{do}\ f_{\mathsf{clone}} \mathsf{in}\ 1 \bullet \exists \mathsf{alert}, \mathsf{give}, \mathsf{switch}, \mathsf{talk}, \mathsf{base}.(\mathsf{do}\ f_{\mathsf{base}}\ \mathsf{in}\ s_{\mathsf{clone}}\sigma) \\
= \quad & \exists \mathsf{alert}', \mathsf{give}', \mathsf{switch}', \mathsf{talk}', \mathsf{base}'.(\mathsf{do}\ f_{\mathsf{clone}} \mathsf{in}\ 1 \bullet (\mathsf{do}\ f_{\mathsf{base}}\sigma\ \mathsf{in}\ s_{\mathsf{clone}}\sigma\sigma')) \\
= \quad & \mathsf{do}\ f_{\mathsf{clone}} \mathsf{in}\ (\mathsf{do}\ f_{\mathsf{base}'}\ \mathsf{in}\ (\mathsf{base}'(\mathit{ff}) \,|\, (\mathsf{clone}(\kappa');\kappa'(\mathsf{alert}', \mathsf{give}', \mathsf{switch}', \mathsf{talk}')))) \\
= \quad & \mathsf{do}\ f_{\mathsf{clone}} \,|\, f_{\mathsf{base}'} \mathsf{in}\ (\mathsf{base}'(\mathit{ff}) \,\|\, \mathsf{clone}(\kappa');\kappa'(\mathsf{alert}', \mathsf{give}', \mathsf{switch}', \mathsf{talk}'))
\end{aligned}$$

2.11. Asynchronous semantics

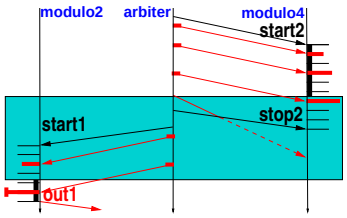
The asynchronous semantics of pre-order transition systems is obtained by abstracting time from traces. In the asynchronous semantics, the reference of time is not unique, as each asynchronously communicating process possesses its own, local, clock. Asynchrony is thus obtained by eliminating any global reference of time.

Example 12 (asynchronous concatenation) *Let us for instance reconsider the trace* t_{switch} *of the* switch *specification. We observe that marks* i *delimit the progression of time. The asynchronous trace of the* switch *is obtained by erasing these marks. The remaining information consists of causality relations between the successive events that occurred during execution.*

$$t_{\mathsf{switch}} = \begin{pmatrix} \mathsf{nat}(0) \to i^0_{\mathsf{nat}} \to \mathsf{nat}(1) \to i^1_{\mathsf{nat}} \to \mathsf{nat}(2) \\ \downarrow \qquad\qquad\qquad\qquad \downarrow \\ \mathsf{even} \to i^0_{\mathsf{even}} \to \quad \to \quad \to i^1_{\mathsf{even}} \to \mathsf{even} \end{pmatrix} (t_{\mathsf{switch}})_\circ = \begin{pmatrix} \mathsf{nat}(0) \to \mathsf{nat}(1) \to \mathsf{nat}(2) \\ \qquad\quad \searrow \qquad\quad \downarrow \\ \qquad\quad \mathsf{even} \to \mathsf{even} \end{pmatrix}$$

Example 13 (asynchronous trace) *The erasure of the synchronization marks from the asynchronous trace of a process has an important impact on the amount of information at disposal. Applying this procedure to the trace depicted in the example 10, leaves* tick *and* switch *(*resp. $\mathsf{out}_{1,2}$ *and* $\mathsf{stop}_{1,2}$*) causaly unrelated. This observation explains the pathology that caused the symptom of race conditions in section 1. Asynchronous composition does not suffice to make every process aware of the messages that are present or absent, at a given time, in other processes. Each process has its own unit of time. The origin of the problem is thus a lack of control on synchronizations.*

$$\boldsymbol{t}_{\mathsf{arbiter}} = \begin{pmatrix} \mathsf{tick} \to \mathsf{tick} \to \mathsf{tick} & \mathsf{switch} \to \mathsf{switch} \\ \downarrow \qquad\quad \downarrow & \downarrow \qquad\quad \downarrow \\ \mathsf{out2} \qquad\quad \mathsf{out1} & \mathsf{stop2} \qquad\quad \mathsf{start1} \end{pmatrix}$$



2.12. Asynchronous traces

The syntax of asynchronous pre-order transition systems is defined by induction on the structure of asynchronous composition starting from atomic synchronous reactions s and processes p. We write $\boldsymbol{p}$, $\boldsymbol{f}$, $\boldsymbol{t}$ and $\boldsymbol{r}$ for asynchronous processes, families, traces and runs.

$$\begin{array}{rcll} \boldsymbol{p} & ::= & p \mid \boldsymbol{p} \parallel \boldsymbol{p}' & \text{(process)} \\ \boldsymbol{f} & ::= & f \mid \boldsymbol{f} \parallel \boldsymbol{f}' & \text{(family)} \\ \boldsymbol{t} & ::= & (t)_\circ \mid \boldsymbol{t} \parallel \boldsymbol{t}' & \text{(trace)} \\ \boldsymbol{r} & ::= & (r)_\circ \mid \boldsymbol{r} \parallel \boldsymbol{r}' & \text{(run)} \end{array}$$

Figure 8. Asynchronous processes, families, traces and runs

Asynchronous traces are defined by the removal of marks $\mathcal{S} = \mathcal{I} \times \mathcal{X}$ from synchronous traces. We write $(t)_\circ = t \backslash \mathcal{S}$ for the asynchronous abstraction of a trace t and $(\mathsf{do}\ f\ \mathsf{in}\ r)_\circ =$

$\mathsf{do}\, f \,\mathsf{in}\, (r)_\circ$. Asynchronous composition of traces is defined by $\boldsymbol{t} \parallel \boldsymbol{t}' = \boldsymbol{t} \cup \boldsymbol{t}'$ iff $\boldsymbol{t} \bowtie \boldsymbol{t}'$. Traces (resp. runs) are composable iff $\boldsymbol{t} \bowtie \boldsymbol{t}'$ (resp. $\mathsf{do}\, \boldsymbol{f} \,\mathsf{in}\, \boldsymbol{r} \bowtie \mathsf{do}\, \boldsymbol{f}' \,\mathsf{in}\, \boldsymbol{r}'$ iff $\boldsymbol{r} \bowtie \boldsymbol{r}'$). The asynchronous composition of traces is subject to the equivalence rules of figure 9.

$$\begin{array}{llllllllll} \boldsymbol{r} \parallel 1 & = & \boldsymbol{r} & & \boldsymbol{r} \parallel \boldsymbol{r}' & = & \boldsymbol{r}' \parallel \boldsymbol{r} & (\mathsf{do}\, \boldsymbol{f} \,\mathsf{in}\, \boldsymbol{r}) \parallel (\mathsf{do}\, \boldsymbol{f}' \,\mathsf{in}\, \boldsymbol{r}') & = & \mathsf{do}\, \boldsymbol{f} \parallel \boldsymbol{f}' \,\mathsf{in}\, \boldsymbol{r} \parallel \boldsymbol{r}' \\ \boldsymbol{r} \parallel \boldsymbol{r} & = & \boldsymbol{r} & \boldsymbol{r} \parallel (\boldsymbol{r}' \parallel \boldsymbol{r}'') & = & (\boldsymbol{r} \parallel \boldsymbol{r}') \parallel \boldsymbol{r}'' & (\mathsf{do}\, \boldsymbol{f} \,\mathsf{in}\, \boldsymbol{r}) \parallel (\mathsf{do}\, \boldsymbol{f}' \,\mathsf{in}\, \boldsymbol{r}) & = & \mathsf{do}\, \boldsymbol{f} \parallel \boldsymbol{f}' \,\mathsf{in}\, \boldsymbol{r} \end{array}$$

Figure 9. Equivalence rules for asynchronous runs

The asynchronous concatenation $\boldsymbol{t} \circ s$ of a pre-order s to a trace $\boldsymbol{t}$ is defined by the removal of instants $i_x \in \mathcal{S}$ from $\boldsymbol{t} \bullet s$ (definitions 1 and 2).

Definition 5 (asynchronous concatenation) *The concatenation of a pre-order s to a trace $\boldsymbol{t}$ satisfying $\boldsymbol{t} \bullet^?$ s is defined by $\boldsymbol{t} \circ s = (\boldsymbol{t} \bullet s) \backslash \mathcal{S}$.*

As in the synchronous semantics, asynchronous concatenation $\circ$ gives rise to a transition relation $\circ\boldsymbol{r}$ (figure 10). This relation defines the denotational semantics $\circ^\omega \boldsymbol{p}$ and the operational semantics $\boldsymbol{r} \circ\!\!\rightarrow \boldsymbol{r}'$ of asynchronous processes.

$$\begin{array}{rcl} \circ(r) & = & \{(r')_\circ \mid r' \in \bullet r\} \\ \circ(\boldsymbol{r}_1 \parallel \boldsymbol{r}_2) & = & \{\boldsymbol{r}'_1 \parallel \boldsymbol{r}'_2 \mid \boldsymbol{r}'_1 \in \circ\boldsymbol{r}_1, \boldsymbol{r}'_2 \in \circ\boldsymbol{r}_2, \boldsymbol{r}'_1 \bowtie \boldsymbol{r}'_2\} \\ \boldsymbol{r} \circ\!\!\rightarrow \boldsymbol{r}' & \Leftrightarrow & \boldsymbol{r}' \in \circ\boldsymbol{r} \\ \circ^0 \boldsymbol{p} & = & \{\boldsymbol{p}\} \\ \forall n \geq 0, \circ^{n+1} \boldsymbol{p} & = & \cup_{\boldsymbol{r} \in \circ^n \boldsymbol{p}} (\circ\boldsymbol{r}) \\ \circ^\omega \boldsymbol{p} & = & \lim_{n \geq 0} \circ^n \boldsymbol{p} \end{array}$$

Figure 10. Semantics of asynchronous pre-order transition systems

---

2.13. Asynchronous equivalence relations

Asynchronous trace equivalence $\boldsymbol{p}_1 \simeq_\circ \boldsymbol{p}_2 \Leftrightarrow \circ^\omega \boldsymbol{p}_1 = \circ^\omega \boldsymbol{p}_2$ is obtained from the denotational semantics. The transition relation gives rise to a weak bisimulation relation similar to that of [15]. We write $\circ\!\!\rightarrow^*$ for a finite sequence of asynchronous transitions.

Definition 6 (asynchronous bisimulation) *Observing $x(u)$ in $\boldsymbol{r}$ is written $\boldsymbol{r} \Downarrow x(u)$ iff $\boldsymbol{r} \circ\!\!\rightarrow^* \boldsymbol{r}'$ and and $x(u) \subseteq \boldsymbol{r}'$. We write $\boldsymbol{r}_1 \approx_\circ \boldsymbol{r}_2$ iff there exists a weak bisimulation $\mathcal{R}_\circ$* s.t. *$\boldsymbol{r}_1 \mathcal{R}_\circ \boldsymbol{r}_2$ and $\boldsymbol{p}_1 \approx_\circ \boldsymbol{p}_2$ iff $\exists \mathcal{R}_\circ, \boldsymbol{p}_1 \mathcal{R}_\circ \boldsymbol{p}_2$. A weak bisimulation $\mathcal{R}_\circ$ is a symmetric binary relation* s.t.*$\boldsymbol{r}_1 \mathcal{R}_\circ \boldsymbol{r}_2$ implies $\boldsymbol{r}_1 \circ\!\!\rightarrow \boldsymbol{r}'_1 \Rightarrow (\boldsymbol{r}_2 \circ\!\!\rightarrow^* \boldsymbol{r}'_2 \wedge \boldsymbol{r}'_1 \mathcal{R}_\circ \boldsymbol{r}'_2)$ and $\boldsymbol{r}_1 \Downarrow x(u) \Rightarrow \boldsymbol{r}_2 \Downarrow x(u)$.*

Property 3 *Asynchronous bisimulation $\approx_\circ$ is a congruence relation*

Proof. Appendix E.

3. Expressivity of pre-order transition systems

In this section, we give an assessment on the expressivity of pre-order transition systems by giving encodings of related asynchronous calculi of mobile processes (the JOIN-calculus and the FUSION-calculus) and of the λ-calculus with synchronous streams into SPOTS.

3.1. Semantics of the JOIN-calculus

In [10], the semantics of the JOIN-calculus is presented under the concept of a chemical abstract machine $D \vdash P$ in which multi-sets of definitions D and processes P asynchronously interact upon the presence of matching patterns J. A process P is a multi-set of messages $x\langle u\rangle$ and inactive definitions $\mathsf{def}\ D\ \mathsf{in}\ P$. A definition D is a multi-set of join-patterns $J \rhd P$ in which J is the combination of messages to be matched in order to trigger the reaction P. Free, receiving and defined names of a term are written $\mathrm{fv}(.)$, $\mathrm{rv}(.)$ and $\mathrm{dv}(.)$.

$$
\begin{array}{rcll}
P & ::= & x\langle u\rangle \mid \mathsf{def}\ D\ \mathsf{in}\ P \mid (P \mid P') \mid 0 & \text{(process)} \\
D & ::= & J \rhd P \mid (D \mid D') \mid 1 & \text{(definition)} \\
J & ::= & x\langle u\rangle \mid (J \mid J') & \text{(join-pattern)}
\end{array}
$$

Figure 11. Syntax of the JOIN-calculus

The operational semantics of the JOIN-calculus (figure 12) consists of a structural equivalence relation $\leftrightharpoons$. It is used to either decompose a process or re-assemble it. In (def), $\mathrm{dom}\,\sigma = \mathrm{dv}(D)$ and $\mathrm{im}\,\sigma$ is fresh. The JOIN-calculus has only one reduction rule $\rightharpoonup$ which consumes a combination of message $J\sigma$ (s.t. $\mathrm{dom}\,\sigma = \mathrm{rv}(J)$) matching a defined pattern J and produces the appropriate instance $P\sigma$ of the defined reaction P. A transition $P \mapsto P'$ in the JOIN-calculus is defined by $P \leftrightharpoons^* \rightharpoonup \leftrightharpoons^* P'$ (a reduction preceded and followed by the application of equivalence rules).

$$
\begin{array}{rcccl}
 & \vdash & 0 & \leftrightharpoons & \vdash \\
1 & \vdash & & \leftrightharpoons & \vdash \\
 & \vdash & P_1 \mid P_2 & \leftrightharpoons & \vdash P_1, P_2 \\
D_1 \mid D_2 & \vdash & & \leftrightharpoons & D_1, D_2 \vdash \\
 & \vdash \mathsf{def}\ D\ \mathsf{in}\ P & \leftrightharpoons & D\sigma & \vdash P\sigma \quad \text{(def)} \\
J \rhd P & \vdash & J\sigma & \rightharpoonup & J \rhd P \vdash P\sigma \quad \text{(red)}
\end{array}
$$

Figure 12. Semantics of the JOIN-calculus

The encoding of the JOIN-calculus into SPOTS is presented in the fig. 13. The asynchronous, point-to-point communication medium of the JOIN-calculus is rendered by the

access to a buffer (of state $x(\mathsf{some}(u))$ when full or $x(\mathsf{none})$ when empty). An input pattern $x\langle u\rangle$ is encoded as $x^-(\mathsf{some}(u));x(\mathsf{none})$ and an output message as $x^-(\mathsf{none});x(\mathsf{some}(u))$. The non-deterministic choice of a reduction starting from a process $|_{i=1}^n P_i$ is rendered by the choice of a continuation $\kappa_i^-(tt) \to \kappa_i(ff)$ (that activate the associated P_i, once) starting from an initial state $\kappa_i(tt)$. The rules for the composition of patterns $J \mid J'$, of definitions $D \mid D'$, for local definitions $\mathsf{def}\ D\ \mathsf{in}\ P$, for atoms 0 and 1, consists of a structural decomposition of the main encoding scheme.

$$
\begin{array}{rl}
[\![1]\!]_{\mathcal{J}} & = 1 \\
[\![D \mid D']\!]_{\mathcal{J}} & = [\![D]\!]_{\mathcal{J}} + [\![D']\!]_{\mathcal{J}} \\
[\![J \rhd P]\!]_{\mathcal{J}} & = \exists \kappa. \forall \mathrm{rv}(J).([\![J]\!]^{\kappa}_{\mathcal{J}} \parallel [\![P]\!]^{\kappa}_{\mathcal{J}}) \\
\forall n \geq 1, [\![|_{i=1}^n J_i]\!]_{\mathcal{J}} & = \prod_{i=1}^n \langle\!\langle J_i \rangle\!\rangle \\
\forall n \geq 1, [\![|_{i=1}^n P_i]\!]_{\mathcal{J}} & = \exists (\kappa_i)_{i=1}^n.(\mathsf{do}\ (1 + \sum_{i=1}^n (\langle\!\langle P_i \rangle\!\rangle \parallel {\kappa_i}^-(tt);\kappa_i(ff)))\ \mathsf{in}\ (\prod_{i=1}^n \kappa_i(tt)))
\end{array}
$$

$$
\begin{array}{rl}
\langle\!\langle 0 \rangle\!\rangle & = 1 \\
\langle\!\langle x\langle u\rangle\ \mathrm{as}\ J \rangle\!\rangle & = x^-(\mathsf{some}(u));x(\mathsf{none}) \\
\langle\!\langle x\langle u\rangle\ \mathrm{as}\ P \rangle\!\rangle & = x^-(\mathsf{none});x(\mathsf{some}(u)) \\
\langle\!\langle \mathsf{def}\ D\ \mathsf{in}\ P \rangle\!\rangle & = \exists \mathrm{dv}(D).(\mathsf{do}\ (1 + [\![D]\!]_{\mathcal{J}})\ \mathsf{in}\ ([\![P]\!]_{\mathcal{J}} \parallel (\prod_{x \in \mathrm{dv}(D)} x(\mathsf{none}))))
\end{array}
$$

Figure 13. Encoding $[\![P]\!]_{\mathcal{J}}$ of the JOIN-calculus

Every pre-order constructed in the encoding of figure 13 corresponds to a possible transition in the JOIN-calculus. We constructively formalize this remark by showing that the encoding $[\![.]\!]_{\mathcal{J}}$ is fully abstract. To this end, we consider the mirror $\approx_{\mathcal{J}}$ of the asynchronous bisimulation relation $\approx_\circ$ in the JOIN-calculus, as defined in [10].

Theorem 1 $P \approx_{\mathcal{J}} P' \Leftrightarrow [\![P]\!]_{\mathcal{J}} \approx_\circ [\![P']\!]_{\mathcal{J}}$

PROOF. Appendix A.

3.2. Semantics of the FUSION-calculus

To probe further on the expressive power of SPOTS, we consider a calculus with asynchronous composition and synchronous communications, the *solos* variant [15] of theFUSION-calculus [23]. In the FUSION-calculus, a process P consists of input and output messages $x(u)$ and $\overline{x}(u)$, asynchronous composition $P \mid P'$, restriction $(v)P$, replication $!P$ and the guard $[u = u']P$ of a process P by an equation $u = u'$.

$$P ::= 0 \mid x(u) \mid \overline{x}(u) \mid P \mid P' \mid (v)P \mid\ !P \mid [u = u']P$$

Figure 14. Syntax of the FUSION-calculus

The structural equivalence of processes is specified by the relation $P \equiv P'$.

$$\begin{array}{rclrclrcll} (v)0 & \equiv & 0 & [u=u]P & \equiv & P & (v)MP & \equiv & M(v)P & (v \notin \mathrm{fv}(M)) \\ (v)(w)P & \equiv & (w)(v)P & !P & \equiv & P \,|\, !P & P \,|\, (v)Q & \equiv & (v)(P \,|\, Q) & (v \notin \mathrm{fv}(P)) \end{array}$$

Figure 15. Equivalence rules in the FUSION-calculus

The transition relation $P \to P'$ consists of four rules: one for composition, one for structural equivalence, one for restriction, and one for reduction. In this last rule, M and M' stand for equations. In (3), the transition $P\sigma$ is enabled upon satisfaction of $u\sigma = u'\sigma$, $M\sigma$, $M'\sigma$, $\mathrm{im}\,\sigma \cap \tilde{v} = \emptyset$ and $\mathrm{dom}\,\sigma = \tilde{v}$.

$$\frac{P \to P'}{P \,|\, Q \to P' \,|\, Q} \qquad \frac{P \equiv Q \to Q' \equiv P'}{P \to P'} \qquad \frac{P \to P'}{(v)P \to (v)P'} \qquad (\tilde{v})(Mx(u) \,|\, M'\overline{x}(u') \,|\, P) \to P\sigma^{(3)}$$

Figure 16. Transitions in the FUSION-calculus

The encoding of the FUSION-calculus (fig. 17) differs from that of the JOIN-calculus because communication is synchronous (i.e. done within a transition and not between two transitions). To model it, we identify every sender/receiver pair by a unique pair of identifiers $(\iota, \overline{\iota})$. An input $x(u)$ is represented by $\exists\iota.\forall\overline{\iota}.x(u,\iota,\overline{\iota})$ and an output $\overline{x}(u)$ by $\forall\iota.\exists\overline{\iota}.x(u,\iota,\overline{\iota})$. Synchronous communication in $P \,|\, Q$ is then rendered by $\|$. Filtering $[u=u']P$ is denoted by a guard $\mathsf{a}(u=u')$ that is propagated within the encoding of $[\![P]\!]_{\mathcal{F}}$. Replication $!P$ is modeled by the definition of $[\![P]\!]_{\mathcal{F}}$. Binding $(v)P$ is denoted by the universal quantification of v, since the FUSION does not discriminate defined names from receiving names.

We state the adequacy of the encoding with respect to the operational semantics of FUSION by showing that, for all ports x of a term P, asynchronous observations of output messages $x(u)$ correspond.

Theorem 2 $\forall x \notin K, \forall u \in \mathcal{D}_x, P \Downarrow x(u) \Leftrightarrow \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota})$

PROOF. Appendix B.

3.3. Synchronous stream functions

We finally consider an encoding of the λ-calculus with synchronous streams [8]. In the syntax of LUCID-synchrone, names are noted u, operations f (from $C^n \to C$), abstractions noted $\lambda x.S$ and applications $S(S')$. The term $\mathsf{rec}\, x\, S$ recursively defines a stream named x. The term $\mathsf{const}\, S$ defines a stream that invariantly emits the value of expression S; $\mathsf{extend}\, S\, S'$ applies the stream of functions S to the stream of values S'; $\mathsf{when}\, S\, S'$ filters

$$
\begin{aligned}
[\![x(u)]\!]^s_{\mathcal{F}} &= \exists\kappa.(\mathsf{do}\ 1 + \exists\iota.\forall\overline{\iota}..(s \mid x(u,\iota,\overline{\iota})) \parallel \kappa^-(\mathit{tt});\kappa(\mathit{ff})\ \mathsf{in}\ \kappa(\mathit{tt})) \\
[\![\overline{x}(u)]\!]^\kappa_{\mathcal{F}} &= \exists\kappa.(\mathsf{do}\ 1 + \forall\iota.\exists\overline{\iota}.(s \parallel x(u,\iota,\overline{\iota}) \mid \kappa^-(\mathit{tt});\kappa(\mathit{ff}))\ \mathsf{in}\ \kappa(\mathit{tt})) \\
[\![P \mid Q]\!]^s_{\mathcal{F}} &= [\![P]\!]^s_{\mathcal{F}} \mid [\![Q]\!]^s_{\mathcal{F}} \\
[\![[u = u']P]\!]^s_{\mathcal{F}} &= [\![P]\!]^{(s \mid \mathsf{a}(u=u'))}_{\mathcal{F}} \\
[\![!P]\!]^s_{\mathcal{F}} &= \mathsf{do}\ 1 + [\![P]\!]^s_{\mathcal{F}}\ \mathsf{in}\ [\![P]\!]^s_{\mathcal{F}} \\
[\![(v)P]\!]^s_{\mathcal{F}} &= \forall v.([\![P]\!]^s_{\mathcal{F}}) \\
[\![P]\!]_{\mathcal{F}} &= [\![P]\!]^1_{\mathcal{F}} \\
[\![0]\!]^s_{\mathcal{F}} &= s
\end{aligned}
$$

Figure 17. Encoding $[\![P]\!]_{\mathcal{F}}$ of the FUSION-calculus

$$
\begin{array}{rcll}
P & ::= & u & \text{(name)} \\
 & \mid & f & \text{(operator)} \\
 & \mid & \lambda x.S & \text{(abstraction)} \\
 & \mid & S(S') & \text{(application)} \\
 & \mid & \mathsf{rec}\ x\ S & \text{(recursive stream)} \\
 & \mid & \mathsf{const}\ S & \text{(constant stream)} \\
 & \mid & \mathsf{extend}\ S\ S' & \text{(extension)} \\
 & \mid & \mathsf{pre}\ S\ S' & \text{(delay)} \\
 & \mid & \mathsf{when}\ S\ S' & \text{(sampling)} \\
 & \mid & \mathsf{merge}\ S\ S'\ S'' & \text{(merge)}
\end{array}
$$

Figure 18. Syntax of synchronous stream functions

the values of stream S' when the value of the boolean stream S is true; the value of $\mathsf{pre}\, S\, S'$ is initially that expression S' and then the previous value of stream S.

The operational semantics $S \rightarrow^r S'$ of synchronous stream functions (figure 25, appendix C) is defined in [8]. The term r here stands for either nil or a value v (a scalar c or s closed S). The term E is used to rename stream names for stepping from a state to another . The principle of the relation $S \rightarrow^r S'$ is to step from an initial state S to a final state S' by synchronously evaluating input streams and by emitting values v on the output stream. A transition $S \rightarrow^{\mathsf{nil}} S'$ means that the output is absent.

The encoding of synchronous stream functions into SPOTS is presented in fig. 19. It distinguishes *eager* λ-terms from *lazy* stream constructors. In the *eager* encoding $[\![S]\!]_\kappa$, names u are directly returned to the continuation κ. In the case of operators on $f \in C^n \rightarrow C$, a guard $\mathsf{a}(v = u(v_{1..n}))$ defines the value v to be returned by an anonymous function x. Similarly, the encoding of $\lambda x.S$ defines a function y that require an argument x and a continuation κ' along which the result of S is emitted. An application $S(S')$ synchronously evaluates S and S' (of respective continuations κ' and κ'' and of respective values v and v') and then calls v with the value v' and the continuation κ.

The encoding of *lazy* stream expressions requires loading values v from solicited input streams, written $\langle\!\langle S \rangle\!\rangle_v$, or just synchronizing, written $\langle\!\langle S \rangle\!\rangle_{\mathsf{nil}}$. The stream of an expression S is accessible along κ in $[\![S]\!]^{\kappa'}_{\kappa}$. The continuation κ' is used if the stream is not solicited. For instance, the expression $\mathsf{rec}\, x\, S$ recursively defines stream named x and $\mathsf{const}\, S$ an anonymous one. The term $\mathsf{extend}\, S\, S'$ loads the values v, v' of the streams S, S' and defines the output stream x by the continuation of the call of v with v'. The term $\mathsf{pre}\, S\, S'$ initially sets the sink zx to the value of the expression S. Then, the source $zx^-(v)$ is loaded, its values returned to x, the value v' of S' loaded and stored in place of v in the register zx. In the term $\mathsf{when}\, S\, S'$, the values of S and S' are loaded. If the value of S is *ff*, then the output stream x is absent. If the value of S is *tt*, then the output stream x is present with the value v of S'. The encoding of merge is identical to that of a when with an *else* branch.

Th. 3 shows that the transition $S \rightarrow^v S'$ of a stream expression in [8] corresponds to the observation of the name u of v along the continuation y of $[\![S]\!]^x_y$. Unlike for the transitions $S \rightarrow^{\mathsf{nil}} S'$, it is not necessary to make the absence of events explicit in SPOTS. Also note that it is not necessary to encode the term E it in order to demonstrate observational equivalence.

Theorem 3

(3a) $S \rightarrow^v S' \Leftrightarrow ([\![S]\!]_\kappa)\!\downarrow(\kappa(x)\,|\,x(u)) \wedge ([\![v]\!]_\kappa)\!\downarrow\kappa(u))$

(3b) $S \rightarrow^{\mathsf{nil}} S' \Leftrightarrow (\exists r \in \bullet([\![S]\!]_\kappa), \kappa(x) \subseteq r \wedge (\forall u, x(u) \notin r))$

PROOF. Appendix C.

4. From synchrony to asynchrony

In this section, we outline the relation between synchronous and asynchronous theories of pre-order transition systems under the practical objective of showing under which properties the synchronous design of a system can safely be deployed on an asynchronous network. This issue is the subject of an in-depth algorithmic study in [27] in the context

$$
\begin{aligned}
[\![u]\!]_\kappa &= \kappa(u)\\
[\![g]\!]_\kappa &= \exists x.(\forall v_{1..n}, \kappa'.(x(v_{1..n},\kappa') \parallel \mathsf{a}(v = g(v_{1..n})) \parallel \kappa'(v)) \parallel \kappa(x))\\
[\![\lambda x.S]\!]_\kappa &= \exists y.(\forall x, \kappa'.(y(x,\kappa') \mid [\![S]\!]_{\kappa'})) \parallel \kappa(y))\\
[\![S(S')]\!]_\kappa &= \exists \kappa', \kappa''.([\![S]\!]_{\kappa'} \parallel [\![S']\!]_{\kappa''} \mid (\forall v, v'.((\kappa'(v) \parallel \kappa''(v')); v(v',\kappa))))
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathsf{const}\ S]\!]_\kappa &= \exists x.(\mathsf{do}\ 1 + \kappa(x) + \exists y.\forall v.([\![S]\!]_y \mid y(v) \mid \langle\!\langle \kappa \rangle\!\rangle_x^v)\ \mathsf{in}\ 1)\\
[\![\mathsf{rec}\ x\ S]\!]_\kappa &= \exists x, y.([\![S]\!]_y \parallel (\mathsf{do}\ 1 + \forall v.(\langle\!\langle y \rangle\!\rangle^v \mid \langle\!\langle \kappa \rangle\!\rangle_x^v)\ \mathsf{in}\ 1))\\
[\![\mathsf{extend}\ S\ S']\!]_\kappa &= \exists x, y, z.([\![S]\!]_y \parallel [\![S']\!]_z \parallel (\mathsf{do}\ 1 + (\langle\!\langle y \rangle\!\rangle^{\mathsf{nil}} \mid \langle\!\langle z \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})\\
&\qquad + \forall f, v.(\langle\!\langle y \rangle\!\rangle^f \parallel \langle\!\langle z \rangle\!\rangle^v \parallel f(v,x) \parallel \kappa(x))\mathsf{in}\ 1))\\
[\![\mathsf{pre}\ S\ S']\!]_\kappa &= \exists x, zx, y.([\![S']\!]_y \parallel (\mathsf{do}\ 1 + (\langle\!\langle y \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})\\
&\qquad + \forall v, w.(\langle\!\langle y \rangle\!\rangle^v \parallel zx^-(w); zx(v) \mid \langle\!\langle \kappa \rangle\!\rangle_x^w)\mathsf{in}\ [\![S]\!]_{zx}))\\
[\![\mathsf{when}\ S\ S']\!]_\kappa &= \exists x, y, z.([\![S]\!]_y \parallel [\![S']\!]_z \parallel (\mathsf{do}\ 1 + (\langle\!\langle y \rangle\!\rangle^{\mathsf{nil}} \mid \langle\!\langle z \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})\\
&\qquad + \forall v.(\langle\!\langle y \rangle\!\rangle^{tt} \langle\!\langle z \rangle\!\rangle^v \mid \langle\!\langle \kappa \rangle\!\rangle_x^v)\\
&\qquad + \forall v.(\langle\!\langle y \rangle\!\rangle^{ff} \langle\!\langle z \rangle\!\rangle^v \mid \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})\ \mathsf{in}\ 1))\\
[\![\mathsf{merge}\ S\ S'\ S'']\!]_\kappa &= \exists x, b, y, z.([\![S]\!]_b \parallel [\![S']\!]_y \mid [\![S'']\!]_z \mid (\mathsf{do}\ 1 + (\langle\!\langle b \rangle\!\rangle^{\mathsf{nil}} \mid \langle\!\langle y \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle z \rangle\!\rangle^{\mathsf{nil}} \mid \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})\\
&\qquad + \forall v.(\langle\!\langle b \rangle\!\rangle^{tt} \parallel \langle\!\langle y \rangle\!\rangle^v \parallel \langle\!\langle z \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle \kappa \rangle\!\rangle_x^v)\\
&\qquad + \forall v.(\langle\!\langle b \rangle\!\rangle^{ff} \parallel \langle\!\langle y \rangle\!\rangle^{\mathsf{nil}} \parallel \langle\!\langle z \rangle\!\rangle^v \mid \langle\!\langle \kappa \rangle\!\rangle_x^v)\ \mathsf{in}\ 1))
\end{aligned}
$$

where $\langle\!\langle \kappa \rangle\!\rangle^{\mathsf{nil}} = \forall x.\kappa(x)$, $\langle\!\langle \kappa \rangle\!\rangle^v = \forall x.(\kappa(x) \parallel x(v))$, $\langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}} = \kappa(x)$, $\langle\!\langle \kappa \rangle\!\rangle_x^v = (\kappa(x) \parallel x(v))$

Figure 19. Encoding $[\![e]\!]_\kappa$ of synchronous stream functions

of the first-order fragment of SPOTS. In this section, we give mathematical evidence on the generalizability of these results in the context of calculi for mobile processes.

4.1. Formal definitions

We start by giving a formal definition to the properties under consideration. The property of endochrony is defined as the equivalence between the internal (synchronous) and external (asynchronous) observations of a process p. It means that every asynchronous observation $\boldsymbol{r}$ of the process p corresponds to a synchronous observation r in which the successive instants i of the execution have been reconstructed from the values of messages $l(u)$ present in $\boldsymbol{r}$. More concretely, it ensures that the process p forms a unit of compilation: interaction with an endochronous process p does not require any knowledge on its internal clock (i.e. any synchronous observation that would not be reconstructible from an asynchronous one).

Definition 7 (endochrony) *p is endochronous iff, for all $\boldsymbol{r} \in \circ^{\omega} p$, there exists a unique $r \in \bullet^{\omega} p$ (up to silent transitions)* s.t. $\boldsymbol{r} = (r)_{\circ}$

Example 14 (endochrony) *To give a concrete and simple illustration of the definition 7, we consider an asynchronous trace $\boldsymbol{t}$ of a pair $p_{1,2}$ of processes.*

$$
\begin{array}{lcl}
p_1 & = & \mathsf{do}\,(1 + (a \,\|\, x(tt)) + (b \,\|\, x(ff)))\,\mathsf{in}\,1 \\
p_2 & = & \mathsf{do}\,\forall v.(1 + (a \,|\, x(v)) + (b \,\|\, x(v)))\,\mathsf{in}\,1
\end{array}
$$

$$
\boldsymbol{t} = \begin{pmatrix}
x(tt) & \to & x(ff) & \to & x(ff) & \to & \dots \\
a & \to & & & & & \dots \\
b & \to & b & \to & & & \dots
\end{pmatrix}
$$

In the process p_1, the consumption of a message a of b available in the queue is controlled by the value carried by the port x. Hence, the only possible message sequence is that described by t_1. In the process p_2, the value of x does not discriminate the choice between consuming an a or a b. Hence, every interleaving of a and b is valid,

$$
t_1 = \begin{pmatrix}
x(tt) & \to & i_x & \to & x(ff) & \to & i_x & \to & x(ff) & \dots \\
a & \to & i_a & \to & & & i_a & & & \dots \\
 & & i_b & \to & b & \to & i_b & \to & b & \dots
\end{pmatrix}
$$

$$
t_2 = \begin{pmatrix}
x(tt) & \to & i_x & \to & x(ff) & \to & i_x & \to & x(ff) & \dots \\
 & & i_a & \to & a & \to & i_a & & & \dots \\
b & \to & i_b & \to & & & i_b & \to & b & \dots
\end{pmatrix}
$$

$$
t_2' = \begin{pmatrix}
x(tt) & \to & i_x & \to & x(ff) & \to & i_x & \to & x(ff) & \dots \\
 & & i_a & \to & & & i_a & \to & a & \dots \\
b & \to & i_b & \to & b & \to & i_b & & & \dots
\end{pmatrix}
$$

Note that $(t_1)_{\circ} = (t_2)_{\circ} = (t_2')_{\circ} = \boldsymbol{t}$. However, p_1 is deterministic, in that it admits only one synchronous trace given the inputs $\boldsymbol{t}$. In contrast, p_2 is not deterministic.

Example 15 (endochrony and mobility) *An illustration of the definition 7 in the presence of mobility can be given by considering the asynchronous trace* $\boldsymbol{t}$ *of messages incoming from* base_1 *and* base_2 *(examples 3 and 6)*

$$\boldsymbol{t} = \begin{pmatrix} t_1 & \to & t_1 & \dots \\ t_2 & \to & t_2 & \dots \\ s_1(t_2, s_2) & & & \dots \end{pmatrix}$$

$$t_1 = \begin{pmatrix} t_1 & \to & i_{t_1} & \dots \\ i_{t_2} & \to & i_{t_2} & \dots \\ i_{s_1} & \to & s_1(t_2, s_2) & \dots \end{pmatrix}$$

$$t_2 = \begin{pmatrix} i_{t_1} & \to & i_{t_1} & \dots \\ i_{t_2} & \to & t_2 & \dots \\ s_1(t_2, s_2) & \to & i_{s_1} & \dots \end{pmatrix}$$

It is easy to observe that the car *may non-deterministically choose to* talk *(synchronous trace* t_1*) or to switch and then* talk *(trace* t_2*), because the internal choice of awaiting* talk *or* switch *is not externally guarded by another message. This phenomenon can be aleviated by the analysis described in this section, yielding a corrected specification of the* car *(example 24, figure 22) and* base *processes which meets a deterministic interaction, bacause the* switch *message now guards every transition by the addition of a tag (*none *or* some *switching).*

$$\boldsymbol{t'} = \begin{pmatrix} t_1 & \to & t_1 & \dots \\ t_2 & \to & t_2 & \dots \\ s_1(\mathsf{none}) & \to & s_1(\mathsf{some}(t_2, s_2)) & \dots \end{pmatrix}$$

$$t' = \begin{pmatrix} t_1 & \to & i_{t_1} & & & \dots \\ i_{t_2} & \to & i_{t_2} & & & \dots \\ s_1(\mathsf{none}) & \to & i_{s_1} & \to & s_1(\mathsf{some}(t_2, s_2)) & \dots \end{pmatrix}$$

The property of isochrony is defined by the equivalence between the synchronous and asynchronous compositions of a pair of processes p and p'. It means that the synchronous design $p \,|\, p'$ supports the deployment $p \,\|\, p'$ over an asynchronous network without loss of semantics.

Definition 8 (isochrony) *p and p' are isochronous iff* $p \,|\, p' \simeq_\circ p \,\|\, p'$

Note that, if p and p' are isochronous, then $(\bullet^\omega(p \,|\, p'))_\circ = \circ^\omega(p \,\|\, p')$ by definition of the denotational semantics.

Example 16 (isochrony) *Let us consider a simple illustration of the property of isochrony by considering the composition of* f_a *with either* f_b^1 *or* f_b^2*. Whereas* f_b^1 *makes an implicit assumption on the presence/absence of* a *to decide whether to do* x *or* y*,* f_b^2 *checks the value of the guard* b*. As a result, synchronous and asynchronous composition coincide for*

f_a and f_b^2, but not for f_a and f_b^1. In f_b^1, the choice of y requires some knowledge on the absence of a. This information that is not available for asynchronous composition.

$$\begin{array}{lcl} f_a & = & 1{+}(a\,\|\,b(tt)){+}b(ff) \\ f_b^1 & = & 1{+}(a\,\|\,x){+}y \\ f_b^2 & = & 1{+}(b(tt)\,|\,x){+}(b(ff)\,|\,y) \end{array}$$

Example 17 (isochrony and mobility) *Similarly, consider the assembly of the p_{car} and p_{base_1} processes using synchronous composition (example 6) and asynchronous composition.*

$$t_{\mathsf{car}} = \left(\begin{array}{ccccc} t_1 & \to & i_{t_1} & \to & \dots \\ i_{s_1} & \to & s_1(t_2,s_2) & \to & \dots \end{array}\right)$$

$$t_{\mathsf{base}_{1,2}} = \left(\begin{array}{ccccccccc} i_{t_1} & \to & i_{t_1} & \to & i_{t_1} & \to & t_1 & \to & \dots \\ i_{t_2} & \to & t_2 & \to & i_{t_2} & \to & i_{t_2} & \to & \dots \\ s_1(t_2,s_2) & \to & i_{s_1} & \to & i_{s_1} & \to & i_{s_1} & \to & \dots \\ i_{s_2} & \to & i_{s_2} & \to & s_2(t_1,s_1) & \to & i_{s_2} & \to & \dots \end{array}\right)$$

Whereas the synchronous composition can rely on the internal observation of the simultaneous presence or absence of the messages t_1 and s_1 to merge the reactions $s_b\,|\,s_1$ (sharing t_1) and $s_c\,\|\,s_3$ (sharing s_1), the asynchronous composition cannot. Hence, the deployment of p_{car} and p_{base_1} denoted by $p_{\mathsf{car}}\,\|\,p_{\mathsf{base}_1}$ misses a semantical equivalence with $p_{\mathsf{car}}\,\|\,p_{\mathsf{base}_1}$, by allowing, e.g. *the car to await the switch s_1 while the* base *emits a talk t_1.*

$$(t_1)_\circ\,\|\,(t_2)_\circ = \left(\begin{array}{ccccccc} t_1 & \to & \dots & & s_1(t_2,s_2) & \to & \dots \\ t_2 & \to & \dots & & s_2(t_1,s_1) & \to & \dots \end{array}\right)$$

4.2. Hierarchic transition systems

In the aim of constructing decision procedures to check the formal properties of section 4.1, we establish the existence of a canonical *hierarchic normal form* for synchronous pre-order transition systems in the presence of mobility.

4.2.1. Disjunctive normal forms

The disjunctive normal form $\mathcal{D}_f$ of a SPOTS f is obtained by the function $\mathcal{D}[\![.]\!]$ (fig. 20).

$$\mathcal{D} \;::=\; \exists\tilde{x}.\forall\tilde{v}.(\mathsf{do}\,\mathcal{D}\,\mathsf{in}\,s) \mid \mathcal{D}{+}\mathcal{D}' \qquad \text{(disjunction)}$$

The recursive flattening function $\mathcal{D}[\![.]\!]$ is defined by induction on the structure of a family f. It consists of the replacement of synchronous compositions of messages in f by redundant combination or unions $s \cup s'$ of messages. We use an abstraction $\widehat{\bowtie}$ of the composability relation $\bowtie$ which checks the presence or absence of messages at given port names. The definability of $s\,\|\,s'$ from $s \cup s'$ is obtained by unifying (e.g. $u\sigma = u'\sigma$) variables carried by the redundant messages of s and s' (e.g. $x(u) \subseteq s$ and $x(u') \subseteq s'$) and by introducing guards (e.g. $\mathsf{a}(v = c)$) to implement the composability relation $\bowtie$ (e.g. $x(c)\,|\,x(v)$).

The construction of the disjunctive normal form makes use of the extensions of union and intersection of pre-orders to processes.

$$(\mathsf{do}\,f\,\mathsf{in}\,p) \cup (\mathsf{do}\,f'\,\mathsf{in}\,p') = \mathsf{do}\,f \mid f'\,\mathsf{in}\,p \cup p' \qquad (\mathsf{do}\,f\,\mathsf{in}\,p) \cap (\mathsf{do}\,f'\,\mathsf{in}\,p') = p \cap p'$$

$$
\begin{array}{rcl}
\mathcal{D}[\![f+f']\!] & = & \mathcal{D}[\![f]\!]+\mathcal{D}[\![f']\!] \\
\mathcal{D}[\![\forall v.f]\!] & = & \forall v.\mathcal{D}[\![f]\!] \\
\mathcal{D}[\![\exists x.f]\!] & = & \exists x.\mathcal{D}[\![f]\!] \\
\mathcal{D}[\![\mathsf{do}\ f\ \mathsf{in}\ p]\!] & = & \mathsf{do}\,\mathcal{D}[\![f]\!]\,\mathsf{in}\,\mathcal{D}[\![p]\!] \\
\mathcal{D}[\![f\,|\,f']\!] & = & 1+\sum_{\forall\tilde{v}.\exists\tilde{x}.\mathsf{do}\,\mathcal{D}\,\mathsf{in}\,s\sqsubseteq\mathcal{D}[\![f]\!]}^{\forall\tilde{w}.\exists\tilde{y}.\mathsf{do}\,\mathcal{D}'\,\mathsf{in}\,s'\sqsubseteq\mathcal{D}[\![f']\!]\,|\,s\widehat{\bowtie}s'} \forall\tilde{v},\tilde{w}.\exists\tilde{x},\tilde{y}.(\mathsf{do}\,\mathcal{D}[\![\mathcal{D}\,|\,\mathcal{D}']\!]\,\mathsf{in}\,\mathcal{U}[\![s,s']\!]) \\
\text{where } s\widehat{\bowtie}s' & \Leftrightarrow & (l\in\mathrm{fv}_{\mathcal{L}}(f)\cap\mathrm{fv}_{\mathcal{L}}(f')\Leftrightarrow(l\in\mathrm{fv}_{\mathcal{X}}(s)\Leftrightarrow l\in\mathrm{fv}_{\mathcal{X}}(s'))) \\
\mathcal{U}[\![s,s']\!] & = & \mathcal{G}[\![(s\,\|\,s')\sigma]\!] \text{ where } \forall l(v),l(v')\subseteq(s,s'),v\neq v'\Rightarrow\sigma\ni[v/v'] \\
 & & \qquad\qquad \forall l(v),l(u)\subseteq(s,s'),v\neq u\Rightarrow\sigma\ni[l(v)\,\|\,\mathsf{a}(v=u)/l(u)] \\
\mathcal{G}[\![s]\!] & = & s\,|\,(\prod_{v(u)\subseteq s}^{l(u)\subseteq s}\mathsf{a}(v=l\Rightarrow u=u'))
\end{array}
$$

Figure 20. Construction of the disjunctive normal form

We state the following property on the equivalence between a family f and its disjunctive normal form $\mathcal{D}_f$.

Property 4 *For all p, $p\simeq\mathcal{D}_p$*

Proof. Appendix F.

Example 18 (disjunction) *For instance, using flattening $\mathcal{D}[\![.]\!]$ and unification $\mathcal{U}[\![.]\!]$ yields the disjunctive normal form of the process* run1*, (similar to* run2*, in example 7). It is defined by a choice between two reactions s_a and s_b.*

$$
\begin{array}{rcl}
f_{\mathsf{run1}} & \simeq & \tau\ +\ \forall b.(\mathsf{run1}^-(b);\mathsf{run1}(\neg b)\ \ |\mathsf{tick}\ \ \ \ \ \ |\mathsf{a}(b=tt))\ \ \ \ \ \ :s_a \\
 & & +\ \forall b.(\mathsf{run1}^-(b);\mathsf{run1}(\neg b)\ \ |\mathsf{tick};\mathsf{out2}\ \ \|\mathsf{a}(b=ff))\ \ \ \ \ \ :s_b
\end{array}
$$

4.2.2. The hierarchic normal form

of a family f is obtained by restructuring f so as to recursively identify messages (e.g. p) upon which actions are triggered (e.g. T) and choices decided (e.g. $T+T'$). This amounts to representing f as a forest F of trees T.

Example 19 (hierarchy) *Let us reconsider the* Dnf *of the process* run1 *(example 18). The idea behind the notion of hierarchy is to do the opposite of the* Dnf*. Instead of enumerating every pattern of reaction, the construction of the* Hnf *proceeds in* factorizing *as much patterns as possible. For instance, in the process* run1*, we can easily observe that the pattern* $(\mathsf{run1}^-(b);\mathsf{run1}(\neg b)\,|\,\mathsf{tick})$ *is common to all reactions. We will call it the root of f_{modulo2}. Then given that root, the idea is to inductively decompose the rest of the specification. We have either $b=tt$ or* tick;out2 *when $b=ff$ (remember that we are considering pre-orders, hence* tick;out2 *"minus"* tick *should still mean* tick;out2*). This*

forms the hierarchic normal form of the process run1 *which will be represented using a factorization relation:*

$$f_{\mathsf{run1}} \simeq \forall b. \left(\begin{array}{c} \left(\mathsf{run1}^-(b);\mathsf{run1}(\neg b) \,\|\, \mathsf{tick} \right) \\ \triangledown \\ \left((\mathsf{a}(b = tt)) + (\mathsf{a}(b = ff) \,\|\, \mathsf{tick};\mathsf{out2}) \right) \end{array} \right)$$

The hierarchic normal form $\mathcal{H}_f$ of a family f consists of a disjunction of trees T, a so-called *forest* F. By construction, the nodes of a forest F are non-empty pre-orders s. By construction, all definitions $\mathsf{do}\, F \,\mathsf{in}\, s$ appear at the leaves of a hierarchy. For s a node of a tree, the relation *"tree T is child of s"* is written $s \triangleright T$. Similarly, for F a forest, we write $s \triangleright F$ provided that F is the disjunction of all trees T. We write $\mathring{T} = s$ for the root of a tree T of the form $s \triangleright F$ or s. We write $\partial T = F$ for the branches of a tree T of the form $s \triangleright F$.

$$F ::= T \mid F + F' \qquad \text{(forest)} \qquad T ::= \mathsf{do}\, F \,\mathsf{in}\, s \mid \exists \tilde{x}. \forall \tilde{v}. (s \triangleright F) \qquad \text{(tree)}$$

4.2.3. Factorization

The relation $s \triangleright F$ denotes a tree that share s and differ in all trees of F.

$$s \triangleright f = \sum_{p \sqsubseteq f} s \,|\, p \;\Leftrightarrow\; s = \bigcap_{p \sqsubseteq f} (s \cup p) \;\wedge\; \forall p \sqsubseteq f, p = \bigcap_{p' \subseteq (s \cup p)}^{s \cup p' = s \cup p} p'$$

It is defined by the structuration of families f into interiors $\mathring{f}$ and boundaries ∂f.

$$\mathring{p} = \bigcup_{p' \sqsubseteq f \,|\, p' \neq p} (p' \cap p) \qquad \partial p = \bigcap_{p' \subseteq p \,|\, \mathring{p} \cup p' = p} p'$$

4.2.4. Hierarchization

The hierarchization procedure is defined figure 21. Given the DNF $\mathcal{D}_f$ of a family f, it first consists of reconstructing the pre-conditions making control explicit, by recursively applying the pre-processing rules to every pre-order s of a DNF $\mathcal{D}$: constraints of the form $l(v) \,\|\, l'(v)$ are made explicit by $l(v) \,|\, l'(v') \,|\, \mathsf{a}(v = v'))$, variables v are related to locations as v_l and guards to sets of variables.

Example 20 (hierarchization) *In the example 19, the* HNF *of* run1 *was computed by determining the root* $\mathring{f}_{\mathsf{run1}} = (\mathsf{run1}^-(b);\mathsf{run1}(\mathsf{not}\, b)\ \mathsf{tick})$ *of* run1*, and then its two branches* $\partial s_a = \mathsf{a}(b = ff)$ *and* $\partial s_b = \mathsf{a}(b = tt) \,\|\, \mathsf{tick};\mathsf{out2}$ *(notice that the pre-order* tick;out2 *"minus"* tick *is not* out2*)*

Hierarchization $\mathcal{H}[\![.]\!]$ inductively applies to a family f and returns its hierarchic normal form $\mathcal{H}_f$ (after application of post-processing rules). The hierarchic normal form $\mathcal{H}_f$ of a family f satisfies:

Property 5 *For all p, $\mathcal{D}_p \simeq \mathcal{H}_p$*

PROOF. Appendix G

pre-processing	$\forall l(c) \in \min s$	$: \forall v.(s[l(v) \,\vert\, \mathsf{a}(v = c)/l(c)])$
	$\forall l(v), l'(v) \in \min s$	$: \forall w.(s[l'(w) \,\vert\, \mathsf{a}(v = w)/l'(v)])$
	$\forall l(v) \subseteq s \,\vert\, v \notin \mathrm{fv}_{\mathcal{V}}(\mathrm{pred}^*_{l(v)}(s)))$	$: s[v_l/v]$
	$\forall \mathsf{a}(e), \mathsf{a}(e') \subseteq s \,\vert\, \mathrm{fv}_{\mathcal{V}}(e) = \mathrm{fv}_{\mathcal{V}}(e')$	$: s[\mathsf{a}(e \wedge e')/\mathsf{a}(e), \mathsf{a}(e')]$
induction	$\mathcal{H}\left[\sum_{s \sqsubseteq f} s \triangleright F_s\right] = \sum_{s \sqsubseteq^\circ f} s \triangleright \left(\sum_{s' \sqsubseteq f}^{\circ_{s'=s}} \partial s' \triangleright F_{s'}\right)$	
post-processing	$s \triangleright s' = s \,\Vert\, s' \quad 1 \triangleright F = F \quad s \triangleright (s' \triangleright F) = (s \,\vert\, s') \triangleright F$	

Figure 21. Hierarchization of a family

4.3. Guarded hierarchies

We characterize the properties of endochrony and of isochrony in tems of the hierarchic representation of SPOTS. A family f is endochronous if the choice of a reaction p in f is determined by the values present at port along the successive nodes of $\mathcal{H}_f$ that form p and by the guards $\mathsf{a}(e)$ present at every branch of $\mathcal{H}_f$. We start by identifying the guards of a hierarchy F.

Definition 9 (guard) *s is* guarded *iff $\exists \mathsf{a}(e) \subseteq s$. If s is guarded, then $\mathsf{a}_s(e) = \wedge_{\mathsf{a}(e) \subseteq s} e$ guards s and $\mathsf{a}_T(e) = \mathsf{a}_s(e)$ guards $T = s \triangleright F$*

Definition 10 (well-guarded hierarchy)

1. *A hierarchy $1 + \exists \tilde{x}.\forall \tilde{v}.s \triangleright F$ is well-guarded iff*
   - *(a) F is well-guarded* w.r.t. *s;*
   - *(b) all leaves of $s \triangleright F$ are mutually guarded.*
2. *A forest $\sum_{i=1}^n s_i \triangleright F_i$ is* well-guarded w.r.t. *s iff*
   - *(a) the path s defines the guard $(\mathsf{a}_{s_i}(e_i))_{i=1}^n$* i.e. *$\forall i, \mathrm{fv}_{\mathcal{V}}(e_i) \subseteq \mathrm{fv}_{\mathcal{V}}(s)$*
   - *(b) the guards $(\mathsf{a}_{s_i}(e_i))_{i=1}^n$ are exclusive* i.e.

     $$\forall (i,j), \forall \mathrm{fv}_{\mathcal{V}}(e_i) \cup \mathrm{fv}_{\mathcal{V}}(e_j), ((e_i \wedge e_j) \Leftrightarrow i = j)$$

   - *(c) the forests F_i are well-guarded* w.r.t. *$s \,\vert\, s_i$, for all i.*
3. *A leaf $\mathsf{do}\, F \,\mathsf{in}\, s'$ of T is well-guarded* w.r.t. *s iff*
   - *(a) F and s' are well-guarded* w.r.t. *s;*
   - *(b) F and T are mutually-guarded.*

Property (1a) summarizes the criteria for checking a given hierarchy F well-guarded. Property (1b) requires all dynamic definitions of F to be mutually-guarded. Property (2a) states that the guard of every node s_i in the hierarchy should be defined by the variables accessible from the path s. Property (2b) requires all assertions $(e_i)_{i=1}^n$ to be mutually exclusive. This property reduces to a satisfaction problem when the variables $\mathrm{fv}_{\mathcal{V}}(e_i)_{i=1}^n$ are defined over finite domains (e.g. $\mathbb{B}$ or $\mathbb{Z}/n\mathbb{Z}, n < \omega$) and when the number of dynamically allocatable ports is finite (by considering, e.g., finite existential quantification, defined section 5). Otherwise, one may resource to abstract interpretation (as, e.g., in [6]), to statically check exclusion. Property (2c) and (3a) are induction hypothesis. Property (3b) requires the activation of every dynamic definition F of a hierarchy T to be guarded.

Example 21 (endochrony) *Let us recall the definition of f_{car} from our introductory example. In light of the above analysis, we easily observe that it is not endochronous: the choice of awaiting t or s is not guarded. Note that the* base *presents the same pathology.*

$$\mathcal{H}_{\mathsf{car}} \quad = \quad 1{+}\forall t, t', t, s'.\, \big(\mathsf{car}^{-}(t,s) \triangleright ((t;\mathsf{car}(t,s)){+}(s(t',s');\mathsf{car}(t',s')))\big)$$

The *isochrony* of a pair of well-guarded families f and f is checked by considering a criterion of mutual guardedness. This criterion consists of observing the assertions which guard the presence of messages along ports shared by the families f and f. We note P a path in a hierarchy and say that P is a path of $s \triangleright F$ (resp. $F{+}F'$) iff $P = s \,\|\, P'$ and P' is a path of F (resp. P is a path of either F or F'). We note $\mathsf{a}_P(e)$ the guard of a path P in a hierarchy.

Definition 11 (mutually guarded hierarchies) *The well-guarded hierarchies $(\mathcal{H}_i)_{i=1}^n$, of the form $1{+}\exists \tilde{x}_i.\forall \tilde{v}_i.T_i$, of leaves $(\mathsf{do}\, F_i^k \,\mathsf{in}\, s_i^k)_{k=1}^{n_i}$, are mutually guarded iff, for all $i = 1, n$ and $j \neq i$,*

1. *Let $(P_k)_{k=i}^j$ two paths of T_i and T_j of guards $\mathsf{a}_{P_k}(e_k)$. For all $k = i, j$ and $k' \neq k$*

   (a) $\forall l \in \mathrm{fv}_{\mathcal{L}}(T_i) \cap \mathrm{fv}_{\mathcal{L}}(T_j)$,

   $$l(u) \subseteq P_k \wedge (\forall u', l(u') \notin P_{k'}) \Rightarrow \forall \mathrm{fv}_{\mathcal{V}}(e_k) \cup \mathrm{fv}_{\mathcal{V}}(e_{k'}), \neg(e_k \wedge e_{k'})$$

   (b) $\forall v(u) \in P_k, \forall l \in \mathrm{fv}_{\mathcal{L}}(T_{k'}) \backslash \mathrm{fv}_{\mathcal{L}}(P_{k'}), \forall \mathrm{fv}_{\mathcal{V}}(e_k) \cup \mathrm{fv}_{\mathcal{V}}(e_{k'}), \neg((v = l) \wedge e_k \wedge e_{k'})$

2. *$F_i^{k_i}$ and $F_j^{k_j}$ are mutually guarded.*

3. *$F_i^{k_i}$ and T_j are mutually guarded.*

Example 22 (mutually-guarded hierarchies) *For instance, reconsider the example 16. The hierarchies of f_a and f_b are not mutually-guarded because the simultaneous presence of a in f_a (guard $\mathsf{a}(v_b = tt)$) and its absence in f_b (not guarded* i.e. *$\mathsf{a}(tt)$) are not exclusive. Conversally, the hierarchies of f_a and f_c are mutually guarded.*

4.4. Formal property

From the definition of well-guarded and mutually-guarded hierarchies, we obtain means for checking the properties of endochrony and isochrony.

Theorem 4 (endochrony) *If $\mathcal{H}_f$ is well-guarded then f is* endochronous.

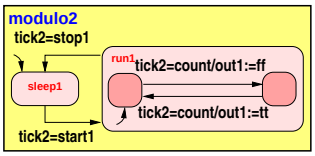
PROOF. Appendix H.

Note that relaxing the property (3a) of definition 10 as "F is well-guarded" and "s' is well-guarded w.r.t. s" preserves the isochrony of the process p i.e. for all $m > 0$, $p \bullet\!\!\rightarrow^n \mathsf{do}\,(\prod_{i=1}^n f_i)\,\mathsf{in}\,t$ yields isochronous $(f_i)_{i=1}^n$. Also note that the property (1b) allows to a implement a dynamically created process f (e.g. $\mathsf{do}\,f_0\,\mathsf{in}\,(\mathsf{do}\,f\,\mathsf{in}\,t)$) as an asynchronous thread (i.e. $(\mathsf{do}\,f_0\,\mathsf{in}\,t)\ \|\ (\mathsf{do}\,f\,\mathsf{in}\,t)$ by ensuring that $f \,|\, f_0 \simeq_\circ f \,\|\, f_0$).

Example 23 (enforcing endochrony) *We are now equipped with an powerful instrumentation to diagnose the pathology outlined in the example 4. One thing that can be easily observed is that the process* modulo2 *does not have an endochronous* SPOTS, *because the* HNF *of f_{modulo2} is simply not a tree: the events* start1, stop1 *of are not synchronized* w.r.t. tick. *It is hence a forest of the form:*

$$\begin{aligned} f_{\mathsf{modulo2}} = 1 + \mathcal{H}_{\mathsf{run1}} \quad &+ \quad (\mathsf{start1} \,|\, \mathsf{modulo2}^-(\mathsf{sleep1});\mathsf{modulo2}(\mathsf{run1})) \\ &+ \quad (\mathsf{stop1} \,\|\, \mathsf{modulo2}^-(\mathsf{run1});\mathsf{modulo2}(\mathsf{sleep1})) \end{aligned}$$

A solution consists of introducing an activation message tick2 *which is sent to* modulo2 *in order to tell it whether to* start, count *or* stop.



The tick2 *message defines the root of f'_{modulo2} and each transition can be hierarchized with respect to a guard on the value carried by that message.*

$$\begin{aligned} f'_{\mathsf{modulo2}} = \ 1 + &(\mathsf{modulo2}^-(\mathsf{sleep1});\mathsf{modulo2}(\mathsf{run1}) \ \ |\mathsf{tick2}(\mathsf{start1})) \\ + &(\mathsf{modulo2}^-(\mathsf{run1}) \ \ ;\mathsf{modulo2}(\mathsf{sleep1}) \,|\mathsf{tick2}(\mathsf{stop1})) \\ + &(\mathsf{modulo2}^-(\mathsf{run1}) \ \ ;\mathsf{modulo2}(\mathsf{run1}) \ \ |\mathsf{tick2}(\mathsf{count}) \,|\, f_{\mathsf{run1}}) \end{aligned}$$

Theorem 5 (isochrony) *If $\mathcal{H}_f$ and $\mathcal{H}_{f'}$ are mutually-guarded then f and f' are isochronous.*

PROOF. Appendix I.

Example 24 (endochrony and mobility) *An endochronous $\mathcal{H}'_{\mathsf{car}}$ can systematically be obtained (figure 22) by,* e.g., *letting the port* switch *be of* option *type (*i.e. some(talk, switch) *or* none*). The non-deterministic choice between* talk *and* switch *is then guarded by the* option *type tag.*

$$\mathcal{H}'_{\mathsf{car}} \;=\; 1{+}\forall t,t',s,s',v_s.\; \left(\begin{array}{c} (\mathsf{car}^-(t,s) \parallel s(v_s)) \\ \triangledown \\ \left(\left(\begin{array}{ll} & \mathsf{a}(v_s=\mathsf{none}) \\ \parallel & t;\mathsf{car}(t,s) \end{array}\right) + \left(\begin{array}{ll} & \mathsf{a}(v_s=\mathsf{some}(t',s')) \\ \parallel & s(v_s);\mathsf{car}(t',s') \end{array}\right)\right) \end{array}\right)$$

$$\mathcal{H}'_{\mathsf{base}_i} \;=\; 1{+}\forall t,s,v_i,v'_i.\; \left(\begin{array}{c} \mathsf{base}_i{}^-(v_i) \\ \triangledown \\ \left(\begin{array}{c} \left(\begin{array}{ll} & \mathsf{a}(v_i=\mathit{ff}) \\ \mid & \mathsf{alert}_i;\mathsf{base}_i(\mathit{tt}) \end{array}\right) \\ + \left(\begin{array}{c} \left(\begin{array}{ll} & \mathsf{a}(v_i=\mathit{tt}) \\ \parallel & \mathsf{give}_i(v'_i);\mathsf{switch}_i(v'_i) \end{array}\right) \\ \triangledown \\ \left(\begin{array}{c} \left(\begin{array}{ll} & \mathsf{a}(v'_i=\mathsf{none}) \\ \parallel & \mathsf{talk}_i;\mathsf{base}_i(\mathit{tt}) \end{array}\right) \\ + \left(\begin{array}{ll} & \mathsf{a}(v'_i=\mathsf{some}(t',s')) \\ \parallel & \mathsf{base}_i(\mathit{ff}) \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right)$$

Figure 22. Hierarchization of the mobile telephone specification

Example 25 (isochrony and dynamicity) *Let us finally reconsider the connection between the* car *and its* base *in the introductory example (figure 22). From the above analysis, the synchronous specification of the* system *can safely be deployed on an asynchronous environment. The endochronous specifications* f'_{car} *and* f'_{base} *both meet the criterion of endochrony and they are isochronous:*

$$\mathcal{H}'_{\mathsf{car}} \parallel \mathcal{H}'_{\mathsf{base}} \simeq_\circ \mathcal{H}'_{\mathsf{car}} \parallel \mathcal{H}'_{\mathsf{base}}$$

Had we used the process clone *of example 11 to let the* system *dynamically allocate bases (and cars), we would then have obtained an endochronous system, thanks to the analysis of definition 10 (property 3b). This definition ensures that a dynamically allocated* base *can be activated as a separate (asynchronously composed) thread in the* system*, without any loss of semantics.*

5. Discussion

On the trade-off between mobility and dynamicity

As demonstrated by the examples 11 and 25, the mobility of port names incurs dynamic allocation and the model of SPOTS naturally accommodate with this feature, thanks to the generality of the properties of endochrony and of isochrony (theorems 4 and 5).

We shall however not conclude before an address on the trade-off between mobility and dynamicity in the presence of finite resources. Reconsidering the original example of the mobile telephone of [18], it is indeed a striking fact that its very first aim is to demonstrate mobility, not dynamicity. Hence, a discrimination of either features deserves insight.

The concern of determining resource bounds for asynchronous process calculi (and more specifically the π-calculus) has been the subject of recent work [20], in which finite-state systems can be isolated by program analysis. Such techniques could be adapted to SPOTS.

We opine that an alternative approach deserves a proposal by observing that, in the design process of an embedded system, non-functional space requirements are usually imposed before the design of the system even starts. For instance, the memory-size of a mobile phone has serious impacts on the cost of its production, and it is often regarded as a given. Hence, a pragmatic approach to the design of resource-bound embedded systems usually consists of imposing bounds to dynamicity (without actually loosing the benefits of mobility), the so-called process of *dimensioning* the system. Limitations to the dynamic creation of names can be specified by considering a finite restriction of existential quantification $\exists x \in \tilde{x}.f$ with $|\tilde{x}| < \omega$ (figure 23). This form implements the dimensioning of a system by allowing to choose a reaction $p \sqsubseteq f$ iff a fresh $x \in \tilde{x}$ can be allocated. It makes it possible to pre-allocate every $f_i = f[x_i/x], i \in [1, n]$ and then to activate one of them each time the definition is invoked (pre-allocation is implemented by replacing the boxed term by $(\prod_{i=1}^{n}(\mathsf{a}(n \geq i) \,|\, f_i))$).

$$\exists x \in (x_i)_{i=1}^{n}.f = \exists c.\left(\mathsf{do}\ \forall n.\left(\begin{array}{l}(c^{-}(n);c(n-1)) \\ \parallel \left(\mathsf{a}(n \leq 0) + \boxed{\mathsf{a}(n > 0) \,|\, \exists x.f}\right)\end{array}\right)\ \mathsf{in}\ c(n)\right)$$

Figure 23. Finite existential quantification to scale dynamic allocation

In this setting, the problem of checking for bound-resources reduces to devicing a program analysis (e.g. abstract interpretation techniques) in order to determine whether the system may invoke a definition more than a suited number of times.

6. Conclusion

In conclusion, we have introduced a calculus allowing the synchronous modeling and the asynchronous deployment of mobile system specifications. We have shown that the expressivity of this calculus compares to the JOIN-calculus (asynchronous communication), to the FUSION-calculus (synchronous communication), and to synchronous stream functions. The elementary structure of this calculus consists of pre-ordered transition systems. We have shown that the technique of hierarchization, introduced in [27], seamlessly generalizes in the presence of mobility, providing formal connections between the synchronous and asynchronous models of concurrency, in the presence of mobility.

Example 26 (conclusion) *To summarize the results related to the introductory* STATECHART *example, we have solved the issues related to correct deployment: we now have at our disposal the concepts and algorithms for equipping* SPOTS *with the hybrid synchronous/asynchronous semantics suited to deployment, while preserving semantic equivalence with the original, fully synchronous, specification.*

The STATECHART *of figure 24 summarizes the construction of the solution developed along all the examples. Its typical message sequence (on the right), outlines the intuition behind the proposition that the composition of now meet the equivalence* $\mathcal{H}'_{\mathsf{arbiter}} \| \mathcal{H}'_{\mathsf{modulo2}} = \mathcal{H}'_{\mathsf{arbiter}} \| \mathcal{H}'_{\mathsf{modulo2}}$. *Each interaction in the system is correctly synchronized.*

Furthermore, and unlike the brute-force distribution of the synchronous semantics depicted in the third message sequence of example 1, the property of isochrony alleviates the need for a global synchronization of, e.g., modulo4 *with the rest of the system.*

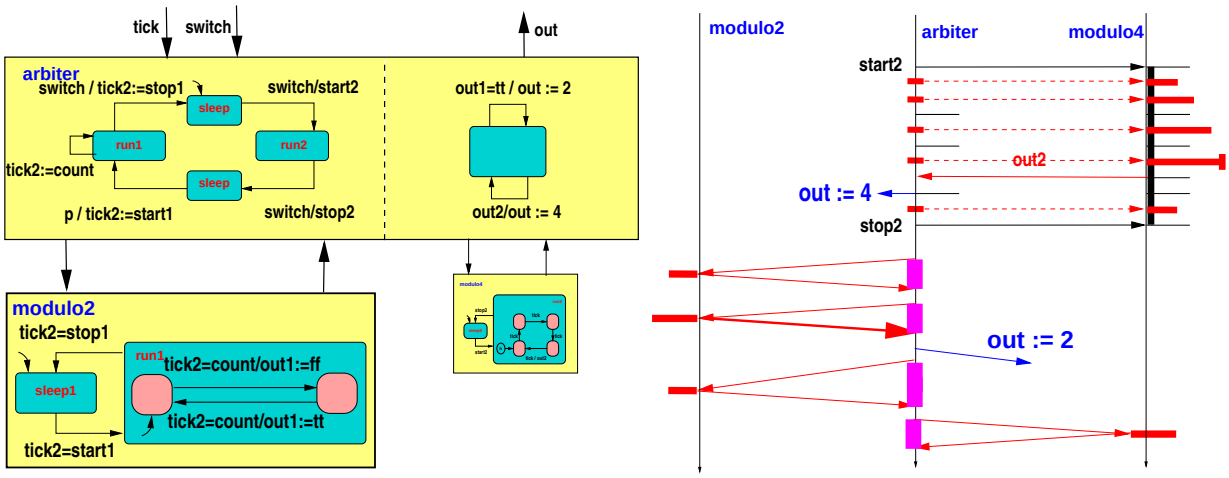


Figure 24. Hierarchization of the arbiter specification

Acknowledgments

The author is indepted to Albert Benveniste for his support and insightful suggestions and wishes to thank Mickael Kerboeuf for his careful proof-reading and valuable comments.

REFERENCES

1. S. Abramsky., S. J. Gay, R. Nagarajan. Interaction Categories and the Foundations of Typed Concurrent Programming. In *Deductive Program Design Proceedings of the 1994 Marktoberdorf Summer School.* Nato Asi Series, Springer-Verlag, 1995.
2. A. Benveniste, P. Le Guernic, C. Jacquemot. Synchronous programming with events and relations: the Signal language and its semantics. In *Science of Computer Programming*, v. 16, 1991.
3. A. Benveniste, G. Berry. Real-time systems design and programming. Another look at real-time programming, special section of In *Proceedings of the* Ieee, v. 79, n. 9. Ieee, September 1991.
4. A. Benveniste, B. Caillaud, P. Le Guernic. From synchrony to asynchrony. *International Conference on Concurrency Theory.* Lectures Notes in Computer Science. Springer Verlag, 1999.
5. G. Berry, G. Gonthier. The Esterel synchronous programming language: design, semantics, implementation. In *Science of Computer Programming*, v. 19, 1992.
6. F. Besson, T. Jensen, J.-P. Talpin. Timed polyhedra analysis for synchronous languages. In *Static Analysis Symposium.* Lectures Notes in Computer Science. Springer Verlag, September 1999.
7. P. Caspi. Clocks in dataflow languages. In *Theoretical Computer Science*, v. 94. Elsevier, 1992.
8. P. Caspi, M. Pouzet. Synchronous Kahn networks. In *International Conference on Functional Programming.* Acm Press, 1996.
9. C. Fournet, G. Gonthier. The reflexive chemical abstract machine and the join-calculus. In *International Symposium on Principles of Programming Languages.* Acm Press, 1996.
10. C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget and D. Rémy. A calculus of mobile agents. In *International Conference on Concurrency Theory.* Springer Verlag, 1996.
11. N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud. The synchronous data-flow programming language Lustre. In *Proceedings of the* Ieee, v. 79(9). Ieee, 1991.
12. N. Halbwachs. Synchronous programming of reactive systems. Kluwer Academic Publishing, 1993.
13. K. Honda, N. Yoshida. On Reduction-Based Process Semantics. In *Theoretical Computer Science*, v. 152(2). Elsevier, 1995
14. D. Harel, A. Naamad. *The* Statemate *semantics of* Statecharts. I-Logix, 1995.
15. C. Lavene, B. Victor. Solos in concert. In *International Colloquium on Automata, Languages, and Programming.* Lectures Notes in Computer Science. Springer Verlag, 1999.
16. R. Milner. Calculi for synchrony and asynchrony. In *Theoretical Computer Science.* Elsevier, 1983.
17. R. Milner. Calculi for interaction. In *Acta Informatica.* Springer Verlag, 1996.
18. R. Milner, J. Parrow, D. Walker. A calculus of mobile processes. In *Information and Computation.* Academic Press, 1992.

19. R. MILNER, D. SANGIORGI. Barbed bisimulation. In *International Colloquium on Automata, Languages and Programming*. Lectures Notes in Computer Science. Springer Verlag, , 1992.
20. U. MONTANARI, M. PISTORE. Finite State Verification for the Asynchronous Pi-Calculus. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Lectures Notes in Computer Science. Springer Verlag, 1999.
21. D. NOWAK, J.-P. TALPIN, P. LE GUERNIC. Synchronous structures. *International Conference on Concurrency Theory*. Lectures Notes in Computer Science. Springer Verlag, 1999.
22. S.-O. NYSTRÖM, B. JONSSON. A fully abstract semantics for concurrent constraint programming. In *Information and Computation*, v. 146. Academic Press, 1998.
23. J. PARROW, B. VICTOR. The FUSION calculus: expressiveness and symmetry in mobile computing. In *Symposium on Logic in Computer Science*. IEEE Press, 1998.
24. J. RUMBAUGH, I. JACOBSON, G. BOOCH. *The unified modeling language reference manual*. Addison-Wesley object technology series, 1999.
25. V. SARASWAT, R. JAGADEESAN, V. GUPTA. Timed default concurrent constraint programming. In *Journal of Symbolic Computation*. Academic Press, 1999.
26. J.-P. TALPIN, A. BENVENISTE, B. CAILLAUD, C. JARD, Z. BOUZIANE, H. CANON. BDL, a language of distributed reactive objects. *International Symposium on Object-Oriented Real-Time Distributed Computing*. IEEE Press, 1998.
27. J.-P. Talpin, A. Benveniste, P. Le Guernic. Asynchronous deployment of synchronous transition systems. *Technical report n. 1269*. IRISA, October 1999.

A. Proof of theorem 1

The proof proceeds by making some observations on the structure of the translation.

Lemma 1 $P \leftrightharpoons^* P' \Leftrightarrow \circ([\![P]\!]_{\mathcal{J}}) = \circ([\![P']\!]_{\mathcal{J}})$

Proof of lemma 1 *Let P and P' be terms of the grammar of processes which are equivalent for the transitive closure $\leftrightharpoons^*$ of the relation $\leftrightharpoons$. Possible re-arrangements $P \leftrightharpoons^* P'$ (modulo alpha-renaming of bound names) are*

$$
\begin{aligned}
P \mid 0 &\leftrightharpoons^* P && (1)\\
\mathsf{def}\, D \mid 1 \,\mathsf{in}\, P &\leftrightharpoons^* \mathsf{def}\, D \,\mathsf{in}\, P && (2)\\
P_1 \mid P_2 &\leftrightharpoons^* P_2 \mid P_1 && (3)\\
(P_1 \mid P_2) \mid P_3 &\leftrightharpoons^* P_1 \mid (P_2 \mid P_3) && (4)\\
\mathsf{def}\, D_1 \mid D_2 \,\mathsf{in}\, P &\leftrightharpoons^* \mathsf{def}\, D_2 \mid D_1 \,\mathsf{in}\, P && (5)\\
\mathsf{def}\, (D_1 \mid D_2) \mid D_3 \,\mathsf{in}\, P &\leftrightharpoons^* \mathsf{def}\, D_1 \mid (D_2 \mid D_3) \,\mathsf{in}\, P && (6)\\
\mathsf{def}\, D_1 \,\mathsf{in}\, \mathsf{def}\, D_2 \,\mathsf{in}\, P &\leftrightharpoons^* \mathsf{def}\, D_1 \mid D_2 \,\mathsf{in}\, P && (7)\\
\mathsf{def}\, D \,\mathsf{in}\, P &\leftrightharpoons^* \mathsf{def}\, D \,\mathsf{in}\, P' \ (P \leftrightharpoons^* P') && (8)
\end{aligned}
$$

(1-6) yield $[\![P]\!]_{\mathcal{J}} = [\![P']\!]_{\mathcal{J}}$, hence $\circ([\![P]\!]_{\mathcal{J}}) = \circ([\![P']\!]_{\mathcal{J}})$ by definition of $\circ$.

(7) is by definition of the transition relation $\circ(.)$,

$$
\begin{aligned}
&\circ([\![\mathsf{def}\, D_1 \,\mathsf{in}\, \mathsf{def}\, D_2 \,\mathsf{in}\, P]\!]_{\mathcal{J}})\\
&\quad = \circ(\mathsf{do}\, [\![D_1]\!]_{\mathcal{J}} \,\mathsf{in}\, \mathsf{do}\, [\![D_2]\!]_{\mathcal{J}} \,\mathsf{in}\, [\![P]\!]_{\mathcal{J}}) && \text{by definition of the translation}\\
&\quad = \circ(\mathsf{do}\, (1{+}[\![D_1]\!]_{\mathcal{J}}) \mid (1{+}[\![D_2]\!]_{\mathcal{J}}) \,\mathsf{in}\, [\![P]\!]_{\mathcal{J}}) && \text{by definition of the relation } \circ(.)\\
&\quad = \circ(\mathsf{do}\, (1{+}[\![D_1]\!]_{\mathcal{J}}){+}[\![D_2]\!]_{\mathcal{J}}) \,\mathsf{in}\, [\![P]\!]_{\mathcal{J}}) && \text{since } [\![D_1]\!]_{\mathcal{J}} \text{ and } [\![D_2]\!]_{\mathcal{J}} \text{ interleave } {}^{(a)}\\
&\quad = \circ([\![\mathsf{def}\, D_1 \mid D_2 \,\mathsf{in}\, P]\!]_{\mathcal{J}}) && \text{by definition of the translation}
\end{aligned}
$$

(a) By construction and for $i = 1, 2$ and with $D_i = (|_{j=1}^{n_i} J_i^j \rhd P_i^j)$, the encoding of D_i is of the form

$$[\![D_i]\!]_{\mathcal{J}} = \sum_{j=1}^{n_i} ([\![J_i^j]\!]_{\mathcal{J}} \,\|\, [\![P_i^j]\!]_{\mathcal{J}})$$

However, $\forall j, \mathrm{dv}(J_1^j) \cap \mathrm{dv}(J_2^j) = \emptyset$, by definition of the JOIN*-calculus. Furthermore, the encoding of each P_i^j is of the form*

$$[\![P_i^j]\!]_{\mathcal{J}} = \mathsf{do}\, f_i^j \,\mathsf{in}\, s_i^j$$

where each s_i^j consists of private scheduling ports $\kappa \in K$, which can only be triggered one after the other. Hence, the transitions of $1{+}[\![D_1]\!]_{\mathcal{J}}) \,\|\, (1{+}[\![D_2]\!]_{\mathcal{J}})$ and $\circ(1{+}[\![D_1]\!]_{\mathcal{J}}{+}[\![D_2]\!]_{\mathcal{J}}$ correspond:

$$\circ(1{+}[\![D_1]\!]_{\mathcal{J}}) \mid (1{+}[\![D_2]\!]_{\mathcal{J}}) = \circ(1{+}[\![D_1]\!]_{\mathcal{J}}{+}[\![D_2]\!]_{\mathcal{J}})$$

(8) is proved by induction. From the premise $P \leftrightharpoons^* P'$ *follows the hypothesis* $\circ([\![P]\!]_{\mathcal{J}}) = \circ([\![P']\!]_{\mathcal{J}})$ *and, by definition of the transition relation* $\circ(.)$*, the result that*

$$\circ([\![\mathsf{def}\ D\ \mathsf{in}\ P]\!]_{\mathcal{J}}) = \circ([\![\mathsf{def}\ D\ \mathsf{in}\ P']\!]_{\mathcal{J}})$$

Lemma 2 $P \rightharpoonup P' \Leftrightarrow [\![P]\!]_{\mathcal{J}} \circ\!\!\rightarrow^* \mathsf{do}\ f\ \mathsf{in}\ \boldsymbol{t} \wedge [\![P']\!]_{\mathcal{J}} = \mathsf{do}\ f + f'\ \mathsf{in}\ \max \boldsymbol{t}$

We write $\max \boldsymbol{t}$ for $\prod_{l(u)\in \max \boldsymbol{t}} l(u)$.

Proof of lemma 2 *The structural equivalence relation* $\leftrightharpoons^*$ *implies that every process* P *admits a representation* $\mathsf{def}\ D\ \mathsf{in}\ J$ *and allows us to reformulate rule (red) as:*

$$\begin{array}{rcl} \mathsf{def}\ (D_0 \mid (J \rhd P))\ \mathsf{in}\ (J_0 \mid (J\sigma)) & \rightharpoonup & \mathsf{def}\ (D_0 \mid (J \rhd P))\ \mathsf{in}\ (J_0 \mid (P\sigma)) \\ & \leftrightharpoons^* & \mathsf{def}\ (D_0 \mid D' \mid (J \rhd P))\ \mathsf{in}\ (J_0 \mid J') \\ \textit{where}\ (P\sigma) & \leftrightharpoons^* & \mathsf{def}\ D'\ \mathsf{in}\ J' \end{array} \tag{9}$$

From (9) it is hence sufficient to prove that

$$\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma \quad \rightharpoonup\!\!\leftrightharpoons^* \quad \mathsf{def}\ D'\sigma \mid J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J'\sigma \tag{10}$$

iff

$$\begin{array}{rcl} [\![\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma]\!]_{\mathcal{J}} & \circ\!\!\rightarrow^* & \mathsf{do}\ f\ \mathsf{in}\ \boldsymbol{t} \\ [\![\mathsf{def}\ D'\sigma \mid J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J'\sigma]\!]_{\mathcal{J}} & = & \mathsf{do}\ f + f'\ \mathsf{in}\ \max \boldsymbol{t} \end{array}$$

Let $J\sigma = |_{i=1}^{n} x_i\langle u_i\rangle$*. From (10) and by definition of the translation,*

$$[\![J\sigma]\!]_{\mathcal{J}} = \exists \kappa^{i=1,n}.(\mathsf{do}\,(1 \sum^{i=1,n} (\kappa^{i^-}(tt);\kappa^i(ff) \mid \langle\!\langle x_i\langle u_i\rangle\rangle\!\rangle))\ \mathsf{in}\ \kappa \mid (\prod_{i=1}^{n} \kappa^i(tt))) \tag{11}$$

$$\langle\!\langle x_i\langle u_i\rangle\rangle\!\rangle = x_i^-(\mathsf{some}(u_i));x_i(\mathsf{none}) \tag{12}$$

From (11-12) and by definition of $\circ(.)$*, there exists*

$$\mathsf{do}\ f\ \mathsf{in}\ \boldsymbol{t}_0 \in \circ^n([\![\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma]\!]_{\mathcal{J}}) \tag{13}$$

such that

$$x_i(\mathsf{some}(u_i)) \in \max \boldsymbol{t}_0, \forall i = 1, n \tag{14}$$

From (10) and by definition of the translation,

$$[\![J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')]\!]_{\mathcal{J}} = \exists \kappa.\forall (u_i)_{i=1}^{n}.(\prod_{i=1}^{n} x_i^-(\mathsf{some}(u_i));x_i(\mathsf{none}) \parallel [\![\mathsf{def}\ D'\ \mathsf{in}\ J']\!]_{\mathcal{J}}) \tag{15}$$

From (14-15) and by definition of $\bullet^?$*,*

$$[\![J\sigma\ as\ J]\!]_{\mathcal{J}}\ \bullet^?\ [\![J\sigma\ as\ P]\!]_{\mathcal{J}} \tag{16}$$

From (16) and (13-14) and by definition of $\circ(.)$,

$$\mathsf{do}\ f\ \mathsf{in}\ (\boldsymbol{t}_0 \circ ([\![J\sigma \rhd (\mathsf{def}\ D'\sigma\ \mathsf{in}\ J'\sigma)]\!]_{\mathcal{J}})) \quad \in \quad \circ^{n+1}([\![\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma]\!]_{\mathcal{J}}) \tag{17}$$

From (17) and the equivalence rule for definitions

$$\mathsf{do}\ f\ \mathsf{in}\ \mathsf{do}\ 1{+}[\![D'\sigma]\!]_{\mathcal{J}}\ \mathsf{in}\ \boldsymbol{t}_0 \circ ([\![J\sigma]\!]_{\mathcal{J}} \,|\, [\![J'\sigma]\!]_{\mathcal{J}}) \quad \in \quad \circ^{n+1}[\![\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma]\!]_{\mathcal{J}} \tag{18}$$

From (18) and by the lemma 1

$$\circ(\mathsf{do}\ f{+}[\![D'\sigma]\!]_{\mathcal{J}}\ \mathsf{in}\ \boldsymbol{t}_0 \circ ([\![J\sigma]\!]_{\mathcal{J}} \,|\, [\![J'\sigma]\!]_{\mathcal{J}})) \quad \subseteq \quad \circ^{n+2}[\![\mathsf{def}\ J \rhd (\mathsf{def}\ D'\ \mathsf{in}\ J')\ \mathsf{in}\ J\sigma]\!]_{\mathcal{J}} \tag{19}$$

$$\tag{20}$$

which proves the lemma with $\boldsymbol{t} = \boldsymbol{t}_0 \circ ([\![J\sigma]\!]_{\mathcal{J}} \,|\, [\![J'\sigma]\!]_{\mathcal{J}})$.

Theorem 1 $P \approx_{\mathcal{J}} P' \Leftrightarrow [\![P]\!]_{\mathcal{J}} \approx_{\circ} [\![P']\!]_{\mathcal{J}}$

Proof of theorem 1 *From lemma 1*

$$P \rightleftharpoons^* P' \Leftrightarrow \circ([\![P]\!]_{\mathcal{J}}) = \circ([\![P']\!]_{\mathcal{J}}) \tag{21}$$

From the lemma 2,

$$P \mapsto Q \,|\, x\langle u\rangle \quad \Leftrightarrow \quad [\![P]\!]_{\mathcal{J}} \circ\!\!\rightarrow^* \mathsf{do}\ f\ \mathsf{in}\ \boldsymbol{t} \wedge [\![Q \,|\, x\langle u\rangle]\!]_{\mathcal{J}} = \mathsf{do}\ f{+}f'\ \mathsf{in}\ \max \boldsymbol{t} \tag{22}$$

From (22) and by definition of asynchronous observation

$$[\![Q \,|\, x\langle u\rangle]\!]_{\mathcal{J}} \Downarrow x(\mathsf{some}(u)) \forall x \notin K, \forall u \in \mathcal{D}_x \tag{23}$$

From (23) and by definition of weak barbed observation

$$P \Downarrow x(u) \Leftrightarrow [\![P]\!]_{\mathcal{J}} \Downarrow x(\mathsf{some}(u)), \forall x \notin K, \forall u \in \mathcal{D}_x \tag{24}$$

By definition of the weak-barbed bisimulation $\approx_{\mathcal{J}}$ *(the observational congruence [10]),*

$$\begin{array}{lll} P \approx_{\mathcal{J}} Q & \Leftrightarrow & P \Downarrow x \Rightarrow Q \Downarrow x \\ & \wedge & P \rightarrow^* P' \Rightarrow (Q \rightarrow^* Q' \wedge P' \approx_{\mathcal{J}} Q') \\ & \wedge & \mathsf{def}\ D\ \mathsf{in}\ P \approx_{\mathcal{J}} \mathsf{def}\ D\ \mathsf{in}\ Q \\ & \wedge & P \,|\, R \approx_{\mathcal{J}} Q \,|\, R \end{array}$$

and by definition of asynchronous bisimulation $\approx_{\circ}$,

$$\boldsymbol{r}_1 \mathcal{R}_{\circ} \boldsymbol{r}_2 \Rightarrow ((\boldsymbol{r}_1 \circ\!\!\rightarrow \boldsymbol{r}_1' \Rightarrow (\boldsymbol{r}_2 \circ\!\!\rightarrow^* \boldsymbol{r}_2', \boldsymbol{r}_1' \mathcal{R}_{\circ} \boldsymbol{r}_2')) \wedge (\boldsymbol{r}_1 \Downarrow x(u) \Rightarrow \boldsymbol{r}_2 \Downarrow x(u)))$$

full-abstraction of $[\![\_]\!]_{\mathcal{J}}$ *follows from (21) and (24),*

$$P \approx_{\mathcal{J}} P' \Leftrightarrow [\![P]\!]_{\mathcal{J}} \approx_{\circ} [\![P']\!]_{\mathcal{J}}$$

B. Proof of theorem 2

The proof consists of lemmas 3, 4 and 5

Lemma 3 $P \equiv Q \Leftrightarrow \circ[\![P]\!]^s_{\mathcal{F}} = \circ[\![Q]\!]^s_{\mathcal{F}}, \forall s$

Proof of lemma 3 *The structural equivalence relation* $\equiv$ *is defined by*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl} (v)0 & \equiv & 0 & [u=u]P & \equiv & P & P \mid (v)Q & \equiv & (v)(P \mid Q) \\ (v)(w)P & \equiv & (w)(v)P & (v)MP & \equiv & M(v)P & !P & \equiv & P \mid !P \end{array}$$

The left-hand side term of each axiom yields the following translation

$$\begin{array}{llll} [\![(v)0]\!]^s_{\mathcal{F}} & = [\![\forall v.1]\!]^s_{\mathcal{F}} & = [\![1]\!]^s_{\mathcal{F}} & = [\![0]\!]^s_{\mathcal{F}} \\ [\![(v)(w)P]\!]^s_{\mathcal{F}} & = \forall v,w.[\![P]\!]^s_{\mathcal{F}} & = \forall w,v.[\![P]\!]^s_{\mathcal{F}} & = [\![(w)(v)P]\!]^s_{\mathcal{F}} \\ [\![(v)MP]\!]^s_{\mathcal{F}} & = \forall v.[\![P]\!]^{\mathsf{a}(M)\mid s}_{\mathcal{F}} & = [\![(v)P]\!]^{\mathsf{a}(M)\mid s}_{\mathcal{F}} & = [\![M(v)P]\!]^s_{\mathcal{F}} \quad (v \notin \mathrm{fv}(M)) \\ [\![P \mid (v)Q]\!]^s_{\mathcal{F}} & = [\![P]\!]_{\mathcal{F}} \mid \forall v.[\![Q]\!]^s_{\mathcal{F}} & = \forall v.([\![P]\!]^s_{\mathcal{F}} \mid [\![Q]\!]^s_{\mathcal{F}}) & = [\![(v)(P \mid Q)]\!]^s_{\mathcal{F}} \quad (v \notin \mathrm{fv}(P)) \end{array}$$

$$\begin{array}{llll} \circ[\![[u=u]P]\!]^s_{\mathcal{F}} & = \circ[\![P]\!]^{\mathsf{a}(u=u)\mid s}_{\mathcal{F}} & = \circ[\![P]\!]^{\mathsf{a}(\mathit{tt})\mid s}_{\mathcal{F}} & = \circ[\![P]\!]^s_{\mathcal{F}} \\ \circ[\![!P]\!]^s_{\mathcal{F}} & = \circ(\mathsf{do}\,1{+}[\![P]\!]^s_{\mathcal{F}}\,\mathsf{in}\,[\![P]\!]^s_{\mathcal{F}}) & = \circ[\![P \mid !P]\!]^s_{\mathcal{F}} & \end{array}$$

which proves the lemma.

Lemma 4 $P \to P' \mid x(u) \Rightarrow \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota})$

Proof of lemma 4 *The proof is by induction on the derivation tree. The reduction rules* $\to$ *are defined by*

$$\frac{P \to P'}{P \mid Q \to P' \mid Q} \quad (a) \qquad \frac{P \to P'}{(v)P \to (v)P'} \quad (c)$$

$$\frac{P \equiv Q \quad Q \to Q' \quad Q' \equiv P'}{P \to P'} \quad (b) \qquad (\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P) \to P\sigma \quad (d)$$

(a) By hypothesis

$$\frac{P \to P' \mid x(u)}{P \mid Q \to P' \mid x(u) \mid Q} \tag{25}$$

By induction hypothesis on P

$$P \to P' \mid x(u) \Rightarrow \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota}) \tag{26}$$

By definition of the translation

$$[\![P \mid Q]\!]^s_{\mathcal{F}} \;=\; [\![P]\!]^s_{\mathcal{F}} \mid [\![Q]\!]^s_{\mathcal{F}} \tag{27}$$

From (27-26), since $[\![Q]\!]_{\mathcal{F}}$ *can be silent*

$$\exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \mid [\![Q]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota}) \tag{28}$$

(b) By hypothesis,

$$\frac{P \equiv Q \qquad Q \to Q' \mid x(u) \qquad Q' \equiv P'}{P \to P' \mid x(u)} \tag{29}$$

By induction hypothesis on Q

$$\exists!(\iota, \overline{t}), [\![Q]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{30}$$

From (29) and by the lemma 3,

$$\circ[\![P]\!]_{\mathcal{F}} = \circ[\![Q]\!]_{\mathcal{F}} \wedge \circ[\![P']\!]_{\mathcal{F}} = \circ[\![Q']\!]_{\mathcal{F}} \tag{31}$$

From (30-31) the result

$$\exists!(\iota, \overline{t}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{32}$$

(c) By hypothesis

$$\frac{P \to P'}{(v)P \to (v)P'} \tag{33}$$

By induction hypothesis on P,

$$\exists!(\iota, \overline{t}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{34}$$

By definition of the translation

$$[\![(v)P]\!]_{\mathcal{F}} = \forall v.[\![P]\!]_{\mathcal{F}} \tag{35}$$

From (34) and by definition of $\circ(.)$,

$$\exists!(\iota, \overline{t}), \forall v.[\![P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{36}$$

Hence the result

$$\exists!(\iota, \overline{t}), [\![(v)P']\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{37}$$

(d) By hypothesis,

$$(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P) \to P\sigma \equiv P' \mid y(t) \tag{38}$$

where σ satisfies $u\sigma = u'\sigma, M\sigma, M'\sigma, \mathrm{im}\,\sigma \cap \tilde{v} = \emptyset, \mathrm{dom}\,\sigma = \tilde{v}$. By definition of the translation,

$$\begin{array}{l} [\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} = \\ \quad = \forall \tilde{v}. \left(\begin{array}{ll} & \exists \kappa.(\mathsf{do}\,1{+}(\exists \iota.\forall \overline{\iota}.x(u,\iota,\overline{\iota}) \parallel \mathsf{a}(M) \mid \kappa^-(\sharp t);\kappa(ff))\,\mathsf{in}\,\kappa(\sharp t)) \\ \mid & \exists \kappa.(\mathsf{do}\,1{+}(\forall \iota.\exists \overline{\iota}.x(u',\iota,\overline{\iota}) \parallel \mathsf{a}(M') \mid \kappa^-(\sharp t);\kappa(ff))\,\mathsf{in}\,\kappa(\sharp t)) \\ \mid & [\![P]\!]_{\mathcal{F}} \end{array} \right) \end{array} \tag{39}$$

Choosing σ and fresh κ_1, κ_2 yields the transition

$$\begin{array}{lll} \circ[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} & \ni & \mathsf{do}\,1{+}(\exists \iota.\forall \overline{\iota}..x(u,\iota,\overline{\iota}) \mid \kappa_1^-(\sharp t);\kappa_1(ff))\,\mathsf{in}\,\kappa_1(\sharp t) \\ & \mid & \mathsf{do}\,1{+}(\forall \iota.\exists \overline{\iota}.x(u,\iota,\overline{\iota}) \parallel \kappa_2^-(\sharp t);\kappa_2(ff))\,\mathsf{in}\,\kappa_2(\sharp t) \\ & \mid & [\![P\sigma]\!]_{\mathcal{F}} \end{array} \tag{40}$$

Choosing fresh $\iota, \overline{\iota}$ yields the transition

$$\circ^2[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} \ni x(u,\iota,\overline{\iota}) \mid [\![P\sigma]\!]_{\mathcal{F}} \tag{41}$$

From (38), (41) and by lemma 3

$$\circ^3[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} \ni [\![P' \mid y(t)]\!]_{\mathcal{F}} \tag{42}$$

From (42) and by definition of the translation

$$\exists!(\iota,\overline{\iota}), [\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} \Downarrow y(t,\iota,\overline{\iota})$$

Lemma 5 $\forall x \notin K, \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota}) \Rightarrow P \to^* P' \mid x(u)$

Proof of lemma 5 *By lemma 3, the proof reduces to a case analysis on reducible terms of the form (a-d)*

(a) By hypothesis

$$\exists!(\iota,\overline{\iota}), [\![P \mid Q]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota}) \tag{43}$$

From (43) and by definition of the translation

$$\exists!(\iota,\overline{\iota}), ([\![P]\!]_{\mathcal{F}} \parallel [\![Q]\!]_{\mathcal{F}}) \Downarrow x(u,\iota,\overline{\iota}) \tag{44}$$

From (44), assuming that $[\![Q]\!]_{\mathcal{F}}$ is silent (we would be in case (d) otherwise) and by induction hypothesis

$$P \to^* P' \mid x(u) \tag{45}$$

From (45) and rule (a)

$$P \mid Q \to P' \mid x(u) \mid Q$$

(b) By hypothesis

$$\exists!(\iota, \overline{t}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \wedge P \equiv Q \tag{46}$$

From (46) and by lemma 3

$$\circ[\![P]\!]_{\mathcal{F}} = \circ[\![Q]\!]_{\mathcal{F}} \tag{47}$$

From (46), (47) and by definition of $\Downarrow$

$$\exists!(\iota, \overline{t}), [\![Q]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{48}$$

From (48) and by induction hypothesis on Q

$$Q \rightarrow^* Q' \mid x(u) \tag{49}$$

From (49) and by rule (b)

$$P \rightarrow^* Q' \mid x(u)$$

(c) By hypothesis

$$\exists!(\iota, \overline{t}), [\![(v)P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{50}$$

From (50) and by definition of the translation

$$\exists!(\iota, \overline{t}), (\forall v.[\![P]\!]_{\mathcal{F}}) \Downarrow x(u, \iota, \overline{t}) \tag{51}$$

From (51) and assuming that observing $x(u)$ *does not require binding* v *(otherwise, we would be in case (d)),*

$$\exists!(\iota, \overline{t}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u, \iota, \overline{t}) \tag{52}$$

From (52) and by induction hypothesis on P

$$P \rightarrow^* P' \mid x(u) \tag{53}$$

From (53) and by rule (c)

$$(v)P \rightarrow^* (v)P' \mid x(u)$$

(d) By hypothesis

$$\exists!(\iota,\overline{\iota}), [\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} \Downarrow y(t,\iota_y,\overline{\iota_y}) \tag{54}$$

where $x \neq y$ *and* σ *satisfies* $u\sigma = u'\sigma, M\sigma, M'\sigma, \mathrm{im}\,\sigma \cap \tilde{v} = \emptyset, \mathrm{dom}\,\sigma = \tilde{v}$. *By definition of the translation,*

$$[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} = \tag{55}$$
$$= \forall\tilde{v}.\left(\begin{array}{ll} & \exists\kappa.(\mathsf{do}\,1{+}(\exists\iota.\forall\overline{\iota}.x(u,\iota,\overline{\iota}) \parallel \mathsf{a}(M) \mid \kappa^-(\mathit{tt});\kappa(\mathit{ff}))\,\mathsf{in}\,\kappa(\mathit{tt})) \\ \mid & \exists\kappa.(\mathsf{do}\,1{+}(\forall\iota.\exists\overline{\iota}.x(u',\iota,\overline{\iota}) \parallel \mathsf{a}(M') \mid \kappa^-(\mathit{tt});\kappa(\mathit{ff}))\,\mathsf{in}\,\kappa(\mathit{tt})) \\ \mid & [\![P]\!]_{\mathcal{F}} \end{array}\right)$$

Choosing σ *and fresh* κ_1, κ_2 *yields the transition*

$$\begin{array}{rcl} \circ[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} & \ni & \mathsf{do}\,1{+}(\exists\iota.\forall\overline{\iota}.x(u,\iota,\overline{\iota}) \parallel \kappa_1^-(\mathit{tt});\kappa_1(\mathit{ff}))\,\mathsf{in}\,\kappa_1(\mathit{tt}) \\ & \mid & \mathsf{do}\,1{+}(\forall\iota.\exists\overline{\iota}.x(u,\iota,\overline{\iota}) \parallel \kappa_2^-(\mathit{tt});\kappa_2(\mathit{ff}))\,\mathsf{in}\,\kappa_2(\mathit{tt}) \\ & \mid & [\![P\sigma]\!]_{\mathcal{F}} \end{array} \tag{56}$$

Choosing fresh $\iota, \overline{\iota}$ *yields the transition*

$$\circ^2[\![(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P)]\!]_{\mathcal{F}} \ni x(u,\iota,\overline{\iota}) \mid [\![P\sigma]\!]_{\mathcal{F}} \tag{57}$$

From (54), (57) and by definition of $\Downarrow$,

$$\exists!(\iota,\overline{\iota}), [\![P\sigma]\!]_{\mathcal{F}} \Downarrow y(t,\iota_y,\overline{\iota_y}) \tag{58}$$

From (58) and by induction hypothesis

$$P\sigma \rightarrow^* P' \mid y(t) \tag{59}$$

From (54) and (59)

$$(\tilde{v})(Mx(u) \mid M'\overline{x}(u') \mid P) \rightarrow^* P' \mid y(t)$$

Theorem 2 $\forall x \notin K, \forall u \in \mathcal{D}_x, P \Downarrow x(u) \Leftrightarrow \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota})$

Proof of theorem 2 *From the lemma 3, 4 and 5, it follows*

$\forall x \notin K, \forall u \in \mathcal{D}_x, P \Downarrow x(u) \Leftrightarrow \exists!(\iota,\overline{\iota}), [\![P]\!]_{\mathcal{F}} \Downarrow x(u,\iota,\overline{\iota})$

which proves the theorem.

$$E, x \to^{r} x' \vdash x \to^{r} x'$$

$$E \vdash \mathsf{const}\, v \to^{v} \mathsf{const}\, v$$

$$E \vdash \mathsf{const}\, v \to^{\mathsf{nil}} \mathsf{const}\, v$$

$$\frac{E \vdash S_1 \to^{v_1} S_1' \quad E \vdash S_2 \to^{v_2} S_2'}{E \vdash \mathsf{extend}\, S_1\, S_2 \to^{v_1(v_2)} \mathsf{extend}\, S_1'\, S_2'}$$

$$\frac{E \vdash S_1 \to^{\mathsf{nil}} S_1' \quad E \vdash S_2 \to^{\mathsf{nil}} S_2'}{E \vdash \mathsf{extend}\, S_1\, S_2 \to^{\mathsf{nil}} \mathsf{extend}\, S_1'\, S_2'}$$

$$\frac{E \vdash S_1 \to^{tt} S_1' \quad E \vdash S_2 \to^{v_2} S_2' \quad E \vdash S_3 \to^{\mathsf{nil}} S_3'}{E \vdash \mathsf{merge}\, S_1\, S_2\, S_3 \to^{v_2} \mathsf{merge}\, S_1'\, S_2'\, S_3'}$$

$$\frac{E \vdash S_1 \to^{ff} S_1' \quad E \vdash S_2 \to^{\mathsf{nil}} S_2' \quad E \vdash S_3 \to^{v_3} S_3'}{E \vdash \mathsf{merge}\, S_1\, S_2\, S_3 \to^{v_3} \mathsf{merge}\, S_1'\, S_2'\, S_3'}$$

$$\frac{E \vdash S_1 \to^{\mathsf{nil}} S_1' \quad E \vdash S_2 \to^{\mathsf{nil}} S_2' \quad E \vdash S_3 \to^{\mathsf{nil}} S_3'}{E \vdash \mathsf{merge}\, S_1\, S_2\, S_3 \to^{\mathsf{nil}} \mathsf{merge}\, S_1'\, S_2'\, S_3'}$$

$$\frac{E \vdash S_1 \to^{\mathsf{nil}} S_1' \quad E \vdash S_2 \to^{\mathsf{nil}} S_2'}{E \vdash \mathsf{when}\, S_1\, S_2 \to^{\mathsf{nil}} \mathsf{when}\, S_1'\, S_2'}$$

$$\frac{E \vdash S_1 \to^{tt} S_1' \quad E \vdash S_2 \to^{v_2} S_2'}{E \vdash \mathsf{when}\, S_1\, S_2 \to^{v_2} \mathsf{when}\, S_1'\, S_2'}$$

$$\frac{E \vdash S_1 \to^{ff} S_1' \quad E \vdash S_2 \to^{v_2} S_2'}{E \vdash \mathsf{when}\, S_1\, S_2 \to^{\mathsf{nil}} \mathsf{when}\, S_1'\, S_2'}$$

$$\frac{E \vdash S_2 \to^{\mathsf{nil}} S_2'}{E \vdash \mathsf{pre}\, v_1\, S_2 \to^{\mathsf{nil}} \mathsf{pre}\, v_1\, S_2'}$$

$$\frac{E \vdash S_2 \to^{v_2} S_2'}{E \vdash \mathsf{pre}\, v_1\, S_2 \to^{v_1} \mathsf{pre}\, v_2\, S_2'}$$

$$\frac{E, x \to^{x_v} x' \vdash S \to^{v} S'}{E \vdash \lambda x.S \to^{\lambda x_v.v} \lambda x_v.\lambda x'.S'}$$

$$\frac{E \vdash S \to^{v} S' \quad (x \text{ scalar}^*)}{E \vdash \lambda x.S \to^{\lambda x.v} \lambda x.S'}$$

$$\frac{E \vdash S_1 \to^{v_1} S_1' \quad E \vdash S_2 \to^{v_2} S_2'}{E \vdash S_1(S_2) \to^{v_1(v_2)} (S_1'(v_2))(S_2')}$$

$$\frac{E \vdash S_1 \to^{v_1} S_1' \quad (S_2 \text{ scalar})}{E \vdash S_1(S_2) \to^{v_1(S_2)} S_1'(S_2)}$$

$$\frac{E, x \to^{x_v} x' \vdash S \to^{v} S'}{E \vdash \mathsf{rec}\, x.S \to^{\mathsf{rec}\, x_v.v} \mathsf{rec}\, x'.S'[\mathsf{rec}\, x_v.v/x_v]}$$

Figure 25. Operational semantics of synchronous stream functions [8]

C. Proof of theorem 3

The proof is a mechanical induction on the structure of the derivation tree. For each case analysis, the proof of the theorem is decomposed into the lemma 6 and 7.

Lemma 6 *(6a)* $S \rightarrow^{v} S' \Rightarrow (\llbracket S \rrbracket_\kappa)\downarrow(\kappa(x) \,|\, x(u)) \wedge (\llbracket v \rrbracket_\kappa)\downarrow\kappa(u))$
(6b) $S \rightarrow^{\mathsf{nil}} S' \Rightarrow (\exists r \in \bullet(\llbracket S \rrbracket_\kappa), \kappa(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r))$

Proof of lemma 6 *We detail the case analysis for expressions of the form* $\mathsf{rec}\, x.S$, $\lambda x.S$, $S(S')$ *and* $\mathsf{pre}\, S\, S'$. *The analysis of the cases of* merge, when *expressions use the same proof technique than that for* pre*; the case of* extend *combines the techniques for* pre *and* $S(S')$*; the case of* const *that of* rec *and the case of* x *is trivial.*

1. **Case of** $\mathsf{rec}\, x.S$

   By hypothesis,

$$\frac{E, x \rightarrow^{x_v} x' \vdash S \rightarrow^{v} S'}{E \vdash \mathsf{rec}\, x.S \rightarrow^{\mathsf{rec}\, x_v.v} \mathsf{rec}\, x'.S'[\mathsf{rec}\, x_v.v/x_v]} \tag{60}$$

   From (60) and by induction hypothesis

$$\exists x, u, (\llbracket S \rrbracket_\kappa)\downarrow(\kappa(x) \,|\, x(u)) \wedge (\llbracket v \rrbracket_\kappa)\downarrow\kappa(u)) \tag{61}$$

   From (61) and by definition of the translation

$$\exists x, x_v, u, (\llbracket \mathsf{rec}\, x_v.v \rrbracket_\kappa)\downarrow(\kappa(x_v) \,|\, x_v(u)) \wedge (\llbracket \mathsf{rec}\, x.S \rrbracket_\kappa)\downarrow(\kappa(x) \,|\, x(u))$$

2. **Case of** $\lambda x.S$ *By hypothesis,*

$$\frac{E, x \rightarrow^{x_v} x' \vdash S \rightarrow^{v} S'}{E \vdash \lambda x.S \rightarrow^{\lambda x_v.v} \lambda x_v.\lambda x'.S'} \qquad \frac{E \vdash S \rightarrow^{v} S'}{E \vdash \lambda x.S \rightarrow^{\lambda x.v} \lambda x.S'} \quad (x \ scalar) \tag{62}$$

   From (62) and by induction hypothesis

$$\exists x, u, (\llbracket S \rrbracket_\kappa)\downarrow(\kappa(x) \,|\, x(u)) \wedge (\llbracket v \rrbracket_\kappa)\downarrow\kappa(u)) \tag{63}$$

   From (63) and by definition of the translation

$$\exists f, g, h, \llbracket \lambda x.v \rrbracket_\kappa\downarrow\kappa(f) \wedge \llbracket \lambda x_v.v \rrbracket_\kappa\downarrow\kappa(g) \wedge \llbracket \lambda x.S \rrbracket_\kappa\downarrow\kappa(h)$$

3. **Case of** $S(S')$ *By hypothesis,*

$$\frac{E \vdash S_1 \rightarrow^{v_1} S_1' \quad E \vdash S_2 \rightarrow^{v_2} S_2'}{E \vdash S_1(S_2) \rightarrow^{v_1(v_2)} (S_1'(v_2))(S_2')} \qquad \frac{E \vdash S_1 \rightarrow^{v_1} S_1'}{E \vdash S_1(S_2) \rightarrow^{v_1(S_2)} S_1'(S_2)} \quad (S_2 \ scalar) \tag{64}$$

From (64) and by induction hypothesis on $S_{1,2}$

$$\forall i = 1,2, [\![S_i]\!]_{\kappa_i}\downarrow\kappa_i(u_i) \wedge [\![v_i]\!]_{\kappa'_i}\downarrow\kappa'_i(u_i) \tag{65}$$

By hypothesis, $v_1 = \lambda x.v'_1$ *and* $v = v1(v_2) = v'_1[v_2/x]$. *In particular, from (65) and by definition of the translation,*

$$\begin{array}{rcl} [\![v_1]\!]_{\kappa'_1} & = & \forall x, \kappa''.(u_1(x,\kappa'') \,|\, [\![v'_1]\!]_{\kappa''}) \,|\, \kappa(u_1) \\ [\![S_1(S_2)]\!]_{\kappa} & = & [\![S_1]\!]_{\kappa_1} \,|\, [\![S_2]\!]_{\kappa_2} \,|\, (\kappa_1(u_1);u_1(u_2,\kappa) \leftarrow \kappa_2(u_2)) \\ [\![v_1(v_2)]\!]_{\kappa} & = & [\![v_1]\!]_{\kappa_1} \,|\, [\![v_2]\!]_{\kappa_2} \,|\, (\kappa_1(u_1);v_1(u_2,\kappa) \leftarrow \kappa_2(u_2)) \end{array} \tag{66}$$

From (66) and the definition of observation,

$$[\![S_1(S_2)]\!]_{\kappa}\downarrow\kappa(u) \wedge [\![v_1(v_2)]\!]_{\kappa}\downarrow\kappa(u)$$

4. **Case of** $\mathsf{pre}\, S\, S'$ *By hypothesis,*

$$\frac{E \vdash S_2 \rightarrow^{\mathsf{nil}} S'_2}{E \vdash \mathsf{pre}\, v_1\, S_2 \rightarrow^{\mathsf{nil}} \mathsf{pre}\, v_1\, S'_2}\ (a) \qquad \frac{E \vdash S_2 \rightarrow^{v_2} S'_2}{E \vdash \mathsf{pre}\, v_1\, S_2 \rightarrow^{v_1} \mathsf{pre}\, v_2\, S'_2}\ (b) \tag{67}$$

By definition of the translation

$$[\![\mathsf{pre}\, v_1\, S_2]\!]_{\kappa} = \exists x, zx, \kappa_2. \tag{68}$$

$$([\![S_2]\!]_{\kappa_2} \,|\, \overbrace{(\mathsf{do}\,1 + (\langle\!\langle\kappa_2\rangle\!\rangle^{\mathsf{nil}} \,|\, \langle\!\langle\kappa\rangle\!\rangle_x^{\mathsf{nil}}) + \forall v, w.(\langle\!\langle\kappa_2\rangle\!\rangle^{v} \,|\, zx^-(w);zx(v) \,|\, \langle\!\langle\kappa\rangle\!\rangle_x^{w})\mathsf{in}\, [\![v_1]\!]_{zx})}^{p})$$

From (67a) and by induction hypothesis on S_2

$$\exists r \in \bullet([\![S_2]\!]_{\kappa_2}), \kappa_2(y) \subseteq r \wedge (\forall u, y(u) \not\subseteq r) \tag{69}$$

By definition of the translation (68), since $\kappa_2(y)$ *is observable but not* $y(u)$ *(69), the only choice of* p *is to trigger* $p' = \langle\!\langle\kappa_2\rangle\!\rangle^{\mathsf{nil}} \,|\, \langle\!\langle\kappa\rangle\!\rangle_x^{\mathsf{nil}}$ *to match* $\kappa_2(y)$ *present and* y *absent. Since that* p' *is concatenable to* $[\![S_1]\!]_{zx}$ *(because it does not use* zx*), we have*

$$\exists r \in \bullet([\![\mathsf{pre}\, v_1\, S_2]\!]_{\kappa}), \kappa(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r)$$

From (67b) and by induction hypothesis on S_2

$$[\![S_2]\!]_{\kappa_2})\downarrow(\kappa_2(x_2) \,|\, x(u_2)) \wedge ([\![v_2]\!]_{\kappa'_2})\downarrow\kappa'_2(u_2)) \tag{70}$$

From (67b) and by induction hypothesis on v_1

$$[\![v_1]\!]_{\kappa_1}\downarrow\kappa_1(u_1) \tag{71}$$

By definition of the translation (68), since $\kappa_2(x_2)$ *and* $x_2(u_2)$ *are now observable (69),* p *must now trigger* $p'' = \langle\!\langle\kappa_2\rangle\!\rangle^{u_2} \,|\, zx^-(u_1);zx(u_2) \,|\, \langle\!\langle\kappa\rangle\!\rangle_x^{u^1}$ *to match* $\kappa_2(x_2)$ *and* $x_2(u_2)$*. Since that* p'' *is concatenable to* $[\![v_1]\!]_{zx}$ *and from (71), we obtain*

$$[\![\mathsf{pre}\, v_1\, S_2]\!]_{\kappa})\downarrow(\kappa(x) \,|\, x(u_1)) \wedge ([\![v_1]\!]_{\kappa'})\downarrow\kappa'(u_1))$$

Lemma 7
(7a) $(\llbracket S \rrbracket_\kappa)\downarrow(\kappa(x) \,|\, x(u)) \wedge (\llbracket v \rrbracket_\kappa)\downarrow\kappa(u)) \Rightarrow S \rightarrow^{v} S'$
(7b) $\exists r \in \bullet(\llbracket S \rrbracket_\kappa), \kappa(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r) \Rightarrow S \rightarrow^{\mathsf{nil}} S'$

Proof of lemma 7 *The proof method is identical to that of lemma 6. We detail the case analysis for* rec *stream definition and for delay* pre.

1. **Case of** $\mathsf{rec}\, x.S$ *By hypothesis,*

$$\llbracket \mathsf{rec}\, x_v.v \rrbracket_\kappa\downarrow(\kappa(x_v) \,|\, x_v(u)) \wedge \llbracket \mathsf{rec}\, x.S \rrbracket_\kappa\downarrow(\kappa(x) \,|\, x(u)) \tag{72}$$

From (72), this requires that

$$\llbracket S \rrbracket_\kappa\downarrow(\kappa(x) \,|\, x(u)) \wedge \llbracket v \rrbracket_\kappa\downarrow\kappa(u)) \tag{73}$$

From (73) and by induction hypothesis: $S \rightarrow^{v} S'$. *With fresh* x_v *and* x' *in* $x \rightarrow^{x_v} x'$ *and by definition of the relation* $\rightarrow -$,

$$\mathsf{rec}\, x.S \rightarrow^{\mathsf{rec}\, x_v.v} \mathsf{rec}\, x'.S'[\mathsf{rec}\, x_v.v/x_v]$$

2. **Case of** $\mathsf{pre}\, v_1\, S_2$ *By definition of the translation,*

$$\begin{array}{l} \llbracket \mathsf{pre}\, v_1\, S_2 \rrbracket_\kappa = \\ \quad \exists x, zx, \kappa_2.(\llbracket S_2 \rrbracket_{\kappa_2} \,\|\, (\mathsf{do}\, 1 \;\; +(\langle\!\langle \kappa_2 \rangle\!\rangle^{\mathsf{nil}} \,|\, \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}}) \\ \qquad\qquad\qquad\quad +\forall v, w.(\langle\!\langle \kappa_2 \rangle\!\rangle^{v} \,\|\, zx^{-}(w);zx(v) \,|\, \langle\!\langle \kappa \rangle\!\rangle_x^{w})\mathsf{in}\, \llbracket v_1 \rrbracket_{zx})) \end{array} \tag{74}$$

There are two cases (75) and (76)

$$\exists r \in \bullet(\llbracket \mathsf{pre}\, v_1\, S_2 \rrbracket_\kappa), \kappa(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r) \tag{75}$$

$$\llbracket \mathsf{pre}\, v_1\, S_2 \rrbracket_\kappa\downarrow(\kappa(x) \,|\, x(u_1)) \wedge (\llbracket v_1 \rrbracket_{\kappa'})\downarrow\kappa'(u_1)) \tag{76}$$

From (75) and by definition of the translation, $(\langle\!\langle \kappa_2 \rangle\!\rangle^{\mathsf{nil}} \,|\, \langle\!\langle \kappa \rangle\!\rangle_x^{\mathsf{nil}})$ *is the only possible transition since* x *was not observed. Hence,*

$$\exists r \in \bullet(\llbracket S_2 \rrbracket_{\kappa_2}), \kappa_2(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r) \tag{77}$$

By induction hypothesis on S_2, $S_2 \rightarrow^{\mathsf{nil}} S_2'$ *and, by definition of the relation* $\rightarrow -$,

$$\mathsf{pre}\, v_1\, S_2 \rightarrow^{\mathsf{nil}} \mathsf{pre}\, v_1\, S_2'$$

From (76) and by definition of the translation, the only possible transition is $\langle\!\langle \kappa_2 \rangle\!\rangle^{u_2} \,|\, zx^{-}(u_1);zx(u_2) \,\|\, \langle\!\langle \kappa \rangle\!\rangle_x^{u_1})$ *since* x *was observed. Hence,*

$$\llbracket S_2 \rrbracket_{\kappa_2}\downarrow(\kappa_2(x_2) \,\|\, x_2(u_2)) \wedge (\llbracket v_2 \rrbracket_{\kappa_2'})\downarrow\kappa_2'(u_2)) \tag{78}$$

From (78) and by induction hypothesis on S_2, $S_2 \rightarrow^{v_2} S_2'$ *and by definition of the relation* $\rightarrow -$,

$$\mathsf{pre}\, v_1\, S_2 \rightarrow^{v_1} \mathsf{pre}\, v_2\, S_2'$$

Theorem 3 *(3a)* $S \rightarrow^v S' \Leftrightarrow ([\![S]\!]_\kappa)\downarrow(\kappa(x)\,|\,x(u)) \wedge ([\![v]\!]_\kappa)\downarrow\kappa(u))$
(3b) $S \rightarrow^{\mathsf{nil}} S' \Leftrightarrow (\exists r \in \bullet([\![S]\!]_\kappa), \kappa(x) \subseteq r \wedge (\forall u, x(u) \not\subseteq r))$

Proof of theorem 3 *Lemma 6 and 7 prove the theorem 3. In addition, one may note that the translation preserves the transitions implemented by the relation* $\rightarrow -$*, which means namely that, if* $S \rightarrow^v S'$ *then the transitions* $\bullet([\![S']\!]_\kappa)$ *are contained in* $\bullet^2([\![S]\!]_\kappa)$*.*

D. Proof of property 1

Property 1 $\forall p, p',\ (p = p') \Rightarrow (p \simeq p')$.

Proof of property 1 *By hypothesis, let* p *and* p' *be such that* $p = p'$*. By induction on the structure of* p*, either* $p = s = p'$ *(hence* $p \simeq p'$ *because no transition is possible) or* $p = \mathsf{do}\, f\, \mathsf{in}\, s$ (resp. $p' = \mathsf{do}\, f'\, \mathsf{in}\, s$) *and* $f = f'$*. By definition of (set-theoretical) equality, we then have that* $p'' \sqsubseteq f$ *iff* $p'' \sqsubseteq f'$*. Hence, by definition of concatenation,* $r \in \bullet p$ *iff* $r \in \bullet p'$*. Thus, by definition,* $p \simeq p'$*.*

E. Proof of properties 2 and 3

Property 2 *Synchronous bisimulation* $\approx$ *is a congruence:*

(a) $\approx$ *is stable under substitution,* i.e. $r_1 \approx r_2 \Rightarrow (r_1[u'/u] \approx r_2[u'/u])$

(b) $\approx$ *is an equivalence relation*

(c) for all r, r_1, r_2 *and* f*,* $r_1 \approx r_2$ *implies*

(1) $r_1 \| r \approx r_2 \| r$

(2) $\exists x.r_1 \approx \exists x.r_2$

(3) $\mathsf{do}\, f\, \mathsf{in}\, r_1 \approx \mathsf{do}\, f\, \mathsf{in}\, r_2$

Proof of property 2 *The proof uses a representation of the properties* $(a, \ldots c3)$ *in terms of the inductive relations* $(\mathcal{R}_i)_{i\in\{a,b,c_1,c_2,c_3\}}$ *defined by*

$$\forall i \in \{a, b, c_1, c_2, c_3\}, \mathcal{R}_i \ = \ \mathrm{lfp}\mathcal{F}_i \approx$$

as a result of the monotonic operators $(\mathcal{F}_i)_{i\in\{a,b,c_1,c_2,c_3\}}$ *and defined by*

$$\begin{aligned}
\mathcal{F}_a &= \lambda\mathcal{R}.(\mathcal{R} \cup \{(r_1[u'/u], r_2[u'/u]) \,|\, r_1\mathcal{R}r_2\}) \\
\mathcal{F}_b &= \lambda\mathcal{R}.(\mathcal{R} \cup \{(r_1, r_2) \,|\, r_1\mathcal{R}r \wedge r\mathcal{R}r_2\}) \\
\mathcal{F}_{c_1} &= \lambda\mathcal{R}.(\mathcal{R} \cup \{(r_1 \| r, r_2 \| r) \,|\, r_1\mathcal{R}r_2\}) \\
\mathcal{F}_{c_2} &= \lambda\mathcal{R}.(\mathcal{R} \cup \{(\exists x.r_1, \exists x.r_2) \,|\, r_1\mathcal{R}r_2\}) \\
\mathcal{F}_{c_3} &= \lambda\mathcal{R}.(\mathcal{R} \cup \{(\mathsf{do}\, f\, \mathsf{in}\, r_1, \mathcal{R}\mathsf{do}\, f\, \mathsf{in}\, r_2) \,|\, r_1\mathcal{R}r_2\})
\end{aligned}$$

The proof consists of showing that the relations $(\mathcal{R}_i)_{i\in\{a,b,c_1,c_2,c_3\}}$ *satisfy the property of (symmetric) synchronous bisimulations* $\mathcal{R}$ *(def. 4):*

$$r_1\mathcal{R}r_2 \Rightarrow (r_1 \bullet\!\!\rightarrow r_1' \Rightarrow (r_2 \bullet\!\!\rightarrow r_2' \wedge r_1'\mathcal{R}r_2')) \wedge (r_1\downarrow x(u) \Rightarrow r_2\downarrow x(u)) \tag{79}$$

To this end, we consider the series of relations $(\mathcal{R}_i^n)_{n\geq 0}$ *defined by*

$$\mathcal{R}_i^0 = \approx \qquad \forall n \geq 0, \mathcal{R}_i^{n+1} = \mathcal{F}_i(\mathcal{R}_i^n)$$

and show that the relations $(\mathcal{R}_i)_{i\in\{a,b,c_1,c_2,c_3\}}$ *satisfy (79).*

$$\forall i \in \{a, b, c_1, c_2, c_3\}, \mathcal{R}_i = \bigcup_{n\geq 0} \mathcal{R}_i^n$$

For all $i \in \{a, b, c_1, c_2, c_3\}$*, the base case* $\mathcal{R}_i^0$ *is trivial, since* $\mathcal{R}_i^0 = \approx$ *is a bisimulation. We prove the inductive case by showing that,*

$$\forall i \in \{a, b, c_1, c_2, c_3\}, \forall n \geq 0, (r_1 \bullet\!\!\rightarrow r_1' \wedge r_1 \mathcal{R}_i^n r_2) \Rightarrow (r_2 \bullet\!\!\rightarrow r_2' \wedge r_1' \mathcal{R}_i^{n+1} r_2') \quad (80)$$

$$r_1 \mathcal{R}_i^n r_2 \Rightarrow (r_1 \downarrow x(u) \Rightarrow r_2 \downarrow x(u)) \quad (81)$$

By induction hypothesis, let $n \geq 0$ *and* $i \in \{a, b, c_1, c_2, c_3\}$ *be such that* $r_1 \mathcal{R}_i^n r_2$ *and let* $r_1 \bullet\!\!\rightarrow r_1'$*. By case analysis, we show that* $r_2 \bullet\!\!\rightarrow r_2'$ *and* $r_1' \mathcal{R}_i^{n+1} r_2'$*.*

Case (a) *Suppose that* $r_1 \mathcal{R}_a^n r_2$ *for which condition (80) holds, consider a name* u' *and write* σ (resp. σ^{-1}) *for* $[u'/u]$ (resp. $[u/u']$)*. By hypothesis, assume that* $r_1\sigma \bullet\!\!\rightarrow r_1'$*. We have:*

$$\begin{aligned} r_1\sigma \bullet\!\!\rightarrow r_1' &\Rightarrow r_1 \bullet\!\!\rightarrow r_1'\sigma^{-1} && (82)\\ &\Rightarrow r_2 \bullet\!\!\rightarrow r_2'' \mathcal{R}_a^n (r_1'\sigma^{-1}) && (83)\\ &\Rightarrow r_2\sigma \bullet\!\!\rightarrow r_2''\sigma && (84)\\ &\Rightarrow r_2\sigma \bullet\!\!\rightarrow r_2''\sigma \mathcal{R}_a^{n+1} r_1' && (85) \end{aligned}$$

Steps (82) and (84) involve a syntactic reasoning with σ*, step (83) is by induction hypothesis, (85) is the conclusion. We have shown that there exists* $r_2 \bullet\!\!\rightarrow r_2' = r_2''\sigma$ s.t. $r_1' \mathcal{R}_a^{n+1} r_2'$*.*

It remains to show that $r_1\sigma \downarrow x(u'')\sigma \Rightarrow r_2\sigma \downarrow x(u'')\sigma$*. By induction hypothesis* $r_1 \mathcal{R}_a^n r_2$*. This requires that* $r_1 \downarrow x(u'') \Rightarrow r_2 \downarrow x(u'')$*. The result is immediate since* $\sigma(u) = u'$*.*

Case (b) *Reflexivity and symmetry are obvious. We only need to show that bisimulation is transitive. By induction hypothesis, suppose that* $r_1 \mathcal{R}_b^n r_2$ *and let* $r_1 \bullet\!\!\rightarrow r_1'$*. By definition of* $\mathcal{R}_b^n$*, there exists* r s.t. $r_1 \mathcal{R}_b^n r$ *and* $r \mathcal{R}_b^n r_2$*. Hence,*

$$\begin{aligned} r_1 \bullet\!\!\rightarrow r_1' &\Rightarrow r \bullet\!\!\rightarrow r' \mathcal{R}_b^n r_1'\\ &\Rightarrow r_2 \bullet\!\!\rightarrow r_2' \mathcal{R}_b^n r'\\ &\Rightarrow r_2 \bullet\!\!\rightarrow r_2' \mathcal{R}_b^{n+1} r_1' \end{aligned}$$

It remains to show that $r_1 \downarrow x(u) \Rightarrow r_2 \downarrow x(u)$ *from the hypothesis* $r_1 \mathcal{R}_b^n r_2$*. This is immediate, since* $r_1 \mathcal{R}_b^n r_2$ *requires that* $r_1 \downarrow x(u) \Rightarrow r \downarrow x(u)$ *and* $r \downarrow x(u) \Rightarrow r_2 \downarrow x(u)$*.*

Case (c1) *Let* $(r_1 \,|\, r) \bullet\!\!\rightarrow r_1'$ *and suppose that* $r_1 \mathcal{R}^n_{c_1} r_2$ *by induction hypothesis. By definition of* $\mathcal{R}^n_{c_1}$*, we have* $r_1 \downarrow x(u) \Rightarrow r_2 \downarrow x(u)$*. By definition of* $\bowtie$ *on traces, synchronous composition may only inhibit the composition of traces having mismatching messages. Hence, for a given* r*, we naturally have that* $(r_1 \,\|\, r)\downarrow x(u); (r_2 \,|\, r)\downarrow x(u)$ *from the hypothesis. Take the same* r*, then there exists* r_2' s.t. $(r_2 \,\|\, r) \bullet\!\!\rightarrow r_2'$*. By definition of* $\mathcal{F}_{c_1}$*,* $r_1' \mathcal{R}^{n+1}_{c_1} r_2'$*.*

Case (c2) *Let* $(\exists x.r_1) \bullet\!\!\rightarrow r_1'$ *and suppose that* $r_1 \mathcal{R}^n_{c_2} r_2$ *holds by induction hypothesis. Let* $\exists x.r_2 \bullet\!\!\rightarrow r_2'$ *then* $r_1' \mathcal{R}^{n+1}_{c_2} r_2'$ *since* x *can be instantiated by the same fresh name* y *in both* r_1' *and* r_2'*. By induction hypothesis, also suppose that* $r_1 \downarrow x(u) \Rightarrow r_2 \downarrow x(u)$*. For the same reason, we have* $(\exists x.r_1)\downarrow x(u) \Rightarrow (\exists x.r_2)\downarrow x(u)$*.*

Case (c3) *Showing that* $r_1' \mathcal{R}^{n+1}_{c_3} r_2'$ *holds is done in an identical way to* (c_1) *since, for* $j = 1, 2$*, the runs* $\mathsf{do}\, f \,\mathsf{in}\, r_j = \mathsf{do}\, f \,\mathsf{in}\, (\mathsf{do}\, f_j \,\mathsf{in}\, t_j)$ *have the same transitions than* $\mathsf{do}\, f \,\|\, f_j \,\mathsf{in}\, t_j$*.*

Property 3 *Asynchronous bisimulation* $\approx_\circ$ *is a congruence*

Proof of property 3 *As the asynchronous transition relation* $\circ(.)$ *is obtained from* $\bullet(.)$ *by the removal of instants* $\mathcal{S}$*, property 3 is a direct corollary of property 2.*

F. Proof of property 4

Property 4 $\forall p, p \simeq \mathcal{D}_p$

Proof of property 4 *The proof is by induction on the structure of* p*.*

1. **Case of** $p = p_1 + p_2$ *(and identically for* $p = \forall v.p'$*,* $p = \exists x.p'$ *and* $p = \mathsf{do}\, p_1 \,\mathsf{in}\, p_2$*)*
   By induction hypothesis,

   $$\forall i = 1, 2, p_1 \simeq \mathcal{D}_{p_i} \tag{86}$$

   From (86) and by transitivity of $\simeq$*,*

   $$p_1 + p_2 \simeq \mathcal{D}_{p_1} + \mathcal{D}_{p_2} \tag{87}$$

   From (87) and by definition of $\mathcal{D}[\![.]\!]$ *and the property 1,*

   $$p \simeq \mathcal{D}_{p_1 + p_2} \simeq \mathcal{D}_{p_1} + \mathcal{D}_{p_2}$$

2. **Case of** $p = p_1 \,\|\, p_2$ *By induction hypothesis,*

   $$\forall i = 1, 2, p_1 \simeq \mathcal{D}_{p_i} \tag{88}$$

   By definition

   $$\mathcal{D}[\![p_1 \,|\, p_2]\!] = \sum_{\substack{\forall \tilde{v_1}.\exists \tilde{x_1}.\mathsf{do}\ \mathcal{D}_1 \,\mathsf{in}\, s_1 \sqsubseteq \mathcal{D}[\![p_1]\!] \\ \forall \tilde{v_2}.\exists \tilde{x_2}.\mathsf{do}\ \mathcal{D}_1 \,\mathsf{in}\, s_1 \sqsubseteq \mathcal{D}[\![p_1]\!] \,|\, s_1 \widehat{\bowtie} s_2}} \forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.(\mathsf{do}\, \mathcal{D}[\![\mathcal{D}_1 \,|\, \mathcal{D}_2]\!] \,\mathsf{in}\, \mathcal{U}[\![s_1, s_2]\!]) \tag{89}$$

where

$$
\begin{array}{rcll}
s_1 \hat{\bowtie} s_2 & \Leftrightarrow & (x \in \mathrm{fv}_{\mathcal{X}}(p_1) \cap \mathrm{fv}_{\mathcal{X}}(p_2) \Leftrightarrow (x \in \mathrm{fv}_{\mathcal{X}}(s_1) \Leftrightarrow x \in \mathrm{fv}_{\mathcal{X}}(s_2))) & (90) \\
\mathcal{U}[\![s_1, s_2]\!] & = & \mathcal{G}[\![(s_1 \,|\, s_2)\sigma]\!] & (91) \\
& \text{s.t.} & \forall l(v), l(v') \subseteq (s_1, s_2), v \neq v' \Rightarrow \sigma \ni [v/v'] & (92) \\
& & \forall l(v), l(u) \subseteq (s_1, s_2), v \neq u \Rightarrow \sigma \ni [u/v] & (93) \\
\mathcal{G}[\![s]\!] & = & s \,\|\, (\prod_{v(u) \subseteq s}^{l(u') \subseteq s} \mathsf{a}(v = l \Rightarrow u = u')) & (94)
\end{array}
$$

From (89) and by induction hypothesis on all $\mathcal{D}_{1,2}$*,*

$$
\mathcal{D}_{\forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.(\mathcal{D}_1 \,|\, \mathcal{D}_2)} \simeq \mathcal{D}_{\forall \tilde{v_1}.\exists \tilde{x_1}.\mathcal{D}_1} \,\|\, \mathcal{D}_{\forall \tilde{v_2}.\exists \tilde{x_2}.\mathcal{D}_2} \tag{95}
$$

From (89) and by definition of $\mathcal{U}[\![.]\!]$*, it remains to show that every pre-order* s *chosen from* $\forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.\mathcal{U}[\![s_1, s_2]\!]$ *corresponds to a* s' *from* $\forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.(s_1 \,\|\, s_2)$ *modulo the guards* $\mathsf{a}(\not\perp) \subseteq s'$ *introduced by* $\mathcal{U}[\![.]\!]$ *and removed by* $\bullet(.)$*.*

This amounts to prove that the abstract composability relation $\hat{\bowtie}$ *and the unification function* $\mathcal{U}[\![]\!]$ *implement the concrete composability relation* $\bowtie$*. Let us chose*

$$s \sqsubseteq \forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.(s_1 \,|\, s_2)$$

By definition of synchronous composition, there exists

$$(s'_i \sqsubseteq \forall \tilde{v_i}.\exists \tilde{x_i}.s_i)_{i=1}^2, s'_1 \bowtie s'_2 \wedge s = s'_1 \cup s'_2.$$

By definition of quantification, there exists

$$\forall i = 1, 2, s_i\sigma_i = s'_i \wedge \mathrm{dom}\,\sigma_i = \tilde{v_i}\tilde{x_i}$$

The inspection of the definition of σ *in (92-93) shows that* $\sigma_1 \circ \sigma_2$ *contains* σ i.e. $\sigma_1 \circ \sigma_2 = \sigma \circ \sigma'$*. The inspection of the definition of* $\mathcal{G}[\![]\!]$ *further shows that* σ' *may only be chosen in such a way to satisfy the guard of* $\mathcal{G}[\![(s_1 \,\|\, s_2)\sigma]\!]\sigma'$*. As a result,*

$$\forall s \sqsubseteq (\forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.(s_1 \,\|\, s_2))\exists s' \sqsubseteq (\forall \tilde{v_{1,2}}.\exists \tilde{x_{1,2}}.\mathcal{U}[\![s_1, s_2]\!]), s = s' \setminus \mathsf{a}(\not\perp) \tag{96}$$

From (95), (96) and by definition of the transition relation

$$p_1 \,\|\, p_2 \simeq \mathcal{D}_{p_1 \,|\, p_2}$$

G. Proof of property 5

Property 5 $\forall p, \mathcal{D}_p \simeq \mathcal{H}_p$

Proof of property 5 *The proof is by induction on the structure of* $\mathcal{H}_p$.

1. **Case of** $\mathcal{H}_p = \mathcal{H}_{p_1} + \mathcal{H}_{p_2}$
   By induction hypothesis, $\forall i = 1, 2, \mathcal{D}_{p_1} \simeq \mathcal{H}_{p_i}$ (a).
   By definition, $\mathcal{D}_{p_1} + \mathcal{D}_{p_2} = \mathcal{H}_{p_1} + \mathcal{H}_{p_2}$ (b).
   From (a-b) and by property 1, $\mathcal{D}_{p_1} + \mathcal{D}_{p_2} \simeq \mathcal{H}_{p_1} + \mathcal{H}_{p_2}$.

2. **Case of** $\mathcal{H}_p = \mathsf{do}\ F\ \mathsf{in}\ s$
   By induction hypothesis $\mathcal{D} \simeq F$ (a).
   From (a) and by definition of the transition relation, $\mathcal{H}_p \simeq \mathcal{D}_p$.

3. **Case of** $\mathcal{H}_p = \overbrace{s \triangleright F}^{T}$ *By induction hypothesis,* $\mathcal{D} \simeq F$ (a). *By definition of the relation* $\triangleright$,

$$s \triangleright F = \sum_{p \sqsubseteq F} s \,\|\, p \;\Leftrightarrow\; s = \bigcap_{p \sqsubseteq F} (s \cup p) \;\wedge\; \forall p \sqsubseteq F, p = \bigcap_{p' \subseteq (s \cup p)}^{s \cup p' = s \cup p} p' \;\; (b)$$

   From (a-b), $\mathcal{D} = F$ *hence* $\mathcal{D} \simeq F$ *from property 1*

H. Proof of theorem 4

Theorem 4 *If* $\mathcal{H}_p$ *is well-guarded then* p *is* endochronous.

Proof of theorem 4 *The proof first requires a reformulation of the property of endochrony amenable to an inductive proof. By definition 7,* p *is endochronous iff*

$$\forall \boldsymbol{r} \in \circ^\omega p, \exists ! r \in \bullet^\omega p, \boldsymbol{r} = (r)_\circ \tag{97}$$

where r *is unique up to silent transitions* 1. *We thus only consider silent-transitions-free runs* r. *By definition of the transition relations* $\circ(.)$ *and* $\bullet(.)$, *equation (97) holds iff*

$$\forall \boldsymbol{r} \in \circ^\omega p, \exists ! (f_i, s_i)_{i \geq 0}, \boldsymbol{r} = \mathsf{do} \prod_{i \geq 0} f_i \ \mathsf{in}\ \circ_{i \geq 0} s_i = r_\circ = \mathsf{do} \prod_{i \geq 0} f_i \ \mathsf{in}\ (\bullet_{i \geq 0} s_i)_\circ \tag{98}$$

Equation (98) provides a unique decomposition of all traces $\boldsymbol{r} = \mathsf{do}\ f\ \mathsf{in}\ \boldsymbol{t}$ *of* p *consisting of*

- *a series of families* $(f_i)_{i \geq 0}$, *available from the leaves of* $\mathcal{H}_p$, s.t. $f = \prod_{i \geq 0} f_i$.
- *a series of pre-orders* $(s_i)_{i \geq 0}$, *whose asynchronous concatenation forms* $\boldsymbol{t}$, s.t.

$$\forall i > 0, \mathsf{do}\ f_i\ \mathsf{in}\ s_i \sqsubseteq f^i \text{ where } f^i = (\prod_{j < i} f_j)$$

Showing that a well-guarded process p satisfies (98) amounts to proving the inductive property $(\mathcal{P}(n))_{n\geq 0}$,

$$\forall n \geq 0, \mathcal{P}(n) : f^n \text{ is well-guarded and } \exists!\boldsymbol{t}'\exists!(\mathsf{do}\ f_n\ \mathsf{in}\ s_n) \sqsubseteq f^n, \boldsymbol{t} = (\circ_{i\leq n} s_i)\circ \boldsymbol{t}'$$

The base case $\mathcal{P}(0)$ is trivial and we prove the inductive case $\forall n \geq 0, \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$. Given $n \geq 0$ and, by induction hypothesis, $\mathcal{P}(n)$, this amounts to showing that $\mathcal{P}(n+1)$ holds true. By hypothesis,

$$f^n \text{ is well-guarded} \tag{99}$$
$$\boldsymbol{t} = (\circ_{i\leq n} s_i)\circ \boldsymbol{t}' \tag{100}$$

Let $\mathcal{H}^n$ be the HNF *of f^n and $(\mathsf{do}\ f_{n+1}\ \mathsf{in}\ s_{n+1}) \sqsubseteq \mathcal{H}^n$. From (99) and the properties (3a) and (3b) of the definition 10,*

$$f^{n+1} = f^n \,|\, f_{n+1} \text{ is well-guarded} \tag{101}$$

By definition of $\mathcal{H}^n$, the pre-order $s_{n+1} = \prod_{j=1}^{m} s^j$ is a path to f_{n+1} in $\mathcal{H}^n$. From property (2b) of definition 10 and at every branch $j = 1, m$ $s^j \rhd (\sum_{k=1}^{m_j} T^j)$ of $\mathcal{H}^n$, the guards $\mathsf{a}_k(e_k)$ of the T^j are mutually exclusive. Hence, all other guards than from s_{n+1} in $\mathcal{H}^n$ yield ff. Therefore, there exists a unique $\boldsymbol{t}''$ and s_{n+1} such that:

$$\boldsymbol{t} = (\circ_{i\leq n+1} s_i)\circ \boldsymbol{t}'' \tag{102}$$

From (101) and (102), $\mathcal{P}(n+1)$ which proves the theorem.

I. Proof of theorem 5

Theorem 5 *If $(p^j)_{j=1}^2$ are well and mutually guarded then they are isochronous.*

Proof of theorem 5 *By definition 5, p^1 and p^2 are isochronous iff*

$$\circ^\omega p^1 \parallel \circ^\omega p^2 \ \subseteq\ \bullet^\omega(p^1 \parallel p^2) \tag{103}$$
$$\bullet^\omega(p^1 \,|\, p^2) \ \subseteq\ \circ^\omega p^1 \parallel \circ^\omega p^2 \tag{104}$$

The second inclusion (104) is straightforward: consider $(r^1, r^2) \in (\bullet^\omega p^1, \bullet^\omega p^2)$ such that $r_1 \bowtie r^2$ then $(r_1)_\circ \bowtie (r^2)_\circ$ hence $(r_1)_\circ \parallel (r^2)_\circ \in \circ^\omega p^1 \parallel \circ^\omega p^2$.

The proof of the first inequation (103) requires an inductive formulation similar to that of theorem H.

$$\forall(\boldsymbol{r}^1, \boldsymbol{r}^2) \in (\circ^\omega p^1, \circ^\omega p^2), \exists(r^1, r^2) \in (\bullet^\omega p^1, \bullet^\omega p^2), \boldsymbol{r}^1 \parallel \boldsymbol{r}^2 = (r^1 \parallel r^2)_\circ \tag{105}$$

By definition of the transition relations $\circ(.)$ and $\bullet(.)$, equation (105) holds iff

$$\begin{aligned} \forall j \in \{1,2\}, \forall \boldsymbol{r}^j \in \circ^\omega p^j, \exists (f_i^j, s_i^j)_{i\geq 0}, \boldsymbol{r}^1 \parallel \boldsymbol{r}^2 &= \parallel_{j=1,2}(\mathsf{do} \prod_{i\geq 0} f_i^j \ \mathsf{in}\ (\circ_{i\geq 0} s_i^j)) \\ &= (\prod_{j=1,2} (\mathsf{do} \prod_{i\geq 0} f_i^j \ \mathsf{in}\ (\bullet_{i\geq 0} s_i^j)))_\circ \\ &= (r^1 \,|\, r^2)_\circ \end{aligned} \tag{106}$$

Equation (106) provides a decomposition of the traces $\boldsymbol{r}^j = \mathsf{do}\ f^j\ \mathsf{in}\ \boldsymbol{t}^j$ of p^j consisting of

- *a series of families* $(f_i^j)_{i\geq 0}$*, available from the leaves of* $\mathcal{H}_{p^j}$*,* s.t. $f^j = \prod_{i\geq 0} f_i^j$.
- *a series of pre-orders* $(s_i^j)_{i\geq 0}$*, whose asynchronous concatenation forms* $\boldsymbol{t}^j$*,* s.t.

$$\forall j, \forall i > 0, \mathsf{do}\ f_i^j\ \mathsf{in}\ s_i^j \sqsubseteq f^{(j,i)} \text{ where } f^{(j,i)} = (\prod_{k<i} f_k^j)$$

Showing that the well-guarded processes $(p^j)_j$ *satisfy (106) amounts to proving the inductive property* $(\mathcal{P}(n))_{n\geq 0}$*,*

$$\begin{array}{lll} \forall n \geq 0, \mathcal{P}(n) : \forall j \in \{1,2\}, & f^{(j,n)} \text{ are mutually guarded} & \\ & \exists (\mathsf{do}\ f_n^j\ \mathsf{in}\ s_n^j) \sqsubseteq f^{(j,n)}, \quad t_n^j = (\bullet_{i\leq n} s_i^j), & \\ & \boldsymbol{t}_n^j = (t_n^j)_\circ & \\ & \|_j \boldsymbol{t}_n^j = (\prod_j t_n^j)_\circ & \end{array}$$

The base case $\mathcal{P}(0)$ *is trivial and we focuse on the inductive case* $\forall n \geq 0, \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$*. Given* $n \geq 0$ *and, by induction hypothesis,* $\mathcal{P}(n)$*, this amounts to showing that* $\mathcal{P}(n+1)$ *holds true. By hypothesis,*

$$\begin{array}{rcll} \mathcal{P}(n) : f^{(j,n)} & \text{are} & \text{mutually guarded} & (107) \\ t_n^j & = & (\bullet_{i\leq n} s_i^j), & \\ \boldsymbol{t}_n^j & = & (t_n^j)_\circ, & \\ \|_j \boldsymbol{t}_n^j & = & (\prod_j t_n^j)_\circ & (108) \end{array}$$

Let $\mathcal{H}^{(j,n)}$ *be the* Hnf *of* $f^{(j,n)}$ *and* $(\mathsf{do}\ f_{n+1}^j\ \mathsf{in}\ s_{n+1}^j) \sqsubseteq \mathcal{H}^{(j,n)}$*. From (107) and the properties (2) and (3) of the definition 11,*

$$\begin{array}{lll} \forall j \forall k \neq j, & f^{(j,n)} \text{ and } f_{n+1}^j \text{ are mutually guarded} & (109) \\ & f^{(j,n)} \text{ and } f_{n+1}^k \text{ are mutually guarded} & (110) \end{array}$$

By definition of $\mathcal{H}^{(j,n)}$*, the pre-orders* s_{n+1}^j *are instances taken from two pathes* P_{n+1}^j *of* $\mathcal{H}^{(j,n)}$*. Let* $L = \cap_j \mathcal{H}^{(j,n+1)}$*, from property (1a-1b) of definition 11, the guards* $\mathsf{a}(e_{n+1}^j)$ *of* P_{n+1}^j *are* s.t.*:*

$$\neg(\wedge_j e_{n+1}^j) \Rightarrow \cap_j(\mathrm{fv}_{\mathcal{L}}(s_{n+1}^j)) = \emptyset \quad \vee \quad (\wedge_j e_{n+1}^j) \Rightarrow (=_j (\mathrm{fv}_{\mathcal{L}}(s_{n+1}^j) \cap L)) \qquad (111)$$

Hence, from (108) and by definition of $\bowtie$ *with (111), we have*

$$\|_j \boldsymbol{t}_{n+1}^j = (\prod_j t_{n+1}^j)_\circ$$

which, together with (109-110), proves $\mathcal{P}(n+1)$ *hence the theorem.*

J. Notations

$$
\begin{array}{lll}
\mathrm{fv}_{\mathcal{V}}\ \mathsf{1} & = \emptyset \\
\quad \mid l(u) & = \{u\} \cap \mathcal{V} \\
\quad \mid \mathsf{a}(g(\tilde{u})) & = \{u \in \tilde{u}\} \cap \mathcal{V} \\
\quad \mid p;p' & = \mathrm{fv}_{\mathcal{V}}(p) \cup \mathrm{fv}_{\mathcal{V}}(p') \\
\quad \mid p \,|\, p' & = \mathrm{fv}_{\mathcal{V}}(p) \cup \mathrm{fv}_{\mathcal{V}}(p') \\
\quad \mid p + p' & = \mathrm{fv}_{\mathcal{V}}(p) \cup \mathrm{fv}_{\mathcal{V}}(p') \\
\quad \mid \forall v.p & = \mathrm{fv}_{\mathcal{V}}(p) \setminus v \\
\quad \mid \exists x.p & = \mathrm{fv}_{\mathcal{V}}(p) \\
\quad \mid \mathsf{do}\, f\, \mathsf{in}\, p & = \mathrm{fv}_{\mathcal{V}}(f) \cup \mathrm{fv}_{\mathcal{V}}(p)
\end{array}
$$

$$
\begin{array}{lll}
\mathrm{fv}_{\mathcal{L}}\ \mathsf{1} & = \emptyset \\
\quad \mid l(u) & = \{l\} \cap \mathcal{L} \\
\quad \mid \mathsf{a}(g(\tilde{u})) & = \emptyset \\
\quad \mid p;p' & = \mathrm{fv}_{\mathcal{L}}(p) \cup \mathrm{fv}_{\mathcal{L}}(p') \\
\quad \mid p \,|\, p' & = \mathrm{fv}_{\mathcal{L}}(p) \cup \mathrm{fv}_{\mathcal{L}}(p') \\
\quad \mid p + p' & = \mathrm{fv}_{\mathcal{L}}(p) \cup \mathrm{fv}_{\mathcal{L}}(p') \\
\quad \mid \forall v.p & = \mathrm{fv}_{\mathcal{L}}(p) \setminus v \\
\quad \mid \exists x.p & = \mathrm{fv}_{\mathcal{L}}(p) \setminus x \\
\quad \mid \mathsf{do}\, f\, \mathsf{in}\, p & = \mathrm{fv}_{\mathcal{L}}(f) \cup \mathrm{fv}_{\mathcal{L}}(p)
\end{array}
$$

$$
\begin{array}{lll}
\mathrm{fv}_{\mathcal{X}}\ \mathsf{1} & = \emptyset \\
\quad \mid l(u) & = \{l, u\} \cap \mathcal{X} \\
\quad \mid \mathsf{a}(g(\tilde{u})) & = \{u \in \tilde{u}\} \cap \mathcal{X} \\
\quad \mid p;p' & = \mathrm{fv}_{\mathcal{X}}(p) \cup \mathrm{fv}_{\mathcal{X}}(p') \\
\quad \mid p \,|\, p' & = \mathrm{fv}_{\mathcal{X}}(p) \cup \mathrm{fv}_{\mathcal{X}}(p') \\
\quad \mid p + p' & = \mathrm{fv}_{\mathcal{X}}(p) \cup \mathrm{fv}_{\mathcal{X}}(p') \\
\quad \mid \forall v.p & = \mathrm{fv}_{\mathcal{X}}(p) \\
\quad \mid \exists x.p & = \mathrm{fv}_{\mathcal{X}}(p) \setminus x \\
\quad \mid \mathsf{do}\, f\, \mathsf{in}\, p & = \mathrm{fv}_{\mathcal{X}}(f) \cup \mathrm{fv}_{\mathcal{X}}(p)
\end{array}
$$

Figure 26. Free variables $\mathrm{fv}_{\mathcal{V}}(.)$, locations $\mathrm{fv}_{\mathcal{L}}(.)$ and ports $\mathrm{fv}_{\mathcal{X}}(.)$

$c \in \mathcal{C}$	constant	$\_ \bullet \_$	synchronous concatenation
$x \in \mathcal{X}$	port	$\bullet(\_)$	synchronous transition
$v \in \mathcal{V}$	variable	σ	substitution
$u \in \mathcal{U}$	names	$\bullet^{\omega}\_$	synchronous denotation
$l \in \mathcal{L}$	label	$\_ \bullet\!\!\rightarrow \_$	synchronous step
e	expression	$\_ \simeq \_$	synchronous equivalence
$g(\_)$	function	$\_ \downarrow \_$	synchronous observation
s	pre-order	$\_ \approx \_$	synchronous bisimulation
p	process	$\_ \uparrow \_$	restriction
f	family	$\boldsymbol{p}$	asynchronous process
$\mathsf{1}$	silence	$\_ \parallel \_$	asynchronous composition
$l(u)$	message	$\boldsymbol{t}$	asynchronous trace
$\mathsf{a}(e)$	guard	$\boldsymbol{r}$	asynchronous run
$\_;\_$	sequence	$\_ \circ \_$	asynchronous concatenation
$\_ \mid \_$	composition	$\circ(\_)$	asynchronous transition
$\forall v.\_$	generalization	$(\_)_{\circ}$	desynchronization
$\exists x.\_$	restriction	$\circ^{\omega}\_$	asynchronous denotation
$\mathcal{M}$	set of messages	$\_ \circ\!\!\rightarrow \_$	asynchronous step
$\mathcal{T}$	transition relation	$\_ \simeq_{\circ} \_$	asynchronous equivalence
$\_ \cup \_$	union	$\_ \Downarrow \_$	asynchronous observation
$\_ \cap \_$	intersection	$\_ \approx_{\circ} \_$	asynchronous bisimulation
$\min \_$	minima	$\mathcal{D}$	DNF
$\mathrm{pred}\ \_\ \_$	predecessors	$\mathcal{H}$	HNF
$\_ \setminus \_$	exclusion	F	forest
$\_ \bowtie \_$	composability	T	tree
$\sum \_$	sum	$\mathring{\_}$	interior
$\prod \_$	product	$\partial \_$	boundary
$\mathrm{wf}(\_)$	well-formedness	$\_ \rhd \_$	factorization
$i_x \in \mathcal{S}$	synchronization	$\mathcal{D}[\![\_]\!]$	DNF algorithm
$i \in \mathcal{I}$	instant	$\mathcal{U}[\![\_]\!]$	unification
t	synchronous trace	$\mathcal{G}[\![\_]\!]$	guarding
r	synchronous run	$\mathcal{H}[\![\_]\!]$	hierarchization
$\_ \bullet^{?} \_$	concatenability		

Figure 27. Summary of notations

Contents

1 Introduction **3**
1.1 Related Work 6
1.2 Contributions 6

2 Synchronous pre-order transition systems **8**
2.1 Structure 8
2.2 Notations 8
2.3 Conventions 9
2.4 Processes 9
2.5 Synchronous composition 10
2.6 Quantifiers 11
2.7 Guards 12
2.8 Synchronous semantics 13
2.9 Synchronous semantics 15
2.10 Synchronous equivalence relations 16
2.11 Asynchronous semantics 17
2.12 Asynchronous traces 17
2.13 Asynchronous equivalence relations 18

3 Expressivity of pre-order transition systems **19**
3.1 Semantics of the JOIN-calculus 19
3.2 Semantics of the FUSION-calculus 20
3.3 Synchronous stream functions 21

4 From synchrony to asynchrony **23**
4.1 Formal definitions 25
4.2 Hierarchic transition systems 27
4.2.1 Disjunctive normal forms 27
4.2.2 The hierarchic normal form 28
4.2.3 Factorization 29
4.2.4 Hierarchization 29
4.3 Guarded hierarchies 30
4.4 Formal property 32

5 Discussion **33**

6 Conclusion **34**

A Proof of theorem 1 **38**

B Proof of theorem 2 **41**

C Proof of theorem 3 **47**

D Proof of property 1 **50**

E Proof of properties 2 and 3 **50**

F Proof of property 4 **52**

G Proof of property 5 **54**

H Proof of theorem 4 **54**

I Proof of theorem 5 **55**

J Notations **57**

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399