On Asynchronous Communication Semantics *

Kohei Honda and Mario Tokoro†

Department of Computer Science,
Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, 223,
Japan

March 10, 1992

Abstract

This paper presents some results concerning equational theories for an elementary calculus based on a fragment of Milner's π-calculus. The system is interesting because it realises asynchronous message passing not by extending but reducing the original fragment, while preserving the computational power. The bisimulation based on a novel asynchronous transition system is introduced and studied. Presented results include congruence of the bisimilarity for the calculus, its relationship with two other asynchronous theories based on traces and failures, strict inclusion of its synchronous counterpart in the asynchronous theory, and the method called the $\mathcal{I}$ *completion* that transforms two asynchronously bisimilar terms into synchronously bisimilar ones.

1 Introduction

This paper presents some results concerning equational theories for an elementary calculus based on a fragment of Milner's π-calculus [22]. The calculus, which first appeared in its present form in [12], expresses asynchronous communication not by adding extra machinery to the original synchronous communication mechanism, but by *reducing* the original one. It is capable of expressing synchronous (or handshake) communication quite concisely in terms of its reduced set of constructs and can express various computational structures elegantly (cf. [12, 13]). These and other interesting findings prompted our investigation to construct a theory of concurrent computation purely based on asynchronous message passing.

The asynchronous communication considered here, is in its most unconstrained form. That is, we assume that a "sender" can freely send messages with no bound in the number

*To appear in *Object-based Concurrent Computing*, ed. by Tokoro,M., Nierstrasz,O., and Wegner,P., LNCS 612, Springer Verlag, June 1992.

†Also with Sony Computer Science Laboratory Inc. 3-14-13 Higashi-Gotanda, Shinagawa-ku, Tokyo, 141, Japan

of pending messages, and the "receiver" will consume a message out of many targeted to it in a quite arbitrary fashion (i.e. nondeterminism in the arrival order). The framework can be thought of as a kind of abstract representation of real-life distributed computing environments [28, 15], or as a logically extreme communication framework where the other extreme is pure synchronous interaction in the form of, say, CCS [17]. The fact that a basic formal construct originally developed for synchronous interaction in process calculi can handle asynchronous interaction comes as a pleasant surprise.

The difference between asynchrony and synchrony in communication lies in whether the event of reception of communication is tied to the output (sender) system or not. Separating the sender from the event of reception abstractly represents the notion of *distance* between two communicating parties. From this viewpoint, the synchronous communication may be considered as a special case where this distance is always zero. Another viewpoint is that asynchronous communication *is* a special case of synchronous communication because it can be formulated using buffers composed of synchronously communicating agents. We do not, however, intend to give arguments for or against these standpoints. Our goal is to pursue formal computational and semantic constructions based purely on asynchronous interaction, and deepen our understanding of the true significance of this theoretically neglected communication paradigm in a formal setting.

In this context, the paper introduces an equational theory called *asynchronous bisimulation* defined for the elementary calculus, and studies its basic properties. The equational theory forms the basis of various investigation related with the calculus exposed elsewhere (cf. e.g. [13, 14]). Section 2 introduces the basic notions of the calculus. The way of presenting the basic transition relation is much simpler than our original formulation [12] due to the normal form representation of term expressions. Section 3 then defines the *asynchronous bisimulation* and studies its basic properties. We prove that the bisimulation is a congruence relation, show several alternative characterizations for the equivalence relation, and position it among other asynchronous theories. Section 4 tries to understand the asynchronous theory in relationship with its synchronous counterpart. We prove that the asynchronous bisimilarity is strictly more general than the synchronous one, and show that a method called the $\mathcal{I}$ *completion* can transform any asynchronously bisimilar pair of terms into synchronously bisimilar pair, without changing their (asynchronous) meaning. The method is effective to weak theories in general. Finally Section 5 concludes the paper.

Note. Throughout the exposition, $\stackrel{def}{=}$ denotes definitional equality.

2 The Calculus

This section introduces the reader to the basic definitions of the calculus, the semantic construction for which is the subject of the present exposition. There are some novel points in the formulation of transition rules, which is surprisingly simple because of the normal form representation of terms.

Syntax

The elementary calculus has its origin in a computationally complete fragment of the π-calculus presented in [22] (cf. [24, 19, 20]). Inheriting important construction by Milner, small syntactic changes given to the original formalism are the contraction of an output-guarded term to a message (i.e. $\bar{a}v.P$, a process which offers v through a port a, becomes $\leftarrow av$, a message to a target a with a value v), and introduction of a restricted form of recursive structure, in place of replication. These changes make the formalism smaller than the original, while retaining the same expressive power [12, 14], and let it embody the notion of *asynchrony* both in computation and communication. Various aspects of the calculus are currently being investigated (see e.g. [12, 13, 14]).

Definitions 2.1 *Syntax.* $\boldsymbol{N}$, $\widetilde{\boldsymbol{N}}$, *and* $\boldsymbol{V}$ *are the sets of infinite (port) names, the set of finite sequences of names, and the set of infinite (structure) variables*[1], *respectively. Then the set of term expressions (or simply terms)* $\boldsymbol{C}$ *is given by the following abstract syntax.*

$$
\begin{array}{rcll}
\boldsymbol{C} & = & \leftarrow \boldsymbol{NN} & \textit{(a message)} \\
 & | & \boldsymbol{NN}.\boldsymbol{C} & \textit{(a receptor)} \\
 & | & \{\boldsymbol{V}(\widetilde{\boldsymbol{N}}) = \boldsymbol{NN}.\boldsymbol{C}\}(\widetilde{\boldsymbol{N}}) & \textit{(a recursively defined receptor)} \\
 & | & \boldsymbol{V}(\widetilde{\boldsymbol{N}}) & \textit{(a variable with a parameter)} \\
 & | & |\boldsymbol{N}|\,\boldsymbol{C} & \textit{(scope restriction)} \\
 & | & \boldsymbol{C}, \boldsymbol{C} & \textit{(concurrent composition)} \\
 & | & \Lambda & \textit{(the null term)} \quad \blacksquare
\end{array}
$$

Conventions 2.2

(i) $a, b, c, ..$ *range over* $\boldsymbol{N}$.

(ii) $\tilde{a}, \tilde{b}, \tilde{c}, ..$ *range over* $\widetilde{\boldsymbol{N}}$.

(iii) $X, Y, Z, ..$ *range over* $\boldsymbol{V}$.

(iv) $P, Q, R, ..$ *range over* $\boldsymbol{C}$. ∎

We will assume that the constructor "," is the weakest in association, others being of the same precedence. When some ambiguity arises, we will use parenthesis as usual. Top level terms are often called *configurations*. For convenience, some parts of syntactic structures are given their respective names, as below.

Definitions 2.3

(i) In $\leftarrow av$, *a is called a* target *and v is called a* value. *When the value is insignificant, we will sometimes write* $\leftarrow a$.

(ii) In $ax.P$, a is called an input port, *x is called a* formal parameter, *and P is called* a body. *When the formal parameter is insignificant, we will sometimes write $a.P$.*

(iii) In $\{X(\tilde{x}) = by.P\}(\tilde{a})$, the part $\{X(\tilde{x}) = by.P\}$ is called a recursive structure, *of which $X(\tilde{x})$ and $by.P$ are called a* head *and a* body, *respectively. $\tilde{a}$ is called a* parameter *to the recursive structure.*

[1] In [12], they are called *term variables.*

(iv) In $X(\tilde{a})$, X is a (structure) variable and $\tilde{a}$ is a parameter to the variable.

(v) In $|v|P$, $|v|$ is called a (scope) restrictor *and P is called a* body. *$|xyz|P$ denotes $|x|(|y|(|z|P))$.* ∎

Some intuitive understanding of each syntactic form will help. First, $\leftarrow av$ represents a message in transit, carrying a name v to a receptor which owns a as its input port name. A message resembles an argument for functional application. Then a receptor, say $ax.P$, resembles a function body, which receives (at the port a) a message and instantiate free occurences of x in P with the received value. A term such as $\{X(\tilde{x}) = by.P\}(\tilde{a})$ is a recursive variant of a receptor. Now an important construction $|v|P$ implies that free occurrences of v in P are *local* to P, or, in other words, different from any other v out of its scope. Combined with recursive structures, it gives to the formalism capability to generate new port names unboundedly. $P,\ Q$ is a *concurrent composition* of P and Q, written $P|Q$ in CCS and π-calculus. Finally Λ is syntactic convention to express "nothing".

Three kinds of bindings play essential roles in the calculus.

Definitions 2.4 *Bindings.*

(i) In $ax.P$, the formal parameter x, occurring bound itself, binds free occurrences of the same name in the body.

(ii) In $\{X(\tilde{x}) = by.P\}(\tilde{a})$, names in $\tilde{x}$ at the head, occurring bound themselves, bind their free occurrences in the body. The variable X at the head, occurring bound, binds its free occurrences in the body. We hereafter assume that the names and the variables occurring free in the body is the subset of the names and the variable occurring in the head, and names in the head are pairwise distinct.

(iii) In $|v|P$, v of $|v|$, occurring bound, binds its free occurrences in the body.

$\mathcal{BN}(P)$ (resp. $\mathcal{FN}(P)$) is a set of bound (resp. free) names in P. $\mathcal{N}(P)$ is the union of these two. ∎

Then we have two kinds of *substitutions.* One is a name to a name, the other is a variable to a structure. Note the difference from e.g. the λ calculi, where we have no such decomposition.

Definitions 2.5 *Substitutions.*

(i) $P[v/x]$ denotes the result of substituting free occurrences of x in P for v, taking care of name collision as usual. α-convertibility among terms is defined in the standard way.

(ii) $P[\Upsilon/X]$ where Υ stands for some recursive structure, denotes the result of substituting free occurrences of X for the structure. ∎

Let us list some basic facts about substitution (cf. Chapter 1 of [10]).

Lemma 2.6 In (iv) and (v) let x and v be distinct.

(i) $P[x/x] \stackrel{def}{=} P$

(ii) $z \notin \mathcal{FN}(P) \quad \Rightarrow \quad P[w/z] \stackrel{def}{=} P$

(iii) $z \notin \mathcal{FN}(P) \quad \Rightarrow \quad P[z/x][y/z] \stackrel{def}{=} P[y/x]$

(iv) $P[w/x][v/y] \stackrel{def}{=} P[v/y][(w[v/y])/x]$

(v) $P[v/x][w/x] \stackrel{def}{=} P[v/x]$

(vi) $z \notin \mathcal{FN}(P) \cup \mathcal{FN}(Q) \quad \Rightarrow \quad (P[v/x],\ Q) \stackrel{def}{=} (P,\ Q[z/x])[v/x][x/z]$ ∎

We omit the proof. To prove (vi), one needs (ii), (iii) and (i) (in this order).

Finally the notion of the set of free input port names of a configuration, by which receptors in it are "ready to interact" with messages from outside, will become important later.

Definitions 2.7 *The set of free input names of P, denoted $\mathcal{FI}(P)$, are inductively given by:* $\mathcal{FI}(ax.P) \stackrel{def}{=} \{a\}$, $\mathcal{FI}(|x|P) \stackrel{def}{=} \mathcal{FI}(P) \setminus \{x\}$, $\mathcal{FI}(\{X(\tilde{x}) = by.P\}(\tilde{a})) \stackrel{def}{=} \{b[\tilde{a}/\tilde{x}]\}$, $\mathcal{FI}(P, Q) \stackrel{def}{=} \mathcal{FI}(P) \cup \mathcal{FI}(Q)$, *and if P is in other syntactic forms we have* $\mathcal{FI}(P) \stackrel{def}{=} \emptyset$. ∎

Structural Equivalence

The following structural rules define equivalence over syntactic representation. It was Milner [22] who first introduced the set of structural rules for formulation of the π-calculus and successfully utilized it. The spirit is that we let tacit and static semantics of various constructors represented in the structural rules, while expressing truly dynamic aspects of the system by transition rules. For port-passing calculi, structural rules play an essential role for tractable formulation of transition rules. The set of rules below basically follow Milner's construction, but are substantially weakened to make computational aspects as explicit as possible (it is even weaker than $\equiv$ in [12]).

Definitions 2.8 Structural Equivalence. $\equiv$ *is the smallest equivalence relation over terms defined by:*

(i) $P \equiv Q$ *(where P is α-convertible to Q)*

(ii) $(P, Q), R \equiv P, (Q, R)$

(iii) $|x|P, Q \equiv |x|(P, Q) \qquad (x \notin \mathcal{FN}(Q))$

(iv) $P, \leftarrow av \equiv \leftarrow av, P$

(v) $\{X(\tilde{x}) = yz.P\}(\tilde{a}) \equiv (yz.P)[\tilde{a}/\tilde{x}][\{X(\tilde{x}) = yz.P\}/X]$

(vi) $P \equiv Q$ *then* $P, R \equiv Q, R$ *and* $R, P \equiv R, Q$ *and* $|x|P \equiv |x|Q$. ∎

Note that $\equiv$ preserves basic properties of terms, i.e.

Lemma 2.9 For any P, Q, x, v,

(i) $P \equiv Q \quad \Rightarrow \quad \mathcal{FN}(P) \equiv \mathcal{FN}(Q) \ \wedge \mathcal{FI}(P) \equiv \mathcal{FI}(Q)$.

(ii) $P \equiv Q \ \Rightarrow \ P[v/x] \equiv Q[v/x]$. ∎

Proof Based on the inductive properties of $\mathcal{FN}(P)$, $\mathcal{FI}(P)$, and $[v/x]$. ∎

Another fact is that we can always transform a term expression into what we call a *normal form representation.*

Proposition 2.10 Let us call terms of the forms $\leftarrow av$, $ax.P$, and Λ, *primitive terms.* Then, for any $P \in \mathbf{C}$, we have $P' \equiv P$ such that, for some $m \geq 0$,

$$P' \stackrel{def}{=} |\tilde{w}|(P_1, P_2, ..., P_m)$$

where $P_1, ..P_m$ denote primitive terms. We call such P' (usually not unique) a *normal form representation* of P. Above we say $P_1, ..P_m$ are *active* in P'. ∎

Proof By simple application of structural rules based on the structure of terms. ∎

In the proposition above, we can write $P_1, P_2, ...$ etc. without any ambiguity (up to $\equiv$), due to the rule (ii) of Definition 2.8. Also note that, with P in a normal form, $\mathcal{FI}(P)$ is by definition the set of free input port names of active receptors in P.

In reasoning about terms in normal form representation, the following notation is convenient.

Conventions 2.11 Γ, Δ, *etc. range over a sequence of concurrent composition of primitive terms.* ∎

The Transition

The essence of the notion of asynchronous communication lies in that, at the time of consumption of a message, its "sender" is not participating in the event any more. Intuitively, then, the sender side *cannot* observe how the messages it sends will be treated by the other party. This changes the foregoing semantic scenario based on "experiments by the outside synchronous experimenter" (cf. [17]) into the new one with an asynchronous experimenter, where one can always proceed one's experiments as far as sending to the target system is concerned, while generation of messages from the target is beyond his control. This notion is formally represented in our labelled transition relation.

We first introduce *labels* for the transition.

Definitions 2.12 *Labels. The set of labels* $\boldsymbol{L}$ *for interaction is given by the following abstract syntax.*

$$\boldsymbol{L} \;=\; \tau \mid \downarrow \boldsymbol{NN} \mid \uparrow \boldsymbol{NN}\,. \quad ∎$$

Conventions 2.13 $l, l', ..$ *range over* $\boldsymbol{L}$. *Sometimes* $\downarrow\alpha$, $\uparrow\beta$, *etc. range over* $\boldsymbol{L}$. ∎

Definitions 2.14

(i) $l[v/x]$ *is defined as in the case of term expressions.*

(ii) $\mathcal{N}(l)$ *is the set of names occurring in labels, where* $\mathcal{N}(\tau) = \emptyset$.

(iii) Two functions on labels are defined as: $\pi_1(\uparrow av) = \pi_1(\downarrow av) = \{a\}$, $\pi_2(\uparrow av) = \pi_2(\downarrow av) = \{v\}$, *and* $\pi_1(\tau) = \pi_2(\tau) = \emptyset$. ∎

From the viewpoint of "semantics by experiments", we can give the following intuitive meaning to three kinds of labels.

(1) τ denotes the internal computation, i.e. communication occurring inside the configuration, thus unseen from the outside.

(2) $\downarrow av$ means that the experimenter sends a message to the configuration.

(3) $\uparrow av$ means the experimenter receives such a message.

Now the *interaction relation* defined below, shows how a configuration, after an experiment or an observation, evolves to the next stage. The element of the relation is of the form (P, l, P'), written as $P \stackrel{l}{\longrightarrow} P'$.

Definitions 2.15 *Asynchronous Interaction.* Let $\tilde{w} \backslash v$ denote a sequence of names where the occurrences of v are taken away from $\tilde{w}$, $\{\tilde{w}\}$ a set of names in the sequence $\tilde{w}$. Then *interaction of terms*, denoted $\stackrel{l}{\longrightarrow}$, is the smallest relation inferred by:

$$IN: \quad |\tilde{w}|(\Gamma, \Delta) \stackrel{\downarrow av}{\longrightarrow} |\tilde{w}|(\Gamma, \leftarrow av, \Delta) \qquad (a, v \notin \{\tilde{w}\})$$

$$OUT: \quad |\tilde{w}|(\Gamma, \leftarrow av, \Delta) \stackrel{\uparrow av}{\longrightarrow} |\tilde{w} \backslash v|(\Gamma, \Delta) \qquad (a \notin \{\tilde{w}\})$$

$$COM: \quad |\tilde{w}|(\Gamma, \leftarrow av, ax.P, \Delta) \stackrel{\tau}{\longrightarrow} |\tilde{w}|(\Gamma, P[v/x], \Delta)$$

$$STRUCT: \quad \frac{P_1' \equiv P_1, \ P_1 \stackrel{l}{\longrightarrow} P_2, \ P_2 \equiv P_2'}{P_1' \stackrel{l}{\longrightarrow} P_2'}$$

where in COM we call $\leftarrow av, ax.P$ a *redex pair* or simply a redex. ∎

Note that:

(1) The presentation of the transition rules is surprisingly simpler than its first version in [12] (cf. [22]). The use of normal form representation and the lack of bound labels are the main source of its simplicity. However it is consistent with the original formulation, as shown in Appendix A.

(2) Intuitively, the transition system defines behaviour of a configuration as asynchronous interaction with the outside world or an experimenter. In this regard **IN** rule is essential in that it directly represents the asynchronous character of experiments. It says that the experimenter can send *any* messages he wants at *any* occasion. The notion is in sharp contrast with a usual input rule in process calculi, where the experiment can proceed only when some agent in the configuration is ready to interact with the experimenter's action (cf. Definition 4.1 later).

(3) From *computational* point of view, only τ transition is substantial, defining the mechanism of computation in this formalism. In this regard we may say that IN and OUT rules define *semantic* transitions (cf. [22, 12]). Thus consistency of these semantic rules with the computational rule becomes an intriguing question (see [14] for more discussions).

The second point is represented in the following proposition.

Proposition 2.16 For any $P \in \mathbf{C}$ and a, v, $\quad P \xrightarrow{\downarrow av} (P, \leftarrow av)$. ∎

Several other facts regarding $\xrightarrow{l}$ (cf. Lemmas 1-3 in [20]) will be useful for our later discussions. The first shows how free names in the configuration can be affected by transitions.

Lemma 2.17 *Free name lemma for* $\xrightarrow{l}$.

$$\begin{array}{lll}
(1)\ P \xrightarrow{\downarrow av} P' & \Rightarrow & \mathcal{FN}(P) \cup \{a, v\} = \mathcal{FN}(P') \\
(2)\ P \xrightarrow{\uparrow av} P' & \Rightarrow & \mathcal{FN}(P) \cup \{v\} = \mathcal{FN}(P') \cup \{a, v\} \\
(3)\ P \xrightarrow{\tau} P' & \Rightarrow & \mathcal{FN}(P) \supset \mathcal{FN}(P')
\end{array}$$ ∎

Proof Easy case analysis based on name occurrences in the initial and derived terms, using Lemma 2.9 for the rule $STRUCT$. The addition of $\{v\}$ in (2) corresponds to the possible removal of a name in the binding in OUT rule of Definition 2.15. ∎

The second lemma shows how substitution affects transitions.

Lemma 2.18 *Substitution lemma for* $\xrightarrow{l}$. For any P, P' and l,

$$P \xrightarrow{l} P' \;\Rightarrow\; P[v/x] \xrightarrow{l[v/x]} P'[v/x] \ .$$

with if $x \notin \mathcal{FN}(P)$ then $v \notin \mathcal{FN}(P)$. ∎

Proof By checking each transition rule and using Lemma 2.9. ∎

3 The Asynchronous Bisimulation

This section introduces the equivalence relation called *asynchronous bisimulation* and studies its basic properties.

Bisimulation in the Context of Asynchronous Interaction

In the synchronous communication setting of process calculi, several equivalence notions have been proposed, i.e. bisimulation [21, 27], failure equivalence (or testing equivalence) [11, 23], and trace equivalence (for comprehensive treatment, see [1]). All of them can be defined over an arbitrary labeled transition system. Yet it is both surprising and pleasing that these notions are proven to be quite robust even in the setting of asynchronous interaction, for the underlying operational context is quite different here. For example, just as in the synchronous setting, the traces are not adequate to capture the operational behaviour of systems.

Definitions 3.1 Traces[2]. *Let $\tilde{l}\backslash l_0$ be the result of taking away all occurrences of l_0 from a sequence of labels $\tilde{l}$. Then the set of (weak)* traces *of a term P, denoted $tr\langle P\rangle$, is given as follows.*

$$tr\langle P\rangle \stackrel{def}{=} \{l_1..l_n\backslash\tau \mid \exists P_1..P_n.\ P \xrightarrow{l_1} P_1 \xrightarrow{l_2} .. \xrightarrow{l_n} P_n\} \ \cup\ \{\varepsilon\} \quad \blacksquare$$

An example which shows traces do *not* characterize the behaviour of communicating agents can be given using the nondeterministic summation operator defined below.

Definitions 3.2 *Nondeterministic summation.* With $c \notin \mathcal{FN}(P) \cup \mathcal{FN}(Q)$,

$$P \oplus Q \stackrel{def}{=} |c|(\leftarrow c,\ c.P,\ c.Q)\ . \quad \blacksquare$$

With this definition, $P \oplus Q$ behaves like P or like Q nondeterministically. Easily we have:

Proposition 3.3 $tr\langle P \oplus Q\rangle \stackrel{def}{=} tr\langle P\rangle \cup tr\langle Q\rangle$. $\blacksquare$

Hence:

Fact 3.4 $tr\langle \leftarrow av \oplus (\leftarrow av, \leftarrow bw)\rangle = tr\langle \leftarrow av, \leftarrow bw\rangle$. $\blacksquare$

But they should be differentiated since, put in a context

$$C[\cdot] \stackrel{def}{=} |ab|(ax.(by. \leftarrow xy),\ [\cdot]),$$

one may or may not emit $\leftarrow vw$ while the other surely does so. Thus the standard argument for the equivalence relation stronger than traces (cf. e.g. [17]) is still valid in the asynchronous setting, leading us to the definition of *asynchronous bisimulation*.

The Bisimulation

To formulate the bisimulation for our calculus, we need the notion of soundness of labels to avoid unnecessary collision between a newly created name and free names of terms.

Definitions 3.5 A label l of P is sound with respect to Q if and only if either $l = \tau$, $l = \downarrow\alpha$, or $l = \uparrow av$ with either $v \in \mathcal{FN}(P)$ or $v \notin \mathcal{FN}(P) \cup \mathcal{FN}(Q)$. $\blacksquare$

Also let us have:

Definitions 3.6 $P \stackrel{\hat{l}}{\Longrightarrow} Q$ is defined as $P \xrightarrow{\tau}{}^{*} \xrightarrow{l} \xrightarrow{\tau}{}^{*} Q$ with $l \neq \tau$ and $P \xrightarrow{\tau}{}^{*} Q$ with $l = \tau$, where $(\mathcal{R})^*$ denotes the reflexive transitive closure of a relation $\mathcal{R}$. $\blacksquare$

Now we can define the notion of *asynchronous bisimulation* as follows[3].

Definitions 3.7 *Asynchronous bisimulation*[4]. P_1 and Q_1 are asynchronously bisimilar, denoted by $P_1 \approx_a Q_1$, if and only if $(P_1, Q_1) \in \mathcal{R}$ with

[2] cf. Definition 3.27 later.

[3] *Weak* asynchronous bisimulation, to be strict.

[4] We replace Definition 10 in [12] with the present one.

(i) For any $(P,Q) \in \mathcal{R}$, whenever $P \xrightarrow{l} P'$ where l is sound with respect to Q, there exists Q' such that $Q \stackrel{\hat{l}}{\Longrightarrow} Q'$ and $(P',Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ is symmetric[5]. ∎

We call $\mathcal{R}$ as above an *asynchronous bisimulation*. Then $\approx_a$ (the *asynchronous bisimilarity*) is the union of all the bisimulations. The closure property of bisimulation is useful for verifying various theoretical and practical properties.

The below shows several examples of bisimilar and non-bisimilar terms. (ii) is the example in Fact 3.4. In (iii) we see a pair which cannot be equated by the bisimilarity but still seem indistinguishable from one another in any context (cf. Proposition 3.30).

Examples 3.8 *Examples of asynchronous (non-)bisimilarity.*

(i) $|c|(cx.P) \ \approx_a \ |a|(\leftarrow av) \ \approx_a \ \Lambda$.

(ii) $\leftarrow av \oplus (\leftarrow av, \leftarrow bw) \ \not\approx_a \ \leftarrow av, \leftarrow bw$.

(iii) $(\leftarrow av_1, \leftarrow bv_2) \oplus (\leftarrow av_1, \leftarrow cv_3) \ \not\approx_a \ \leftarrow av_1, \ \leftarrow bv_2 \oplus \leftarrow cv_3$. ∎

For other examples and applications of the bisimulation, see [12, 13, 14].

Finally, we need to extend $\approx_a$ to make it applicable to subexpressions in general. The extended $\approx_a$ can equate two expressions in the body of recursively defined receptors where structure variables may occur free. With this extension, we can discuss equivalence between any subexpressions of legitimate terms.

Definitions 3.9 Given $\{X(\tilde{x}) = yz.P\}(\tilde{a})$ and $\{X(\tilde{x}) = yz.Q\}(\tilde{a})$, we define

$$P \approx_a Q \quad \Leftrightarrow_{def} \quad P[\{X(\tilde{x}) = yz.P\}/X] \approx_a Q[\{X(\tilde{x}) = yz.Q\}/X]$$

and similarly for any subexpressions of P and Q. ∎

Now we are ready to discuss basic properties of $\approx_a$.

Substitutivity of Term Expressions

In CCS and other process calculi with general summation, the bisimulation is not a congruence relation because of the pre-emptive τ transition. That is, though we have $\tau.\Lambda \approx \Lambda$, $\tau.\Lambda + P \approx P$ does not usually hold. Our system lacks summation as a primitive[6], so that we may hope $\approx_a$ is preserved within any context. Name substitution presents subtle issues, but the following development validates the congruence of $\approx_a$.

First, Definition 3.7 almost directly tells us that:

Proposition 3.10 $\approx_a$ is an equivalence relation. That is, it is symmetric, reflexive and transitive. ∎

[5] The requirement for symmetry (cf. [27]) can be weakened as shown in [22], but no difference comes about as a semantic theory.

[6] which does not mean that it cannot express branching structures [12, 13].

To show that $\approx_a$ is closed under every syntactic construction, one essential fact is that $\approx_a$ is preserved under concurrent composition with arbitrary messages. The form of composition itself is also often used in proving or disproving the asynchronous bisimilarity (cf. Proposition 3.15), which reflects its special position in our semantic theory (also see Proposition 3.19).

Proposition 3.11 With M being an arbitrary sequence of concurrent composition of messages, i.e. $M \equiv (\leftarrow a_1 v_1, \leftarrow a_2 v_2, .., \leftarrow a_n v_n)$ $(n \geq 0)$, we have

$$P \approx_a Q \quad \Rightarrow \quad (P, M) \approx_a (Q, M) \qquad \blacksquare$$

Proof By proving that the relation $\mathcal{R} \stackrel{def}{=} \{((P, M),\ (Q, M)) \mid P \approx_a Q\}$ is an asynchronous bisimulation. The only non-trivial case is when $(P, M) \xrightarrow{\tau} P''$ and one of the redex for this τ transition is in M. Then there should be a message, say $\leftarrow av$, in M, so that $M \equiv (M', \leftarrow av)$ and that $P \xrightarrow{\downarrow av} (P, \leftarrow av) \xrightarrow{\tau} P'$ with $P'' \equiv (P', M')$. But because $P \approx_a Q$, we have, for some $Q' \approx_a P'$, $Q \stackrel{\downarrow av}{\Longrightarrow} Q'$, and $((P', M'),\ (Q', M')) \in \mathcal{R}$ as required. $\blacksquare$

Note that the converse is not true, that is,

Proposition 3.12 $(P, \leftarrow av) \approx_a (Q, \leftarrow av)$ does not imply $P \approx_a Q$. $\blacksquare$

Proof Take a pair

$$P \stackrel{def}{=} \leftarrow b, b.Q\ , \qquad Q \stackrel{def}{=} \{X(xy) = xz.(\leftarrow xz, \leftarrow y, y.X(xy))\}(ab)$$

and show that $(P, \leftarrow av) \approx_a (Q, \leftarrow av)$ but $P \not\approx_a Q$. $\blacksquare$

The above implies that pending messages in a configuration play an essential role in our semantic framework.

Now the substitutivity for concurrent composition and scope restriction can easily be obtained.

Proposition 3.13 For any P, Q, R and x,

(i) $P \approx_a Q \quad \Rightarrow \quad |x|P \approx_a |x|Q$.
(ii) $P \approx_a Q \quad \Rightarrow \quad (P, R) \approx_a (Q, R)$ $\blacksquare$

Proof

(i) Take the relation $\{(|x|P,\ |x|Q) \mid P \approx_a Q\} \cup \approx_a$. For input, we use Proposition 3.11 above.
(ii) Take the relation $\{((P, R),\ (Q, R)) \mid P \approx_a Q\}$ where R is an arbitrary term. We use (i) above for τ transitions. $\blacksquare$

What is subtle is the substitutivity of expressions for a body of a receptor. The following non-trivial lemma, established by extensively using Lemmas 2.6 and 2.18, is essential for proving it.

Lemma 3.14 *For any P, Q, v and x,* $P \approx_a Q \;\Rightarrow\; P[v/x] \approx_a Q[v/x]$. ∎

Proof See Appendix B. ∎

Now we can show that the asynchronous bisimulation is indeed preserved by the prefix operation in constructing a receptor, and consequently also by recursively defined receptors.

Proposition 3.15 For any P, Q and a, x,

$$P \approx_a Q \quad\Rightarrow\quad ax.P \approx_a ax.Q \;. \;\blacksquare$$

Proof We take the relation

$$\mathcal{R} = \{((ax.P,\ M),\ (ax.Q,\ M))\} \cup \approx_a$$

where $P \approx_a Q$ and M is zero or more messages. For input and output, the resulting pair is still in $\mathcal{R}$. For τ transition, we should consider the pair $((P[v/x], M'),\ (Q[v/x], M'))$ where $P \approx_a Q$. But we know by Lemma 3.14 and Proposition 3.13 that they are in $\approx_a$, i.e. in $\mathcal{R}$. Thus $\mathcal{R}$ is a bisimulation, concluding the proof. ∎

Proposition 3.16 Given any well-formed $\{X(\tilde{x}) = yz.P\}(\tilde{a})$ and $\{X(\tilde{x}) = yz.Q\}(\tilde{a})$,

$$P \approx_a Q \quad\Rightarrow\quad \{X(\tilde{x}) = yz.P\}(\tilde{a}) \approx_a \{X(\tilde{x}) = yz.Q\}(\tilde{a}) \;.$$

with $P \approx_a Q$ as defined in Definition 3.9. ∎

Proof With $\Upsilon_1 \stackrel{def}{=} \{X(\tilde{x}) = yz.P\}$ and $\Upsilon_2 \stackrel{def}{=} \{X(\tilde{x}) = yz.Q\}$, and also using the structural rule for recursion (cf. Definition 2.8), what we should prove is

$$P[\Upsilon_1/X] \approx_a Q[\Upsilon_2/X] \quad\Rightarrow\quad (yz.P[\Upsilon_1/X])[\tilde{a}/\tilde{x}] \approx_a (yz.Q[\Upsilon_2/X])[\tilde{a}/\tilde{x}] \;.$$

It is easy to see that the above case is reduced to the substitutivity of expressions in prefix operation and substitution, already proved in Proposition 3.15 and Lemma 3.14. ∎

Summarising Propositions 3.13, 3.15 and 3.16, we have:

Proposition 3.17 $\approx_a$ *is a congruence relation.* ∎

Thus two asynchronously bisimilar expressions can be used interchangeably as a part of any composite systems as far as we are based on our asynchronous semantic theory. Moreover another study not presented here shows that such a pair *does* behave quite compatibly in any context (in the sense that they will affect their environment identically). In this context, discussions in Section 4 provide an interesting comparison between the asynchronous theory and its synchronous counterpart in regard of the discernibility of a pair of terms apparently indistinguishable in their behaviour (see also [14] for related discussions).

Alternative Characterizations

In the following, we give a series of characterization results for the asynchronous bisimilarity. They shed light on various aspects of this bisimulation, as well as providing us with alternative proof methods.

Input-less Bisimulation

The following definition and proposition are based on the observation that the preservation of equivalence under composition with arbitrary messages is the crucial element for $\approx_a$.

Definitions 3.18 We define $\xrightarrow{\Uparrow\alpha,\tau}$ where l is the subset of $\xrightarrow{l}$ as the smallest relation inferred by the rules in Definition 2.15 minus IN rule. ∎

Proposition 3.19 $P_1 \approx_a Q_1$ if and only if $(P_1, Q_1) \in \mathcal{R}$ where for any $(P, Q) \in \mathcal{R}$ we have, in terms of $\xrightarrow{\Uparrow\alpha,\tau}$, whenever $P \xrightarrow{l} P'$ where l is sound with respect to Q, there exists Q' such that $Q \overset{\hat{l}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$, and:

(i) $\mathcal{R}$ is symmetric.
(ii) For any P, Q and $\leftarrow av$, $(P,\ Q) \in \mathcal{R} \quad \Rightarrow \quad ((P, \leftarrow av),\ (Q, \leftarrow av)) \in \mathcal{R}$. ∎

Proof The clause (ii) above is exactly what IN transition means for the bisimulation property. ∎

The result is a repercussion of Proposition 2.16 which says every term is capable of performing *any* input transition. Thus, naturally, to check whether one party can simulate an input transition of the other is vacuous as a verification of the capability itself, but is significant only in terms of the constraint that *the resulting configurations are still in the closure*, hence Proposition 3.19. The fact is well reflected in the construction of the relation in e.g. the proof of Propositions 3.15.

Restricting Input Labels

Another and possibly more enlightening characterization comes from a question asking whether we really need all the possible labels for input. As we noted, the significance of IN rule does not lie in expressing the concerned term's capability (to perform the transition), but as a probe to know potential behaviour of the expression by putting a message in it (cf. Proposition 2.16). The following result is significant in this respect. The proof is direct from the definition of $\xrightarrow{l}$ and omitted.

Proposition 3.20 *Suppose $P' \equiv (P, \leftarrow av)$, or equivalently $P \xrightarrow{\Downarrow av} P'$, with $a \notin \mathcal{FI}(P)$. Then the following holds.*

(i) $P' \xrightarrow{\Downarrow bw} Q' \quad \Leftrightarrow \quad P \xrightarrow{\Downarrow bw} Q \wedge Q' \equiv (Q, \leftarrow av)$.
(ii) $P' \xrightarrow{\Uparrow bw} Q' \quad \Leftrightarrow \quad P \xrightarrow{\Uparrow bw} Q \ \wedge\ Q' \equiv (Q, \leftarrow av) \ \vee \ \leftarrow bw \overset{def}{=} \leftarrow av \ \wedge\ Q' \equiv P$.
(iii) $P' \xrightarrow{\tau} Q' \quad \Leftrightarrow \quad P \xrightarrow{\tau} Q \wedge Q' \equiv (Q, \leftarrow av)$. ∎

These facts show that the possible transitions of a term after receiving a message whose target receptor is not active, is exactly as before (except a trivial case of the output of that very message). This observation leads us to the following result, which gives us another characterization of the bisimilarity.

Proposition 3.21 We say $P_1 \approx^{\star}_{a} Q_1$ when $(P_1, Q_1) \in \mathcal{R}$ with

(i) For any $(P, Q) \in \mathcal{R}$, whenever $P \stackrel{l}{\longrightarrow} P'$ where l is sound with respect to Q and, if l is $\downarrow av$, $a \in \mathcal{FI}(P)$, then there exists Q' such that $Q \stackrel{\hat{l}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ is symmetric.

Then $P_1 \approx_a Q_1$ if and only if $P_1 \approx^{\star}_{a} Q_1$. ∎

Proof
(⇒) By showing that $\approx_a$ is a relation satisfying the conditions.
(⇐) We construct a relation $\mathcal{R} \stackrel{def}{=} \{((P, M), (Q, M))\}$ where $P \approx^{\star}_{a} Q$ and, with $n > 0$ and $a_i \notin \mathcal{FI}(P) \cap \mathcal{FI}(Q)$, $M \equiv (\leftarrow a_1 v_1, .., a_n v_n)$, to show that it is a bisimulation. But Proposition 3.20 tells us that all the one-step transitions from $(P, \leftarrow a_1 v_1, .., a_n v_n)$ where $n \geq 1$ and $a_i \notin \mathcal{FI}(P)$, can be inferred from transitions starting from P except the trivial case of the output of one of $\leftarrow a_i v_i$'s. This establishes the closure property of $\mathcal{R}$. Alternatively, we can do without the construction by proving the analogue of Proposition 3.11 for $\approx^{\star}_{a}$, from which the result is immediate. ∎

A minor variant of the above proposition is:

Corollary 3.22 *If we change "$\mathcal{FI}(P)$" in Proposition 3.21 to "some $\mathcal{F} \supset \mathcal{FI}(P)$", the same holds.* ∎

Proof The argument of Proposition 3.20 is still valid. ∎

Finally we note that, though these results may seem to suggest that a more restricted kind of IN rule will suffice, we still need the full generality of IN rule for sound formulation of the asynchronous interaction semantics. Interested readers may consult Appendix C.

Bisimilarity as a Weak Theory

It is well-known that the weak bisimulation in general can be characterised by weak transitions, i.e. transitions which ignore τ's. Not only the same holds for the asynchronous bisimilarity (which is not a surprise), but it enables us to position the theory among other (asynchronous) weak theories.

Definitions 3.23 Weak asynchronous transitions. *A multi-step transition* $P \stackrel{l_1 l_2 .. l_n}{\Longrightarrow} Q$ $(n \geq 0)$ *with* $l_1, .., l_n \in \boldsymbol{L} \backslash \{\tau\}$ *is defined when*

$$P \stackrel{\tau}{\longrightarrow}^{*} \stackrel{l_1}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^{*} P_1 \stackrel{\tau}{\longrightarrow}^{*} \stackrel{l_2}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^{*} P_2 \cdots \stackrel{\tau}{\longrightarrow}^{*} \stackrel{l_n}{\longrightarrow} \stackrel{\tau}{\longrightarrow}^{*} P_n \stackrel{def}{=} Q$$

for some $P_1 .. P_n$. *Specifically,* $P \Longrightarrow Q$ *denotes* $P \stackrel{\tau}{\longrightarrow}^{*} Q$ *for some* Q. ∎

Conventions 3.24

(i) $t, t', ..$ *range over* $(\boldsymbol{L}\backslash\{\tau\})^*$.

(ii) $\mathcal{N}(t)$ *is the set of names occurring in* t. ∎

Definitions 3.25 A label t of P is sound with respect to Q if and only if, for each label $l_i = \uparrow av$ in t, either $v \in \mathcal{FN}(P)$ or $v \notin \mathcal{FN}(P) \cup \mathcal{FN}(Q)$. ∎

Proposition 3.26 Characterization of asynchronous bisimilarity by weak transitions. $P_1 \approx_a Q_1$ *if and only if* $(P_1, Q_1) \in \mathcal{R}$ *where*

(i) For any $(P, Q) \in \mathcal{R}$, *whenever* $P \stackrel{t}{\Longrightarrow} P'$ *where* t *is sound with respect to* Q, *there exists* Q' *such that* $Q \stackrel{t}{\Longrightarrow} Q'$ *and* $(P', Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ *is symmetric.* ∎

Weak (asynchronous) *theories* are those which can be characterized by the weak asynchronous transitions (possibly with the soundness condition) and abstract nature of the relation. We will show two such examples, i.e. asynchronous trace equivalence and asynchronous failure equivalence. Let $P \stackrel{t}{\Longrightarrow}$ denote $P \stackrel{t}{\Longrightarrow} P'$ for some P'.

Definitions 3.27 *Asynchronous trace equivalence.* We say P and Q are (asynchronously) trace equivalent, denoted by $P \approx_{ta} Q$, if and only if

(i) $P \stackrel{t}{\Longrightarrow} \quad \Rightarrow \quad Q \stackrel{t}{\Longrightarrow}$ (for any t of P sound for Q).

(ii) $Q \stackrel{t}{\Longrightarrow} \quad \Rightarrow \quad P \stackrel{t}{\Longrightarrow}$ (for any t of Q sound for P). ∎

Definitions 3.28 *Asynchronous failure equivalence.* Let us first formulate the notion of failures as follows. Given a pair P and Q, we say P has a failure $(t, \mathcal{L})$, where $\mathcal{L} \subset \mathbf{L}\backslash\{\tau\}$ if and only if, for some P',

(1) $P \stackrel{t}{\Longrightarrow} P'$.

(2) For all $l \in \mathcal{L}$, we have $\neg\, P' \stackrel{l}{\Longrightarrow}$.

Then P and Q are failure equivalent, written $P \approx_{fa} Q$, if and only if they possess the identical failures, where the equality of the failure set is relativised by the soundness condition in Definition 3.25 [7]. ∎

The failure equivalence is more generous than the usual one, in that it does not care about the divergence. We can of course give a more strict version[8], but here we will be content with the fact that this way of formulation allows us easier comparison with other theories.

Proposition 3.29

[7] E.g. even if $l \in \mathcal{L}$ and $Q \stackrel{t}{\Longrightarrow} Q' \stackrel{l}{\Longrightarrow}$ above, we neglect the difference if l of Q' is not sound with respect to P'.

[8] However, our current stipulation is that this weak failure equivalence is more natural in the asynchronous setting.

(i) $P \approx_a Q \quad \Rightarrow \quad P \approx_{fa} Q$
(ii) $P \approx_{fa} Q \quad \Rightarrow \quad P \approx_{ta} Q.$ ∎

Proof

(i) Suppose that $P \approx_a Q$ and that P owns a failure $(t, \mathcal{L})$ with $P \stackrel{t}{\Longrightarrow} P'$ where t is sound with respect to Q. Clearly there is Q' such that $Q \stackrel{t}{\Longrightarrow} Q'$ and $P' \approx_a Q'$. But if $Q' \stackrel{l}{\Longrightarrow}$ with $l \in \mathcal{L}$ then $P' \stackrel{l}{\Longrightarrow}$ (assuming w.l.o.g. l is sound with respect to P'), which contradicts the assumption. Thus Q owns the same failure as P relative to the soundness condition. Similarly for the symmetric case.
(ii) Immediate from the definitions. ∎

What is more interesting is that these inclusions are indeed *strict*.

Proposition 3.30

(i) $\approx_{fa} \not\supset \approx_{ta}$.
(ii) $\approx_a \not\supset \approx_{fa}$. ∎

Proof

(i) By (ii) of Example 3.8 which has a family of differentiating failures $(\uparrow av,\ \mathcal{L})$, where $\mathcal{L}$ includes $\uparrow bw$.
(ii) By (iii) of Example 3.8. ∎

Thus the asynchronous bisimulation is positioned as strictly stronger than the failure equivalence, and the latter is again strictly stronger than the trace equivalence.

In the setting of synchronous interaction, the notion of *deadlocks* is essential in the operational understanding of failure equivalence and bisimilarity. Interestingly, the notion is still significant in the asynchronous setting, if seen in the (abstract) framework of distributed computing. A simple example is enough to understand the situation.

$$P \stackrel{def}{=} \leftarrow av \oplus \leftarrow bw \oplus (\leftarrow av, \leftarrow bw) \quad Q \stackrel{def}{=} \leftarrow av, \leftarrow bw.$$

A failure $(\varepsilon, \mathcal{L})$ belongs to P, but not to Q, with $\mathcal{L} \not\supset \{\uparrow av, \uparrow bw\}$. Or, alternatively, if P's summand is non-deterministically selected, bisimulation cannot hold after $\uparrow av$.

Now, interpreted in the synchronous setting, the property of P means that the environment should offer *both* the a-input and the b-input, to ensure that P can proceed computation. However, in the context of asynchronous interaction, they embody the fact that, if the environment waits with both ports ready, it will surely get *either* $\leftarrow av$ *or* $\leftarrow bw$. If not, however, it may possibly get nothing. But we should notice that this *is* the basic element of deadlocks in the real distributed computing environment! Thus reinterpretation from a different perspective recovers the operational significance of the original notion developed in an apparently dissimilar setting.

The above discussions connote the soundness of applying the new semantic framework to problems in the real-world distributed computing environments. Such applications of the asynchronous interaction semantics should be important and challenging research topics in coming days.

4 Asynchronous Theories through Synchronous Theories

This section studies the proper characteristics of the asynchronous bisimilarity in terms of the difference from its synchronous counterpart, referring to the related results in other weak theories.

The Synchronous Theory

To understand the proper characteristics of $\approx_a$, comparison with its synchronous counterpart is helpful. The synchronous theory is, though restricted to our syntactic construction, embodiment of classic behavioural equivalence for processes, hence to know the difference between two theories should be instructive to appreciate the uniqueness of our asynchronous theory.

The synchronous theory is defined with a different transition relation with another IN rule, which is more in the spirit of the foregoing synchronous experiments framework and indeed has almost the same functionality as the one found in [22], except ours uses the so-called *early instantiation* rather than the late one[9].

Definitions 4.1 *Synchronous interaction of terms*, denoted by $\xrightarrow{l}_s$, is the smallest relation inferred by the same rules as Definition 2.15, with $\xrightarrow{l}$ replaced by $\xrightarrow{l}_s$ except IN rule which is reformulated as

$$IN_s: \quad |\tilde{w}|(\Gamma, ax.P, \Delta) \xrightarrow{\downarrow av}_s |\tilde{w}|(\Gamma, P[x/v], \Delta) \quad (a, v \notin \{\tilde{w}\}) \quad \blacksquare$$

Thus the synchronous "input" experiments can only proceed when some receptors are ready to receive them, in other words (cf. Proposition 2.16):

Proposition 4.2 Let us write $P \xrightarrow{l}_s$ when $P \xrightarrow{l}_s P'$ for some P'. Then

$$P \xrightarrow{\downarrow av}_s \quad \Leftrightarrow \quad a \in \mathcal{FI}(P) \ . \quad \blacksquare$$

As an elementary comparison between $\xrightarrow{l}_s$ and $\xrightarrow{l}$, it is easy to see the following.

Proposition 4.3 For a specific l, we let $\xrightarrow{l}$ represent all (P, Q) such that (P, l, Q) is in the transition relation. Then, for any α, we have

$$\begin{aligned} \xrightarrow{\uparrow\alpha} &= \xrightarrow{\uparrow\alpha}_s \\ \xrightarrow{\tau} &= \xrightarrow{\tau}_s \\ \xrightarrow{\downarrow\alpha}\xrightarrow{\tau} &\supset \xrightarrow{\downarrow\alpha}_s \ . \quad \blacksquare \end{aligned}$$

[9] The consistency of early instantiation scheme with reduction relation ($\xrightarrow{\tau}$) which we regard more fundamental than transition relation is basis of our choice, though we do not present the argument here. In passing, however, let us note that "instantiation" only occurs in COM rule in $\xrightarrow{l}$ (hence no dispute on instantiation schemes), and that the synchronous transition using early instantiation is the only natural candidate for comparison with this $\xrightarrow{l}$ (cf. Proposition 4.3).

Proof Direct from the definition. For the last subset relation, we have a transition

$$|\tilde{w}|(\Gamma, ax.P, \Delta) \xrightarrow{\downarrow av} \xrightarrow{\tau} |\tilde{w}|(\Gamma, P[v/x], \Delta)$$

which effectively captures $\xrightarrow{\downarrow av}_s$. ∎

To define the synchronous bisimulation, derived transition is introduced as in the case of the asynchronous one (the soundness condition is the same as in Definition 3.5).

Definitions 4.4 $P \overset{\hat{l}}{\Longrightarrow}_s Q$ is defined as $P {\xrightarrow{\tau}_s}^* \xrightarrow{l}_s {\xrightarrow{\tau}_s}^* Q$ with $l \neq \tau$ and $P {\xrightarrow{\tau}_s}^* Q$ with $l = \tau$. ∎

We now define another equivalence relation for our system, this time based on $\xrightarrow{l}_s$. The new equivalence is called *synchronous bisimulation* and is given as follows.

Definitions 4.5 Synchronous bisimulation. *P_1 and Q_1 are synchronously bisimilar, denoted by $P_1 \approx_s Q_1$ if and only if $(P_1, Q_1) \in \mathcal{R}$ with*

(i) For any $(P, Q) \in \mathcal{R}$, whenever $P \xrightarrow{l}_s P'$ which is sound with respect to Q, there is Q' such that $Q \overset{\hat{l}}{\Longrightarrow}_s Q'$ and $(P', Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ is symmetric. ∎

The following holds for $\approx_s$ in our formal system. We can prove that the same is also true in the fragment of the π-calculus presented in [22] (consult [14] for the proof). In the full calculus, we have a counter example in [19] [10].

Proposition 4.6 $\approx_s$ is a congruence relation. ∎

Proof Proofs for restriction and composition are more straightforward than those for $\approx_a$. For substitution, it is along the same line of $\approx_a$ except in the case of input, where we use a slightly different argument. The cases for prefix and recursion can be handled in the same way as in $\approx_a$. ∎

Later it becomes meaningful to understand this bisimilarity as a weak theory, i.e. as a theory based on the weak synchronous transition.

Definitions 4.7 Weak synchronous transitions. *A multi-step transition $P \overset{l_1 l_2 .. l_n}{\Longrightarrow}_s Q$ $(n \geq 0)$ with $l_1, .., l_n \in \boldsymbol{L} \backslash \{\tau\}$ is defined when*

$$P {\xrightarrow{\tau}_s}^* \xrightarrow{l_1}_s {\xrightarrow{\tau}_s}^* P_1 {\xrightarrow{\tau}_s}^* \xrightarrow{l_2}_s {\xrightarrow{\tau}_s}^* P_2 \cdots {\xrightarrow{\tau}_s}^* \xrightarrow{l_n}_s {\xrightarrow{\tau}_s}^* P_n \overset{def}{=} Q$$

for some $P_1..P_n$. Specifically, $P \Longrightarrow_s Q$ denotes $P {\xrightarrow{\tau}_s}^* Q$ for some Q. ∎

[10] The counterexample is: $x\bar{y} + \bar{y}x \approx x|\bar{y}$ but $(x\bar{y} + \bar{y}x)[y/x] \not\approx (x|\bar{y})[y/x]$ with $x \neq y$. We can find another example which uses matching, as in: $[x = y]P \approx [x = z]P$ but $([x = y]P)[x/y] \not\approx ([x = z]P)[x/y]$ where x, y, z are all distinct and $P \not\approx \Lambda$.

Then, with the same soundness condition as defined in 3.25, we have a characterization result for synchronous bisimilarity, corresponding to Proposition 3.26.

Proposition 4.8 Characterization of synchronous bisimilarity by weak transitions.
$P_1 \approx_s Q_1$ *if and only if* $(P_1, Q_1) \in \mathcal{R}$ *where*

(i) For any $(P, Q) \in \mathcal{R}$*, whenever* $P \stackrel{t}{\Longrightarrow}_s P'$ *which is sound with respect to* Q*, there exists* Q' *such that* $Q \stackrel{t}{\Longrightarrow}_s Q'$ *and* $(P', Q') \in \mathcal{R}$ *.*

(ii) $\mathcal{R}$ *is symmetric.* ∎

Other weak theories can be defined just as in the asynchronous case.

Definitions 4.9 *Synchronous trace equivalence.* We say P and Q are synchronously trace equivalent, denoted by $P \approx_{ts} Q$, if and only if, for any t,

(i) $P \stackrel{t}{\Longrightarrow}_s \quad \Rightarrow \quad Q \stackrel{t}{\Longrightarrow}_s$ (with t sound for Q).

(ii) $Q \stackrel{t}{\Longrightarrow}_s \quad \Rightarrow \quad P \stackrel{t}{\Longrightarrow}_s$ (with t sound for P). ∎

Definitions 4.10 *Synchronous failure equivalence.* Given a pair P and Q, we say P has a synchronous failure $(t, \mathcal{L})$, with respect to Q, where $\mathcal{L} \subset \mathbf{L} \backslash \{\tau\}$ if and only if

(1) $P \stackrel{t}{\Longrightarrow}_s P'$.

(2) For all $l \in L$, we have $\neg\ P' \stackrel{l}{\Longrightarrow}_s$.

Then P and Q are synchronously failure equivalent, written $P \approx_{fs} Q$, if and only if they possess the identical failures, where the equality of the failure set is relativised by the soundness condition in Definition 3.25. ∎

As the following proposition shows, here again bisimilarity is the most strict among three theories (cf. Propositions 3.29 and 3.30). We omit the proof, but only note that the *same* examples as used in Proposition 3.30 are effective for proving the strictness.

Proposition 4.11

(i) $\approx_{fs} \subset \approx_{ts}$ but $\approx_{fs} \not\supset \approx_{ts}$.

(ii) $\approx_s \subset \approx_{fs}$ but $\approx_s \not\supset \approx_{fs}$. ∎

Inclusion

The first result concerning the relationship between $\approx_a$ and $\approx_s$ shows that $\approx_a$ at least includes $\approx_s$, that is, the asynchronous theory is no less generous than the synchronous one. The result is almost immediate from the preceding propositions.

Proposition 4.12 For any P and Q, $\quad P \approx_s Q \quad \Rightarrow \quad P \approx_a Q$. ∎

Proof We show that, when $P \approx_s Q$ holds,

$$P \xrightarrow{l} P' \quad \Rightarrow \quad \exists\, Q' \; s.t.\; (\, Q \overset{\hat{l}}{\Longrightarrow} Q' \;\wedge\; P' \approx_s Q')$$

holds (note that the transition relation is *not* the synchronous transition relation but the asynchronous one). This tells us that $\approx_s$ is an asynchronous bisimulation, hence $\approx_s \subset \approx_a$.

When $l = \uparrow\alpha$ or $l = \tau$, the result is immediate because of Proposition 4.3.

In the case $l = \downarrow av$, we have $P \xrightarrow{\downarrow av} (P, \leftarrow av) \;\wedge\; Q \xrightarrow{\downarrow av} (Q, \leftarrow av)$. But because $\approx_s$ is a congruence relation, we have $(P, \leftarrow av) \approx_s (Q, \leftarrow av)$ and the proof is done. ∎

Thus we have shown that $\approx_a$ equates no less terms than $\approx_s$ does. The next question is whether it equates the same class of terms as $\approx_s$ or it equates strictly more than the synchronous theory. For the purpose, we should find out if there are any pair of expressions which are equivalent in $\approx_a$ but not so in $\approx_s$. The following shows that such a pair does exist.

Definitions 4.13 $\mathcal{I}(a)$ *which does nothing.* We define a family of *identity receptors* as

$$\mathcal{I}(a) \;\stackrel{def}{=}\; \{X(x) = xy.(\leftarrow xy, X(x))\}(a)$$

for an arbitrary port name a. ∎

Immediately we have

$$\mathcal{I}(a) \;\equiv\; ay.(\leftarrow ay,\, \mathcal{I}(a))$$

and also, for any $n \geq 0$,

$$\leftarrow av,\, \mathcal{I}(a) \;\; (\xrightarrow{\tau})^n \;\; \leftarrow av,\, \mathcal{I}(a)\ .$$

Thus the expression $\mathcal{I}(a)$ behaves as if it were nothing. And this nothingness of $\mathcal{I}(a)$ turns out to be important for our study of the difference between two equational theories.

Proposition 4.14 $\approx_s \subset \approx_a$ but $\approx_a \not\subset \approx_s$.

Proof The first half is Proposition 4.12.

For the second half, let us take the pair $(\mathcal{I}(a), \Lambda)$ (a can be arbitrary) and show that these two are not equivalent as far as $\approx_s$ goes, but *are* in the theory of $\approx_a$, so that it becomes a counterexample to $\approx_a \subset \approx_s$.

First, as $\mathcal{I}(a) \xrightarrow{\downarrow av}_s$ but $\Lambda \overset{\downarrow av}{\not\Longrightarrow}_s$ (of course Λ has no derivation whatsoever!), we know immediately $\mathcal{I}(a) \not\approx_s \Lambda$.

Second, take a relation $\mathcal{R} = (\ (\mathcal{I}(a), P),\ P\ )$, where P is any term in $\mathbf{C}$.

(1) In case $(\mathcal{I}(a), P) \xrightarrow{\downarrow av} (\mathcal{I}(a), P, \leftarrow av)$. Then clearly $P \xrightarrow{\downarrow av} (P, \leftarrow av)$ where $(\ (\mathcal{I}(a), P, \leftarrow av),\ (P, \leftarrow av)\ ) \in \mathcal{R}$. We can similarly verify when $(\mathcal{I}(a), P) \xrightarrow{\uparrow av} (\mathcal{I}(a), P')$.

(2) In case $(\mathcal{I}(a), P) \xrightarrow{\tau} Q$. The case $Q \equiv (\mathcal{I}(a), P') \wedge P \xrightarrow{\tau} P'$ is trivial. The only remaining possibility is $P \equiv (P', \leftarrow av)$ for some P'. Then

$$(\mathcal{I}(a),\ \leftarrow av,\ P') \quad \xrightarrow{\tau} \quad (\mathcal{I}(a),\ \leftarrow av,\ P') \quad \equiv \quad (\mathcal{I}(a),\ P)$$

As obviously $P \overset{\varepsilon}{\Longrightarrow} P$, this case holds.

(3) Because the symmetric case can be proved trivially, the relation $\mathcal{R} \cup \mathcal{R}^{-1}$ is a bisimulation. To know $\mathcal{I}(a) \approx_a \Lambda$, take $P \overset{def}{=} \Lambda$. ∎

Some remarks are due in terms of this inclusion result.

(1) Proposition 4.14 shows that the theory of $\approx_a$ thinks the term $\mathcal{I}(a)$ is nothing, while the theory of $\approx_s$ treats it as something different from nothing. This is semantic discussion. But then we should think about *computational* behaviour of this expression $\mathcal{I}(a)$, to find out that (while it may consume computational resources somehow) $\mathcal{I}(a)$ behaves as nothing in any computational context whatsoever. In other words, $\mathcal{I}(a)$ (for any a) *may never give any influence to its environment*, situated anywhere[11]. Thus, albeit intuitively, the computational meaning of $\mathcal{I}(a)$ is better captured in the theory of $\approx_a$.

(2) We have similar strict inclusion results between other pairs of weak theories, i.e. $\approx_{ts} \subset \approx_{ta}$ but $\approx_{ta} \not\subset \approx_{ts}$ and $\approx_{fs} \subset \approx_{fa}$ but $\approx_{fa} \not\subset \approx_{fs}$. It is easy to see that we can use the same pair of terms as in the proof of Proposition 4.14 for strictness. Thus the position of the pair of agents as that which tells difference between synchronous and asynchronous theories, seems somewhat universal. In fact we can verify two terms have the identical weak asynchronous transition graph; and any sound synchronous theory may not equate two terms whose traces are different.

We conclude this subsection by presenting two additional kinds of identity receptors. They are examples of those receptors which behave as nothing asynchronously but as something synchronously (i.e. which are asynchronously bisimilar to Λ but are not synchronously). There are infinite kinds of such receptors; it is known, however, that one can construct any such receptors by a certain syntactic formation rules from three basic ones, one of which is $\mathcal{I}(x)$ and the other two are as defined below. Of the two kinds, "eager" one will play an essential role in the $\mathcal{I}$ completion we will discuss soon.

Definitions 4.15 *Variants of Identity Receptors.*

(1) *A degenerated variant.* $\mathcal{I}_0(x) \overset{def}{=} xy. \leftarrow xy.$

(2) *An eager variant.* $\mathcal{I}_2(a) \overset{def}{=} \{X(x) = xy.(\leftarrow xy, X(y), X(x))\}(a).$

Let $\mathcal{I}_1(a) \overset{def}{=} \mathcal{I}(a)$ to conform to the above notations. ∎

Proposition 4.16 We have $\mathcal{I}_0(a) \approx_a \mathcal{I}_1(a) \approx_a \mathcal{I}_2(a) \approx_a \Lambda$, but any pair of these are *not* synchronously bisimilar. ∎

[11] When fairness is to be considered, the statement needs some modification.

$\mathcal{I}$ Completion

We already know that the relation $\approx_a$ is strictly wider than $\approx_s$, and that at least the identity receptors of various forms constitute the difference. As we noted the same holds for other weak theories. The exact difference between these two bisimilarities, however, is not clear yet. To study this, a method called $\mathcal{I}$ completion is suggestive.

The $\mathcal{I}$ completion is a method which syntactically transforms $\approx_a$ into $\approx_s$ by filling concerned configurations with enough identity receptors. It uses $\mathcal{I}_2(a)$ just defined, instead of $\mathcal{I}(a)$ in order to cope with a certain invariant condition[12], namely, covering of the free input ports of the configuration by identity receptors. More exactly, it transforms the set of terms which are asynchronously bisimilar and whose free names are bounded by some finite set, into synchronously bisimilar. The basic idea is to enable a term to emulate all the (essential) asynchronous transitions by synchronous transitions with the help of identity receptors. But then all the original synchronous transitions are emulated by the corresponding asynchronous transitions from the beginning (cf. Proposition 4.3), hence the desired result. The emulation is done at the level of the *weak* transitions, so that the operation transforms weak asynchronous theories in general into their synchronous counterparts.

Definitions 4.17 *The $\mathcal{I}$ Completion.* With $P \in \mathbf{C}$, let $\mathcal{F}$ be a finite set of port names. We define the function $\psi : \mathbf{C} \to \mathbf{C}$ as follows.

$$\begin{aligned}
\psi(\Lambda) &\stackrel{def}{=} \Lambda \\
\psi(\leftarrow av) &\stackrel{def}{=} \leftarrow av \\
\psi(X(\tilde{a})) &\stackrel{def}{=} X(\tilde{a}) \\
\psi(ax.P) &\stackrel{def}{=} ax.(\mathcal{I}_2(x), \psi(P)) \\
\psi(|x|P) &\stackrel{def}{=} |x|(\mathcal{I}_2(x),\ \psi(P)) \\
\psi(P,\ Q) &\stackrel{def}{=} \psi(P),\ \psi(Q) \\
\psi(\{X(\tilde{x}) = yz.Q\}(\tilde{a})) &\stackrel{def}{=} \{X(\tilde{x}) = yz.(\mathcal{I}_2(z), \psi(Q))\}(\tilde{a})
\end{aligned}$$

Then the $\mathcal{I}$ *Completion* of P for $\mathcal{F}$, denoted by $\Phi_{\mathcal{F}}(P)$, is given as

$$\Phi_{\mathcal{F}}(P) \stackrel{def}{=} \mathcal{I}_2(\mathcal{F}),\ \psi(P)$$

where, with $\mathcal{A} = \{a_1, a_2, .., a_n\}$, $\mathcal{I}_2(\mathcal{A})$ stands for $\mathcal{I}_2(a_1), .., \mathcal{I}_2(a_n)$. ∎

It is easy to see that the function ψ is computable and total, hence the whole procedure is. Now we establish the emulation results, starting from the level of non-weak one-step transitions. First we give a preliminary definition.

Definitions 4.18 $\Phi^*_{\mathcal{F}}(P)$ is the smallest set constructed by the rule:

(i) $\Phi_{\mathcal{F}}(P) \in \Phi^*_{\mathcal{F}}(P)$.

[12] We know that an important subclass of $\mathbf{C}$ where we can have the same effect by an operation using $\mathcal{I}(a)$, but the discussion on this subclass is beyond the scope of the present exposition.

(ii) $P \in \Phi^*_{\mathcal{F}}(P) \ \wedge\ a \in \mathcal{F} \quad \Rightarrow \quad P, \mathcal{I}_2(a) \ \in\ \Phi^*_{\mathcal{F}}(P)$. ∎

Now we show how the asynchronous one-step transitions in the original term are transformed into the synchronous ones, with a possible increase of an active identity receptor.

Proposition 4.19 With $P \in \Phi^*_{\mathcal{F}}(P_0)$ for any P_0 and $\mathcal{F} \supset \mathcal{FN}(P_0)$,

(i) $P_0 \xrightarrow{\downarrow av} P'_0 \ \wedge\ a \in \mathcal{F} \quad \Rightarrow \quad P \xrightarrow{\downarrow av}_s P'$ with $P' \in \Phi^*_{\mathcal{F}\cup\{v\}}(P'_0)$.

(ii) $P_0 \xrightarrow{\uparrow av} P'_0 \quad \Rightarrow \quad P \xrightarrow{\uparrow av}_s P'$ with $P' \in \Phi^*_{\mathcal{F}\cup\{v\}}(P'_0)$.

(iii) $P_0 \xrightarrow{\tau} P'_0 \quad \Rightarrow \quad P \xrightarrow{\tau}_s P'$ with $P' \in \Phi^*_{\mathcal{F}}(P'_0)$. ∎

Proof Use the normal form representation and check transitions. ∎

The second shows the reverse side, which states that all the synchronous one-step transitions in the terms after the completion are really covered (somehow) by the original asynchronous transitions.

Proposition 4.20 With $P \in \Phi^*_{\mathcal{F}}(P_0)$ and $\mathcal{F} \supset \mathcal{FN}(P_0)$,

(i) $P \xrightarrow{\downarrow av}_s P' \quad \Rightarrow \quad (P_0 \xrightarrow{\downarrow av}\xrightarrow{\tau} P'_0 \ \vee\ P_0 \xrightarrow{\downarrow av} P'_0)$ with $P' \in \Phi^*_{\mathcal{F}\cup\{v\}}(P'_0)$.

(ii) $P \xrightarrow{\uparrow av}_s P' \quad \Rightarrow \quad P_0 \xrightarrow{\uparrow av} P'_0$ with $P' \in \Phi^*_{\mathcal{F}\cup\{v\}}(P'_0)$.

(iii) $P \xrightarrow{\tau}_s P' \quad \Rightarrow \quad (P_0 \xrightarrow{\tau} P'_0 \ \vee\ P'_0 \equiv P_0)$ with $P' \in \Phi^*_{\mathcal{F}}(P'_0)$. ∎

Proof

(i) By noticing P may receive a message by P_0's original receptor or by identity receptor.

(ii) Trivial.

(iii) By noticing there can be a void τ transition between a message and an identity receptor. ∎

A small but essential corollary follows.

Corollary 4.21 *Let us have* $P \in \Phi^*_{\mathcal{F}}(P_0)$ *with* $\mathcal{F} \supset \mathcal{FN}(P_0)$. *Then*

(i) $\mathcal{FI}(P) = \mathcal{FN}(P) = \mathcal{F}$

(ii) $P \xrightarrow{l} P' \quad \Rightarrow \quad P' \in \Phi^*_{\mathcal{F}\cup\mathcal{N}(l)}(P'_0)$ *for some* P'_0 *with* $\mathcal{F} \cup \mathcal{N}(l) \supset \mathcal{FN}(P'_0)$. ∎

Proof (i) trivially holds by the form of P. (ii) is direct from Proposition 4.20 and Lemma 2.17. ∎

Based on preceding results we gain the emulation result at the level of weak transitions.

Proposition 4.22 With $P \in \Phi^*_{\mathcal{F}}(P_0)$ for $\mathcal{F} \supset \mathcal{FN}(P_0)$, we have

(i) $P \stackrel{t}{\Longrightarrow}_s P' \quad \Rightarrow \quad P_0 \stackrel{t}{\Longrightarrow} P'_0$ for some P'_0 with $P' \in \Phi^*_{\mathcal{F}\cup\mathcal{N}(t)}(P'_0)$.

(ii) $P_0 \overset{t}{\Longrightarrow} P'_0 \quad \Rightarrow \quad P \overset{t}{\Longrightarrow}_s P'$ for some P' with $P' \in \Phi^*_{\mathcal{F}\cup\mathcal{N}(t)}(P'_0)$, if t of P_0 is sound with respect to P. ∎

Proof

(i) Using induction on the length of t, we check each transition by Proposition 4.20. The repeated applications of the proposition are possible because of the invariant conditions in Corollary 4.21.

(ii) Similarly, using Proposition 4.19. ∎

Thus, relative to the parameterized set of names and neglecting inessential restriction (due to collision of names), we have clean correspondence between two weak transitions, including the closure property of derivatives. The main result in this subsection follows.

Proposition 4.23 For some finite set of names $\mathcal{F}$ let us have $\mathbf{C}_\mathcal{F} = \{P \mid \mathcal{FN}(P) \subset \mathcal{F}\}$. Suppose $P_0, Q_0 \in \mathbf{C}_\mathcal{F}$. Then the following holds.

(i) $P \in \Phi^*_\mathcal{F}(P_0) \quad \Rightarrow \quad P \approx_a P_0 \ \wedge \ P \in \mathbf{C}_\mathcal{F}$.

(ii) $P_0 \approx_a Q_0 \quad \Leftrightarrow \quad \Phi_\mathcal{F}(P_0) \approx_s \Phi_\mathcal{F}(Q_0)$.

(iii) $\Phi_\mathcal{F}(\Phi_\mathcal{F}(P_0)) \approx_s \Phi_\mathcal{F}(P_0)$. ∎

Proof

(i) By Corollary 4.16.

(ii) $(\Rightarrow)$ For some finite set of names $\mathcal{F}$, let us have:

$$\mathbf{R}_\mathcal{F} = \{(P,\ Q) \mid P \in \Phi^*_\mathcal{F}(P_0),\ Q \in \Phi^*_\mathcal{F}(Q_0) \text{ for } P_0, Q_0 \in \mathbf{C}_\mathcal{F} \text{ and } \ P_0 \approx_a Q_0.\}$$

Then we show that the union of all such relations, $\bigcup_{\mathcal{F}\subset_f \mathbf{N}} \mathbf{R}_\mathcal{F}$, where "$\subset_f$" stands for "is a finite subset of", is a synchronous bisimulation. Assuming $(P,\ Q) \in \mathbf{R}_\mathcal{F}$ for some $\mathcal{F}$, we have a following sequence of inferences.

$$\begin{array}{llll} P \overset{t}{\Longrightarrow}_s P' & \Rightarrow & P_0 \overset{t}{\Longrightarrow} P'_0 \ \text{ with } \ P' \in \Phi^*_{\mathcal{F}\cup\mathcal{N}(t)}(P'_0) & \text{(by Proposition4.22 (i))} \\ & \Rightarrow & Q_0 \overset{t}{\Longrightarrow} Q'_0 \ \text{ with } \ P'_0 \approx_a Q'_0 & \text{(by } P_0 \approx_a Q_0\text{)} \\ & \Rightarrow & Q \overset{t}{\Longrightarrow}_s Q' \ \text{ with } \ Q' \in \Phi^*_{\mathcal{F}\cup\mathcal{N}(t)}(Q'_0) & \text{(by Proposition4.22 (ii))} \end{array}$$

Note that for each of these inference steps, the soundness condition is automatically satisfied, specifically because of $\mathcal{F} \supset \mathcal{FN}(P_0) \cup \mathcal{FN}(Q_0)$. Finally, since we easily get $(P',\ Q') \in \mathbf{R}_{\mathcal{F}\cup\mathcal{N}(t)} \subset \bigcup_{\mathcal{F}\subset_f \mathbf{N}} \mathbf{R}_\mathcal{F}$, and as symmetricity is obvious, using the characterization of $\approx_a$ in Proposition 4.8, the proof is done.

$(\Leftarrow)$ Using Proposition 4.12. i.e. the fact $P \approx_s Q$ implies $P \approx_a Q$, in combination with (i) above.

(iii) By (i) and (ii) above. ∎

Let us give some remarks on the result above.

(1) The clause (ii), which can be seen as another characterization result of $\approx_a$, tells us that the exact difference in two bisimilarities, induced by two IN rules, can be captured by some collection of identity receptors.

(2) As is easily guessed, the exact image of the above result holds for asynchronous and synchronous trace equivalences and failure equivalences. This suggests the universality of the effect of the completion for weak theories. Indeed the core of the completion method lies in that it makes synchronous weak transitions proceed much like weak asynchronous transitions, while asynchronous behaviour of the concerned term remains unchanged (cf. (i) above). This means that, in the light of the discernibility of terms' behaviour, what the completion does is to let the synchronous theory forget insignificant details (i.e. the capability to consume a message and immediately process it). Note that, intuitively, when we send messages asynchronously, one may not know when they will arrive at the target, in which order, or whether they ever will or not. In this regard one may say that the obscuration is done rightly.

(3) (i) of Proposition 4.19 tells us that the completion *maximizes* the synchronous input transitions of the concerned term as far as the parameterized set of names allow, to have the synchronous input rule mimic the extravagance of the asynchronous IN rule. Also (iii) shows that the operation sends a term to some kind of the *saturated* state (in terms of the synchronous theory). These observations motivate us to understand the effect of the operation on a certain synchronous ordering. Assume given the *synchronous simulation preorder* denoted by $\sqsubseteq_s$ (which is the synchronous bisimulation minus symmetry). Then with $\mathbf{A}$ being any equivalence class of $\approx_a$ *within* some $\mathbf{C}_{\mathcal{F}}$, we have:

   - $P, Q \in \mathbf{A} \quad \Rightarrow \quad \Phi_{\mathcal{F}}(P) \approx_s \Phi_{\mathcal{F}}(Q)$.
   - $P \in \mathbf{A},\ Q \in \Phi_{\mathcal{F}}(\mathbf{A})(\subset \mathbf{A}) \quad \Rightarrow \quad P \sqsubseteq_s Q$.

   Thus the operation sends all these terms to the top element in the equivalence class ordered by $\sqsubseteq_s$. Seen conversely, it means that the top element can be characterized syntactically by the completion. This suggests that we can capture the difference between synchronous bisimilarity and asynchronous bisimilarity in a more precise way than the completion method.

In regard of (3), detailed study reveals that not only the top elements but also *all* the points in the ordering relation can be characterized in a syntactic fashion. This result has non-trivial semantic consequences, one of which is the existence of a "downward" (but not necessarily finitary) synonym of the $\mathcal{I}$ completion, that is, the *erasure* of $\mathcal{I}$'s. Unfortunately we should leave the presentation of these and other results concerning the inequational theories for the calculus, to another occasion.

5 Conclusion

Our development so far shows that the general semantic framework of concurrent computation can be constructed based on the notion of pure asynchronous interaction, with non-trivial difference from the foregoing semantic framework for synchronous interaction.

At the same time, many semantic concepts developed in terms of the synchronous interaction can find their respective positions in the new framework after suitable reconstruction. As our study has just begun, however, intense investigation is necessary for the asynchronous interaction semantics itself, as well as for the calculus itself. Before concluding, we pinpoint two further issues.

- While the results about the asynchronous interaction semantics in the present exposition, can be applied to other formalisms, such as CCS and CSP (after suitable syntactic contraction) or Boudol's γ-calculus [6], various evidences strongly suggest that the capability to create new ports and pass them should be indispensable ingredients for constructing the general computation in the setting of pure asynchrony. Computationally we already gained some results [12, 13, 14] but what is intriguing is whether we can find the significance of these properties in the semantic context. Logical characterization will play an important role here.
- There is a notable work by de Boer, Paramidessi, and his colleagues [4, 5], which tries to capture asynchronous communication using the notion of the common black board which serves as a kind of global mail delivery system, with the assumption that a system does not hold generated messages in it, but emit them immediately. In this context, notice that Proposition 3.12 shows that the pending messages are the indispensable elements in the setting of asynchronous interaction. Thus the framework by de Boer and others can be regarded as a special case where one neglects pending messages (or assumes they will get consumed immediately by the environment). Yet it is probable that the restricted theory may form an important subclass of the asynchronous interaction semantics, where reasoning etc. is more tractable than in the general case. The point deserves further investigation.

We finally note that the way to express pending messages by syntactic terms in process calculi-related formalisms, which plays a crucial role in our semantic construction, was concurrently discovered by us (early 1990), by Meseguer [16], and by Nierstrasz [25].[13] We can even trace back the idea to the representation of asynchronous messages in the actor event diagram [7] in the context of the actor model by Hewitt and others [9, 2]. But semantic significance of asynchronous communication in the general theoretical setting in contrast to synchronous communication, seems not to have been studied so much, in spite of the early work such as [3]. Our work develops the theory of asynchronous interaction based on the clean semantic and computational construction of process calculi, especially upon Milner's work on π-calculus, to acquire the general theoretical framework of concurrent computation interesting in its own right.

Acknowledgements

We would like to thank Professor Robin Milner for active and beneficial discussions in the Geneva workshop. We heartily thank Professor Peter Wegner who read the first version of the paper and gave us valuable suggestions. Our thanks also go to Makoto Kubo for many discussions, especially in terms of the new formulation of transition rules in Definition 2.15.

[13]Nierstrasz further developed the work in [26].

Vasco Vasconcelos pointed out several typographical errors and ambiguities in the final version, to which the authors are grateful.

References

[1] Abramsky, S., Observational Equivalence as a Testing Equivalence, *Theoretical Computer Science*, 53, 1987.

[2] Agha, G.: *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[3] Bergstra, J., Klop, J., and Tucker, J., Process Algebra with Asynchronous Communication Mechanisms, In *Seminar on Concurrency*, LNCS 197, 1984, Springer Verlag.

[4] de Boer, F., and Palamidessi, C., On the asynchronous nature of communication in concurrent logic languages: a fully abstract model based on sequences, In *CONCUR '90*, LNCS 458, Springer-Verlag, 1990.

[5] de Boer, F., Kok, J., Palamidessi, C., and J.J.M.M. Rutten. The Failure of failures in a paradigm for asynchronous communication. In *CONCUR '91*, LNCS, Springer-Verlag, 1991.

[6] Boudol, Towards the Lambda Calculus for Concurrent Computation. In *Proc. TAPSOFT 1989*, LNCS 351, Springer Verlag, 1989.

[7] Clinger, W. *Foundations of Actor Semantics.* AI-TR-633, MIT Artificial Intelligence Laboratory.

[8] Goguen, J., *Sheaf semantics for concurrent interacting objects.* To appear in Proc. REX School on Foundations of Object-Oriented Programming, Noorwijkerhout, The Netherlands, May 28-June1, 1990.

[9] Hewitt, C., *Viewing Control Structures as Patterns of Passing Messages.* Artificial Intelligence, 1977.

[10] Hindley,J.R., and Seldin, J.P., *Introduction to Combinators and λ-calculus.* Campbridhe university Press, 1986.

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

[12] Honda, K. and Tokoro, M., An Object Calculus for Asynchronous Communication, In: *Proc. of European Conference on Object-Oriented Programming*, LNCS, Springer-Verlag, July 1991. Extended version to appear as a Keio CS report, Novemver 1992.

[13] Honda, K., *Functions and Functionals in Interaction.* A Manuscript, November 1991.

[14] Honda, K., *A Semantically Sound Mapping Between Two Calculi.* A Manuscript, November 1991.

[15] Lampson, B.W., Paul, M., and Siegert, H.J., *Distributed Systems – Architecture and Implementation An Advanced Course.* LNCS 105, Springer-Verlag, 1981.

[16] Meseguer J., *Conditional Rewriting Logic as a Unified Model of Concurrency.* SRI-CSL-91-05, Computer Science Laboratory, SRI International, 1991. Also to appear in *Theoretical Computer Science.*

[17] Milner, R., *Calculus of Communicating Systems.* LNCS 92, Springer-Verlag, 1980.

[18] Milner, R., *Calculi for Synchrony and Asynchrony.* TCS, 1983.

[19] Milner, R., Parrow, J.G. and Walker, D.J., *A Calculus of Mobile Processes. Part I.* ECS-LFCS-89-85, Edinburgh University, 1989

[20] Milner, R., Parrow, J.G. and Walker, D.J., *A Calculus of Mobile Processes. Part II.* ECS-LFCS-89-86, Edinburgh University, 1989

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

[22] Milner, R., Functions as Processes. In *Automata, Language and Programming*, LNCS 443, 1990. The extended version under the same title as *Rapports de Recherche No.1154*, INRIA-Sophia Antipolis, February 1990.

[23] de Nicola, E., and Hennessy, M., Testing Equivalence for Processes. *Theoretical Computer Science*, 34, 1983.

[24] Nielson and Engberg, *A Calculus of Communicating Systems with Label Passing.* Research Report DAIMI PB-208, Computer Science Department, University of Aarhus, 1986.

[25] Nierstrasz, O., *A Guide to Specifying Concurrent Behaviour with Abacus.* in [29].

[26] Nierstrasz, O., Towards an Object Calculus, in this volume, 1992.

[27] Park, D., *Concurrency and Automata on Infinite Sequences.* LNCS 104, Springer-Verlag, 1980.

[28] Tokoroœ*B*!œ(B M., Toward Computing Systems in the 2000's. *Operating Systems of the 90s and Beyond*œ*B*!œ(B LNCS 563œ*B*!œ(B Springer Verlagœ*B*!œ(B 1992.

[29] Tsichritzis, D., ed. *Object Management.* Centre Universitaire D'informatique, Universite de Geneve, July 1990.

Appendix A: Consistency with the Original $\xrightarrow{l}$

The reasoning below shows that $\xrightarrow{l}$ of Definition 2.15 faithfully captures the original $\xrightarrow{l}$ in [12] up to the difference in the strength of $\equiv$, in spite of the novel way of presentation and the lack of bound outputs. Let $\equiv_{org}$ and $\xrightarrow{l}_{org}$ denote the relations defined in

Definition 3 and 8 of [12], respectively[14] We will use the extended set of labels which adds the labels of the forms $\uparrow a|v|$ to the labels in $\mathbf{L}$.

Definitions 6.1 We construct the relation $\cdot\!\xrightarrow{l}$ from $\xrightarrow{l}$ as follows.

$$\begin{array}{lcll} P \xrightarrow{\uparrow av} P' & \Rightarrow & P \cdot\!\xrightarrow{\uparrow a|v|} P' & \text{with } v \notin \mathcal{FN}(P). \\ P \xrightarrow{l} P' & \Rightarrow & P \cdot\!\xrightarrow{l} P' & \text{(otherwise)} \quad .\blacksquare \end{array}$$

As $\equiv_{org}$ is stronger than $\equiv$, we cannot immediately say $\xrightarrow{l}_{org} = \cdot\!\xrightarrow{l}$. But the following proposition shows that, if we add $\equiv_{org}$, we faithfully recover the original relation.

Proposition 6.2 $P \xrightarrow{l}_{org} P'$ if and only if $P \cdot\!\xrightarrow{l}\equiv_{org} P'$. ∎

Proof By checking that each rule of one system is inferred in the other system. The fact that if $P \xrightarrow{\uparrow a|v|}_{org} P'$ then $v \notin \mathcal{FN}(P)$ while $P \xrightarrow{\uparrow av}_{org} P'$ then $v \in \mathcal{FN}(P)$, is crucial in the case of bound output. ∎

Finally we note that the above result does not imply that bound labels are useless: on the contrary, the introduction of bound labels becomes indispensable when one should deal with transition relation at a higher level of abstraction, e.g. in treating the transition graph *per se*, or in discussing multi-step transition relation. We leave the details to another occasion.

Appendix B: Proof of Lemma 3.14

The Lemma. For any P, Q, v and x, $P \approx_a Q \;\Rightarrow\; P[v/x] \approx_a Q[v/x]$. ∎

Proof We take the relation

$$\mathcal{R} \stackrel{def}{=} \bigcup_{i \geq 0} \mathcal{R}_i$$

where

$$\begin{array}{lccc} \mathcal{R}_0 : & \approx_a & & \\ \mathcal{R}_1 : & P\,\mathcal{R}_0\,Q & \Rightarrow & P[v/x]\,\mathcal{R}_1\,Q[v/x] \\ & \vdots & & \\ \mathcal{R}_{n+1} : & P\,\mathcal{R}_n\,Q & \Rightarrow & P[v/x]\,\mathcal{R}_{n+1}\,Q[v/x]\ . \end{array}$$

and prove that this is a bisimulation. As $(P,Q) \in \mathcal{R}_1 \subset \mathcal{R}$, this shows that $(P,Q) \in \mathcal{R}_1$ implies $P \approx_a Q$ (in fact, $\mathcal{R}_1 = \approx_a$). We show this by proving that, for any $n \geq 0$, if $(P,Q) \in \mathcal{R}_n$,

$$P \xrightarrow{l} P' \quad \Rightarrow \quad \exists\, Q'\ s.t.\ (Q \stackrel{\hat{l}}{\Longrightarrow} Q')$$

with

[14] Minor typograpghical errors in Definition 8 there should be corrected; τ has to be added as a label of the transition in *COM* rule, and the side condition for *STRUCT* rule is unnecessary (though harmless).

(1) $l = \uparrow av$ or $l = \tau$ then $(P', Q') \in \mathcal{R}_n$, and
(2) $l = \downarrow av$ then $(P', Q') \in \mathcal{R}_{n+2}$.

W.l.o.g. we can omit the symmetric case.

Our proof is not based on induction on n, but uses the following construction. For $(P, Q) \in \mathcal{R}_n$, let P_0 and Q_0 denote two expressions such that

$$(P_0, Q_0) \in \mathcal{R}_0 (= \approx_a) \quad \wedge \quad P \stackrel{def}{=} P_0[v_1/x_1]..[v_n/x_n] \quad \wedge \quad Q \stackrel{def}{=} Q_0[v_1/x_1]..[v_n/x_n] \quad .$$

This pair of P_0 and Q_0 will be used when we check that each transition of one party can somulate that of the other. Note also for any n we have $\mathcal{R}_{n+1} \supset \mathcal{R}_n$.

Case $P \xrightarrow{\downarrow av} P'$. Immediately $P' \equiv (P, \leftarrow av)$. Similarly $Q \xrightarrow{\downarrow av} (Q, \leftarrow av)$. Using (iv) and (vi) of Lemma 2.6 repeatedly, we have

$$\begin{aligned} (P, \leftarrow av) &\stackrel{def}{=} (P_0[v_1/x_1]..[v_n/x_n], \leftarrow av) \\ &\stackrel{def}{=} (P_0, \leftarrow av[z_1/x_i][z_2/x_j])[v_1/x_1]..[v_n/x_n][x_i/z_1][x_j/z_2] \end{aligned}$$

where $z_1, z_2 \notin \mathcal{FN}(P) \cup \mathcal{FN}(Q) \cup \{x_1, .., x_n\}$ and that for any x_k with $1 \leq k \leq n$ and $k \neq i, j$, $x_k \notin \{a, v\}$, and similarly for Q (note we have $n+2$ substitutions outside the parenthesis above). But by Proposition 3.13 we have

$$(P_0, \leftarrow av[z_1/x_i][z_2/x_j]) \quad \approx_a \quad (Q_0, \leftarrow av[z_1/x_i][z_2/x_j])$$

which shows $((P, \leftarrow av),\ (Q, \leftarrow av))$ is in $\mathcal{R}_{n+2}$.

Case $P \xrightarrow{\uparrow av} P'$. Using the normal form representation, we have $P \equiv |\tilde{w}|(\Gamma, \leftarrow av, \Delta)$ with $a \notin \{\tilde{w}\}$. Hence $P_0 \equiv P'_0 \stackrel{def}{=} |\tilde{w}|(\Gamma', \leftarrow a'v', \Delta')$ where $a' \notin \{\tilde{w}\}$ and $|\tilde{w}|(\Gamma', \leftarrow a'v', \Delta')[v_1/x_1]..[v_n/x_n] \stackrel{def}{=} P'_0$. The rest is easy reasoning using the property of $\approx_a$ and Substitution lemma for transition (Lemma 2.18).

Case $P \xrightarrow{\tau} P'$. Again using the normal form representation, we have

$$P \equiv |\tilde{w}|(\Gamma, \leftarrow av, ay.P, \Delta) \xrightarrow{\tau} |\tilde{w}|(\Gamma, P[v/y], \Delta) \equiv P'$$

hence $P_0 \equiv |\tilde{w}|(\Gamma', \leftarrow a_1 v', a_2 y.P', \Delta')$. Now it is easy to see that a_1 and a_2 should be either both free or both bound by $|\tilde{w}|$. If both are bound, we can use the property of $\approx_a$ and (iii) of Lemma 2.18. If both are free, we have transitions

$$|\tilde{w}|(\Gamma', \leftarrow a_1 v', a_2 y.P', \Delta') \xrightarrow{\uparrow a_1 v'} \xrightarrow{\downarrow a_2 v'} \xrightarrow{\tau} |\tilde{w} - v'|(\Gamma', P[v'/y], \Delta')$$

with $(\Gamma', P[v'/y], \Delta')[v_1/x_1]..[v_n/x_n] \equiv P'$. Using the property of $\approx_a$ and repeatedly applying Lemma 2.18, we can show that there exists Q' such that $Q \stackrel{\hat{\tau}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}_n$. ∎

Appendix C: Consequences of Restricted IN Rule.

Some results are presented below concerning a transition system which restricts the IN Rule of $\xrightarrow{l}$, and bisimulation induced by it. They suggest that the full generality of IN rule is necssary for sound formulation of the asynchronous interaction semantics.

Definitions 6.3 We define a restricted asynchronous interaction, denoted $\xrightarrow{l}_r$, as the smallest relation inferred by the same rules as Definition 2.15, with $\xrightarrow{l}$ replaced by $\xrightarrow{l}_r$ except IN rule which is reformulated as

$$IN_r : \quad Q \stackrel{def}{=} |\tilde{w}|(\Gamma, \Delta) \xrightarrow{\downarrow av}_r |\tilde{w}|(\Gamma, \leftarrow av, \Delta) \quad (v \notin \{\tilde{w}\} \wedge a \in \mathcal{FI}(Q)) \quad . \blacksquare$$

The induced bisimulation is:

Definitions 6.4 $P_1 \approx_r Q_1$ if and only if $(P_1, Q_1) \in \mathcal{R}$ with

(i) For any $(P, Q) \in \mathcal{R}$, whenever $P \xrightarrow{l}_r P'$ which is sound with respect to Q, there is Q' such that $Q \stackrel{\hat{l}}{\Longrightarrow}_r Q'$ and $(P', Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ is symmetric. $\blacksquare$

What we show below is that this relation lies strictly between the synchronous bisimilarity defined in Definition 4.5 and the asynchronous bisimilarity. First, it is easy to see that:

Proposition 6.5 $P \approx_r Q \quad \Rightarrow \quad P \approx_a Q.$ $\blacksquare$

Proof To prove $P \approx_r Q \quad \Rightarrow \quad P \approx_a^\star Q$ is easy. Then we use Proposition 3.21. $\blacksquare$

But the converse is not true.

Proposition 6.6 $P \approx_a Q$ *does not imply* $P \approx_r Q.$ $\blacksquare$

Proof By the pair in Proposition 4.14. $\blacksquare$

Thus $\approx_r$ is more strict than $\approx_a$. This shows that we cannot restrict the input label at the transition level if we want to get the generality of $\approx_a$. Finally we show that $\approx_r$ is not $\approx_s$ (cf. Definition 4.5), which is known to be more strict than $\approx_a$ in a meaningful way. The reasoning to establish this negative result uses an example where the synchronous bisimulation fails not by the lack of input ports but by the *forced* input transition whose result is disastrous, and is interesting in its own right.

Proposition 6.7 $P \approx_r Q$ does not imply $P \approx_s Q$. $\blacksquare$

Proof By a pair $ax.(\leftarrow ax,\ \mathcal{I}(a),\ \mathcal{I}(b))$ and $ax.(\leftarrow ax,\ \mathcal{I}(b)),\ \mathcal{I}(a)$. ∎

In the pair which constitutes the counterexample, whenever one party has an asynchronous transition, the other has a simulating transition which satisfies the condition in Definition 6.4, hence they are in $\approx_r$. However, the synchronous transition of the second expression:

$$ax.(\leftarrow ax,\ \mathcal{I}(b)),\ \mathcal{I}(a) \xrightarrow{\downarrow av}_s ax.(\leftarrow ax,\ \mathcal{I}(b)),\ \mathcal{I}(a),\ \leftarrow av$$

cannot be simulated *synchronously* by the first one. Note that the corresponding transitions in the asynchronous setting,

$$ax.(\leftarrow ax,\ \mathcal{I}(b)),\ \mathcal{I}(a) \xrightarrow{\downarrow av}\xrightarrow{\tau} ax.(\leftarrow ax,\ \mathcal{I}(b)),\ \mathcal{I}(a),\ \leftarrow av$$

can be easily simulated trivially by

$$ax.(\leftarrow ax,\ \mathcal{I}(a),\ \mathcal{I}(b)) \xrightarrow{\downarrow av} ax.(\leftarrow ax,\ \mathcal{I}(a),\ \mathcal{I}(b)),\ \leftarrow av$$

satisfying the condition of Definition 6.4.