非同期通信意味論について

本田耕平 所 真理雄

慶応義塾大学 計算機科学科
横浜市港北区日吉3-14-1

本稿では非同期通信にもとづく並行計算の形式系に対する二つの等価性の基本的な性質と、それらの間の関係について述べる。本稿の結果は、今後のわれわれの形式系に関する様々な研究の基礎の一つとなるべきものである。

On Asynchronous Communication Semantics

Kohei Honda, Mario Tokoro

Department of Computer Science,

Keio University

3-14-1, Hiyoshi, Kohoku-ku,
Yokohama, 223, Japen

This paper presents some theoretical results concerning asynchronous communication semantics for a formal system based on a fragment of Milner's π-calculus. It compares two equational theories for the formal system, and proves several basic results concerning their relationship.

On Asynchronous Communication Semantics (An Abbreviated Version)

Kohei Honda and Mario Tokoro*

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

October 1, 1991

1 Introduction

This paper presents some theoretical results concerning asynchronous communication semantics for a formal system based on a fragment of Milner's π-calculus [15]. The formal system, which first appeared in its present form in [9], 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 the synchronous communication framework quite concisely in terms of fewer constructs (see [9]). This and other interesting findings prompted our investigation to construct a theory of concurrent computation purely based on asynchronous message passing.

This paper is an abbreviated version of [10] and presents some basic facts about equational theories for our asynchronous formal system. To be more concrete, it compares two naturally-arising equational theories for our formal system, one asynchronous and the other synchronous, and shows relationship between these two. Section 2 gives basic definitions. Section 3 is the main results of this exposition, focusing on the equational theory called *asynchronous bisimulation.* After showing that this bisimulation is a congruence relation for our system, we study the characteristics of our equational theory in comparison with its synchronous counterpart (which is a loyal recapitulation of usual synchronous bisimulation in process calculi) and prove that the asynchronous bisimilarity is strictly more inclusive than the synchronous bisimilarity. Section 4 gives some comparison. Finally Section 5 concludes the paper. All the proofs are omitted: interested readers may refer to [10].

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

2 Basic Definitions

Syntax and Structural Equivalence

The following is the syntax for π_A-calculus. It is a subset of a fragmentary π-calculus as presented in [15], expressing the notion of asynchronous communication and retaining the same expressive power in spite of its reduction of a syntactic construct. Let $\mathcal{N}$ be the infinite set of port names (or simply *names*), on which a, b, c, x, y, etc. range over. $\tilde{a}$ etc. denotes sequences of names. P, Q etc. denote terms (processes) in our calculus. The set of terms is denoted as $\mathbf{C}$.

Definition 2.1 *Syntax.*

$$\begin{array}{rcll}
P & = & \leftarrow av & \text{(a message)} \\
 & \mid & ax.P & \text{(a receptor)} \\
 & \mid & \{X(\tilde{x}) = ay.P\}(\tilde{w}) & \text{(a recursively defined receptor)} \\
 & \mid & |a|P & \text{(scope restriction)} \\
 & \mid & P, Q & \text{(concurrent composition)} \\
 & \mid & \Lambda &
\end{array}$$

Next we have structural equivalence, an extended set of α-convertibility relation. Indeed we can regard the syntactic objects as expressing some abstract entities which are those objects *modulo* these equivalence. We also owe this construction mostly to [15], though the below is much weaker as an equational theory.

Definition 2.2 Structural Equivalence.
$\equiv$ is the smallest equivalence relation induced from the following rules.

(i) $P \equiv Q$ (where P is α-convertible to Q)
(ii) $(P, Q), R \equiv P, (Q, R)$
(iii) $|x|P, Q \equiv |x|(P, Q) \quad (x \notin \mathcal{FN}(Q))$
(iv) $P, Q \equiv Q, P$
(v) $\{X(\tilde{x}) = yz.P\}(\tilde{a}) \equiv$
$yz.P[\tilde{a}/\tilde{x}][\{X(\tilde{x}) = yz.P\}/X]$.
(vi) Suppose $P \equiv Q$ then $P, R \equiv Q, R$. Then $|x|P \equiv |x|Q$。 ■

Here we have an important fact concerning *normal form representation.* Let us call the receptors and messages of the form $ax.P$ and $\leftarrow bv$ *primitive terms.* Also $|\tilde{w}|$ denotes basically a sequence of scope restricion, but as easily seen and without proof this can be safely regarded as restriction by a *set* of names.

Proposition 2.3 For any $P \in \mathbf{C}$, we have a normal form representation such that, for some $m \geq 0$,

$$P \;\equiv\; |\tilde{w}|(t_1, t_2, ..., t_m)$$

where t_i denotes primitive terms. ■

Also we have;

Lemma 2.4 For any x and v,

$$P \equiv Q \implies P[v/x] \equiv Q[v/x] \ \blacksquare$$

Asynchronous Transition System

Now let us see the asynchronous transition system for our calculus. As labels we have τ, $\downarrow \alpha$, *and* $\uparrow \alpha$ where α is either in the form $a|v|$ or av. The way of representing the transition system is simple because of the use of normal form representation. We will let Γ etc. denote sequences of primitive terms. Also we use below a function $\sigma_V(v)$ where V is a set of names and v is a name, as denoting v itself if v is not in V and else $|v|$. $\{\tilde{w}\}$ denotes a set of names consisting of elements in $\tilde{w}$. As said above, $|\tilde{w}|$ can be regarded as restriction by a set of names, so that $|\tilde{w} - v|$ denotes restriction by a set of names $\{\tilde{w}\} - \{v\}$.

Definition 2.5 *Interaction of terms*, denoted by $\xrightarrow{l}$, is the smallest relation inferred by:

$$\begin{array}{lll}
\mathrm{OUT}: & |\tilde{w}|(\Gamma, \leftarrow av, \Delta) \xrightarrow{\uparrow a\sigma_{\tilde{w}}(v)} |\tilde{w} - v|(\Gamma, \Delta) & (a \notin \{\tilde{w}\}) \\
\mathrm{IN}: & |\tilde{w}|(\Gamma, \Delta) \xrightarrow{\downarrow a\sigma_{FN(\Gamma,\Delta)}(v)} |\tilde{w}|(\Gamma, \leftarrow av, \Delta) & (v, a \notin \{\tilde{w}\}) \\
\mathrm{COM}: & |\tilde{w}|(\Gamma, \leftarrow av, ax.P, \Delta) \xrightarrow{\tau} |\tilde{w}|(\Gamma, P[v/x], \Delta) & \\
\mathrm{STRUCT}: & \dfrac{P_1' \equiv P_1,\ P_1 \xrightarrow{l} P_2,\ P_2 \equiv P_2'}{P_1' \xrightarrow{l} P_2'} & \blacksquare
\end{array}$$

The Synchronous Counterpart

Intuitively, our transition system defines behaviour of a configuration in response to its interaction with the outside as asynchronous exchange of messages between them. 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. To see its uniqueness, it is instructive to formulate another IN rule, which is more to the spirit of the foregoing synchronous experiments framework and indeed almost the same one with that in [15]. [1]

Definition 2.6 *Synchronous interaction of terms*, denoted by $\xrightarrow{l}_s$, is the smallest relation inferred by the same rules as Definition 2.5, with $\xrightarrow{l}$ replaced by $\xrightarrow{l}_s$ except IN rule which is reformulated as

$$\mathrm{IN}: \ |\tilde{w}|(\Gamma, ax.P, \Delta) \xrightarrow{\downarrow a\sigma_{FN(\Gamma,ax.P,\Delta)}(v)} |\tilde{w}|(\Gamma, P[x/v], \Delta) \ \ (v, a \notin \{\tilde{w}\})$$

[1]There is another subtle difference from the original one in the treatment of bound names. See [9] for a discussion.

Thus the synchronous "input" experiments can only proceed when some receptors are ready to receive them. As an elementary comparison between $\longrightarrow_s$ and $\longrightarrow$, it is easy to see the following.

Proposition 2.7 We let $\xrightarrow{l}$ represent all (P, Q) such that (P, l, Q) is in the transition relation.

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

These two transition systems are fundamental machinery for our semantic investigation in the next section, based on which our semantic theories for term expressions are constructed.

3 The Theory of Asynchronous Bisimulation

The Asynchronous Bisimulation

In the synchronous communication setting of process calculi, several equivalence notions have been proposed, i.e. bisimulations [14, 19], testing equivalences (or failure equivalence) [8, 16], and trace equivalences (for comprehensive treatment, see [1]). All of them can be defined over arbitrary labeled transition systems and therefore is applicable to our system. Our construction here is based on bisimulation because of its technical tractability. The similar result, however, might be obtained for other equivalence theories.

Definition 3.1 *Asynchronous bisimulation.* Let us define $\overset{\hat{l}}{\Longrightarrow}$ as $\xrightarrow{\tau}^* \xrightarrow{l} \xrightarrow{\tau}^*$ if $l \neq \tau$ and if else as $\xrightarrow{\tau}^*$. Then P_1 and Q_1 are asynchronously bisimilar, denoted by $P_1 \approx_a Q_1$ if and only if $(P_1, Q_1) \in \mathcal{R}$ where for any $(P, Q) \in \mathcal{R}$ we have

(i) Whenever $P \xrightarrow{l} P'$, for some Q', $Q \overset{\hat{l}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.

(ii) $\mathcal{R}$ is symmetric. ■

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

Substitutability of Term Expressions

Our bisimulation is quite consistent with the underlying formal system in that $\approx_a$ enjoys substitutability, i.e. it is a congruence relation. First, Definition 3.1 almost directly tells us that:

Proposition 3.2 $\approx_a$ is an equivalence relation. That is, it is symmetric, reflexive and transitive. ■

The substitutability property for concurrent composition and scope restriction can be easily obtained.

Proposition 3.3 For any P, Q, R and x,

(i) $P \approx_a Q \implies |x|P \approx_a |x|Q$.

(ii) $P \approx_a Q \implies (P, R) \approx_a (Q, R)$ ■

What is more subtle is substitutability of expressions for a body of a receptor. The following lemma is essential for proving it.

Lemma 3.4 For any P, Q, v and x, $P \approx_a Q \implies P[v/x] \approx_a Q[v/x]$ ■

Now we can show that asynchronous bisimulation is indeed preserved by prefix operation in constructing a receptor, making $\approx_a$ a congruence relation (recursive expressions can be handled using structural rules so that it is reduced to the case of prefix operation).

Proposition 3.5 $\approx_a$ *is a congruence relation.* ■

Thus two asynchronously bisimilar agents 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 effect their environment identically). In this respect a version of testing equivalence is known to be more general equivalence theory while being compatible in the bahavioural sense. For the present purposes, however, the bisimulation-based theory is more than enough.

Synchronous Bisimulation

Let us now have another equivalence theory for our system, this time based on $\xrightarrow{l}_s$. The theory has a special position in our semantic construction, because it is a loyal transportation of semantic theories for process calculi and because we can have the corresponding theory in π-calculus, which is *not* possible with the asynchronous one[2] The new equivalence is called *synchronous bisimulation* and is defined as follows.

Definition 3.6 *Synchronous bisimulation.* Let us define $\overset{\hat{l}}{\Longrightarrow}_s$ as $\xrightarrow{\tau}_s^* \xrightarrow{l}_s \xrightarrow{\tau}_s^*$ if $l \neq \tau$ and if else as $\xrightarrow{\tau}_s^*$. Then P_1 and Q_1 are synchronously bisimilar, denoted by $P_1 \approx_s Q_1$ if and only if $(P_1, Q_1) \in \mathcal{R}$ where for any $(P, Q) \in \mathcal{R}$ we have

[2]While we do not present in this paper, it seems impossible to incorporate consistently asynchronous communication semantics into synchronous communication-based process calculi, e.g. π-calculus.

(i) Whenever $P \xrightarrow{l}_s P'$, for some Q', $Q \overset{\hat{l}}{\Longrightarrow}_s Q'$ and $(P',Q') \in \mathcal{R}$.
(ii) $\mathcal{R}$ is symmetric. ■

The following holds for $\approx_s$ in our formal system. We do not know yet whether the same is also true for the fragment of π-calculus presented in [15]. For the full calculus, we have a counterexample in [13] [3].

Proposition 3.7 $\approx_s$ is a congruence relation. ■

The Inclusion Result

The $\approx_s$ is, though restricted to our syntactic construction, recapitulation of classic bahavioural equivalence theory for processes. What is intriguing is to know how much our asynchronous version differs from the synchronous one and in what way. Is the asynchronous bisimulation, after all, meaningful equivalence relation? What difference has the extraordinary **IN** rule conveyed to it? 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 our preceding propositions.

Proposition 3.8 For any P and Q, we have $P \approx_s Q \implies P \approx_a Q$ ■

Thus we have shown that $\approx_a$ equates no less terms than $\approx_s$ does. The next question is exactly how much it equates in addition to $\approx_s$. Does it equates the same class of terms as $\approx_s$? Or does it equate strictly more than the synchronous theory does? If so, what class of terms does it equate more? The next two subsections try to give answers to these questions.

The Strict Inclusion Result

Now that we know $\approx_a \supset \approx_s$, the next task is to 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.

Definition 3.9 *$\mathcal{I}(x)$ which does nothing.* We define a family of *identity receptors* as

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

for an arbitrary port name a. ■

[3]The counterexample is: $x\bar{y} + \bar{y}x \sim x|\bar{y}$ but $(x\bar{y} + \bar{y}x)[y/x] \not\sim (x|\bar{y})[y/x]$. It seems that the lack of subsequent behaviour in our formal system may be more important than the lack of summation, but details are to be seen.

Immediately we have

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

and also, for any $n \sqsupseteq 1$,

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

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 3.10 $\approx_s \subset \approx_a$ but $\approx_a \not\subset \approx_s$.

We will give the outline of the proof. The first half has been already verified. 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 $\Lambda \not\xRightarrow{\downarrow av}_s$ (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 zero or more messages without bound names. Then it is easy to show that this is a bisimulation as follows.

(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 $\mathcal{I}(a), P \xrightarrow{\tau} Q$. then the only possibility is there is some P' such that $P \equiv P', \leftarrow av$. Then

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

As obviously $P \xRightarrow{\epsilon} P$, this case holds.

(3) The symmetric case can be proved trivially.

To know $\mathcal{I}(a) \approx_a \Lambda$, take $P \stackrel{\text{def}}{=} \Lambda$.

One remark is due here. Proposition 3.10 shows that the theory of $\approx_a$ thinks that 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, $\mathcal{I}$ *may never give any influence to its environment*, situated anywhere[4]. Thus we find out:

(1) The computational meaning of $\mathcal{I}(a)$ is better captured in the theory of $\approx_a$.

(2) While $\mathcal{I}(a)$ is a meaningless grain of computation, as a grain of *semantics* it plays an important role between two semantic theories.

[4]When fairness is to be considered, the statement should be reconsidered.

4 A Note on Other Asynchronous Communication-based Formalisms

We finally note that our concept of "expression of pending messages by syntactic terms" as found in the present exposition are almost concurrently discovered in the context of process calculi-related formalisms by us (early 1990), by Meseguer [11], and by Nierstrasz [18]. It is possible that the similar idea may have existed for some years. We can even trace back the idea to the representation of asynchronous messages in the actor event diagram [5]. But semantic and computational significance of asynchronous communication in the theoretical setting in contrast to synchronous communication seems not to have been studied so much, in spite of the early work such as [3] and the conceptual emphasis on it in the context of the study of the actor model by Hewitt and his colleagues [7]. Please see, however, a recent notable work on asynchronous communication in the context of so-called concurrent logic programming languages by de Boer and Palamidessi [4], which tries to capture asynchronous communication using the notion of the common black board which serves as a kind of global mail delivery system. The important difference is the treatment of pending messages in their framework where they do not constitute a part of systems' states.

5 Conclusion

The study of the theoretical content of asynchronous communication has just started. We hope that our investigation will lead us to clarification of the meaning of synchrony and asynchrony in communication in a formal setting, thus providing a firm foundation for theoretical and pragmatic development in the area of concurrent and distributed computing. Lastly we would like to thank Professor Robin Milner for discussions, Professor Peter Wegner for his kind suggestions, to Makoto Kubo for his assistance in some of the proofs and for long discussions.

References

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

[2] Barendreght, H. *The Lambda Calculus: Its Syntax and Semantics.* North Holland, 1984.

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

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

[6] 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.

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

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

[9] 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.

[10] Honda, K. and Tokoro, M., On Asynchronous Communication Semantics. To appear in *Proc. of ECOOP '91 Workshop on Object-based Concurrent Computing*, LNCS, Springer-Verlag, February 1992.

[11] 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.*

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

[13] Milner, R., Parrow, J.G. and Walker, D.J., *A Calculus of Mobile Processes. Part I and II.* ECS-LFCS-89-85/86, Edinburgh University, 1989

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

[15] 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.

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

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

[18] Nierstrasz, O., *A Guide to Specifying Concurrent Behaviour with Abacus.* in [21].

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

[20] Tokoro, M., Computational Field Model: Toward a New Computing Model/Methodology for Open Distributed Environment. In *Proc. of The 2nd IEEE Workshop on Future Trends in Distributed Computing Systems*, Cairo, 1990.

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

[22] Yonezawa, A., and Tokoro, M., ed., *Object-Oriented Concurrent Programming.* MIT Press, 1986.