Theory by Process

Yuxi Fu

BASICS, Department of Computer Science
Shanghai Jiaotong University, Shanghai 200240
MOE-MS Key Laboratory for Intelligent Computing and Intelligent Systems

Abstract. Theories defined in a process model are formalized and studied. A theory in a process calculus is a set of perpetually available processes with finite interactability, each can be regarded as a service, an agent behind the scene or an axiom. The operational and observational semantics of the theories are investigated. The power of the approach is demonstrated by interpreting the asynchronous π-calculus as a theory, the asynchronous theory, in the π-calculus. A complete axiomatic system is constructed for the asynchronous theory, which gives rise to a proof system for the weak asynchronous bisimilarity of the asynchronous π.

1 Introduction

In a network computing environment, how do we evaluate the capacities of two service providers S_1 and S_2? Suppose we place a request to both the providers. One, say S_1, finds the appropriate service in its repertoire and immediately delivers the service. The other provider S_2 does not specialize in the kind of service we are interested in. But since it has a strong search ability, it simply redirects our request to a third party who can offer the service we want for free. Moreover S_2 does it in such a manner that we are not aware of the existence of the third party at all. If S_1 and S_2 can always supply the services with similar qualities, we are led to believe that they are equally powerful. The point is that in a distributed computing environment, we are not testing S_1 and S_2 in isolation. That is clearly impossible. No one can stop S_1 and S_2 from exploring the resources freely available on the network. Although we can pretend that we are testing S_1, what we are really doing is to test S_1 plus all the resources accessible by S_1. Worse still S_1 is often blurred with the environment so that we are not always able to tell precisely which is which. The service providers S_1 and S_2 could have quite different capacities when isolated. They can be however equivalent in a resource rich network environment.

We would like to formalize the above scenario in interaction models. Our basic idea is to regard the services available on the network as a set $\mathcal{S} = \{S_1, S_2, \ldots\}$ of processes. We shall always assume that each member of the set is well-founded. It's no good to have a service that never delivers anything. A one-shot service is not of much use as a piece of resource. So the actual services distributed over different locations are perceived as the processes $!S_1, !S_2, \ldots$. The finite behavior of a process P sitting among $!S_1, !S_2, \ldots$ can always be inspected by considering the finite action sequences of the shape

$$P \mid S_{i_1} \mid \ldots \mid S_{i_k} \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_n} P'$$

since in a finite number of steps, the process P may only consult a finite number of service providers. If we consider a single step action of P, we only have to focus on the actions of the form $P \mid S' \xrightarrow{\lambda} P'$ for $S' \in \mathcal{S}$. Now the crucial point is that the services really stay invisibly in the background. So an action $P \mid S' \xrightarrow{\lambda} P'$ would appear to us as the action $P \xrightarrow{\lambda} P'$. This is the starting point for the semantics of a theory.

From a model theoretical point of view, sometimes we need to work with an open system within a closed world. The idea is best explained by the notion of asynchrony. The asynchronous π-calculus [Bou92,HT91a,HT91b,HY95,ACS98] is obtained from the synchronous π-calculus by detaching the output prefixes from any continuations. At first look it appears a bit simplistic since asynchrony can not be a syntactical issue. In reality asynchrony is implemented by extra mechanism which we choose to ignore. In a closed model like π-calculus, the detachment between the prefix and the continuation is the only sensible thing to do to achieve asynchrony without introducing any additional gadgets.

There ought to be an alternative treatment to the asynchrony in π-calculus that does not resort to any syntactical manipulation. An output process $\overline{a}c.P$ may communicate to a background environment. The latter picks up the name c and sends it to an input process in a later time. The background environment consists of a bunch of processes of the form $!a(x).\overline{a}x$. These processes are supposed to be hidden from the users. They form a theory in the above sense.

One may also study (constructive) logics in process models. Abramsky's work [Abr93] and a number of related works have shown how to model a logic in a process calculus. But how about a logic theory, say the Peano theory, defined in a logic? From our perspective, a particular logical theory can be formulated as a theory defined in a process model. An element of the theory codes up an axiom of the logic theory. In the process approach to logic, a proposition is interpreted as a process of the form $!A$. To make use of the axiom, an environment has to interact with a copy of A. When the verification (proof) is complete, the environment has gathered enough evidence for the validity of the proposition.

In a similar fashion to the interpretations of logics, programs in different styles have been translated into the π-model [Wal95,HO95,Mil92,CF10]. What is lacking in this research field is any idea about implementations of programming languages. Again the notion of theory is in sight. The implementation of a programming language amounts to defining a theory that codes up the system functions or routines that come with the definition of the language.

The motivation for this paper is that the notion of theory defined by processes arises naturally in many applications of process models. It is worthwhile to give an application independent study of the problems pertaining to such theories. Formal studies in this area are likely to shed new light on some familiar topics.

2 Pi Calculus

The π-calculus of Milner, Parrow and Walker [MPW92] has been widely studied in both theory and in practice. Our presentation of the model is slightly different from the standard one. The main difference is that we draw a firm line between the names and the name variables. The reader is advised to consult [FZ10] for the discussion why the distinction between the two categories is important. Throughout the paper the following notational convention will be observed:

- The set $\mathcal{N}$ of the *names* is ranged over by a, b, c, d, e, f, g, h.
- The set $\mathcal{N}_v$ of the *name variables* is ranged over by u, v, w, x, y, z.
- The set $\mathcal{N} \cup \mathcal{N}_v$ is ranged over by l, m, n, o, p, q.

An assignment ρ is a partial function from $\mathcal{N}_v$ to $\mathcal{N}$ whose domain of definition is cofinite. A condition is a finite conjunction of atomic propositions. An atomic proposition is either a match $[m{=}n]$ or a mismatch $[m{\neq}n]$. We always omit the conjunction operator. The set of the conditions is ranged over by ϕ, φ, ψ. We write $\varphi \Leftrightarrow \top$ ($\varphi \Leftrightarrow \bot$) if φ is evaluated to the true value $\top$ (the false value $\bot$) no matter how the name variables appearing in the condition are instantiated. Similarly we can define $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$.

Our definition of the π-calculus is influenced by the results obtained in [FL10]. It is equivalent to the standard presentation in terms of expressiveness and it has better algebraic property. The set of *terms* is inductively constructed from the following grammar:

$$T := \sum_{i \in I} \varphi_i \lambda_i.T_i \mid T \,|\, T' \mid (c)T \mid !\pi.T,$$

where $\lambda_i \in \{n(x), \overline{n}m, \overline{n}(c)\} \cup \{\tau\}$ and $\pi \in \{n(x), \overline{n}m, \overline{n}(c)\}$; they are prefixes. Here τ indicates an interaction, $ab, \overline{a}b, \overline{a}(c)$ denote respectively an input action, an output action and a bound output action. In the guarded choice $\sum_{i \in I} \varphi_i \lambda_i.T_i$ the indexing set I must be finite. We shall often write $\varphi_1\lambda_1.T_1 + \ldots + \varphi_n\lambda_n.T_n$ for $\sum_{i \in \{1,..,n\}} \varphi_i\lambda_i.T_i$. We write $\mathbf{0}$ for the guarded choice whose indexing set is the empty set. Due to the set theoretical nature the guarded choice $\varphi_1\lambda_1.T_1 + \ldots + \varphi_i\lambda_i.T_i + \varphi_{i+1}\lambda_{i+1}.T_{i+1} + \ldots + \varphi_n\lambda_n.T_n$ is the same as $\varphi_1\lambda_1.T_1 + \ldots + \varphi_{i+1}\lambda_{i+1}.T_{i+1} + \varphi_i\lambda_i.T_i + \ldots + \varphi_n\lambda_n.T_n$. Sometimes we will abuse notation by writing for instance $\varphi_0\lambda_0.T_0 + \sum_{i \in \{1,2\}} \varphi_i\lambda_i.T_i$ for $\varphi_0\lambda_0.T_0 + \varphi_1\lambda_1.T_1 + \varphi_2\lambda_2.T_2$. We will also abbreviate $[\top]\lambda.T + \sum_{i \in I} \varphi_i\lambda_i.T_i$ to $\lambda.T + \sum_{i \in I} \varphi_i\lambda_i.T_i$. A name c appearing underneath the localization operator (c) or a bound output prefix $\overline{a}(c)$ is local. A name is global if it is not local. A name variable x is bound if it is underneath an input prefix, say $n(x)$. A name variable is free if it is not bound. Both local names and bound name variables are subject to α-conversion. We will use the functions $gn(\_), ln(\_), n(\_), fv(\_), bv(\_), v(\_)$ with the obvious meanings. A *process* is a term in which all the name variables are bound. Let $\mathcal{T}$ denote the set of the terms, ranged over by R, S, T, and $\mathcal{P}$ the set of the processes, ranged over by $A, B, C, \ldots, O, P, Q$. The term $!\pi.T$ is in replication form. A term without any occurrence of the replication operator is called *finite*.

The operational semantics of this π-calculus is defined by the labeled transition system generated inductively from the following rules.

Action

$$\frac{}{\sum_{i\in I}\varphi_i\lambda_i.T_i \xrightarrow{ac} T_i\{c/x\}}\ \begin{array}{l}\varphi_i \Leftrightarrow \top,\\ \lambda_i = a(x).\end{array} \qquad \frac{}{\sum_{i\in I}\varphi_i\lambda_i.T_i \xrightarrow{\lambda_i} T_i}\ \begin{array}{l}\varphi_i \Leftrightarrow \top,\\ \lambda_i \text{ is not}\\ \text{an input.}\end{array}$$

Composition

$$\frac{T \xrightarrow{\lambda} T'}{S \,|\, T \xrightarrow{\lambda} S \,|\, T'} \qquad \frac{S \xrightarrow{ab} S' \qquad T \xrightarrow{\overline{a}b} T'}{S \,|\, T \xrightarrow{\tau} S' \,|\, T'} \qquad \frac{S \xrightarrow{ac} S' \qquad T \xrightarrow{\overline{a}(c)} T'}{S \,|\, T \xrightarrow{\tau} (c)(S' \,|\, T')}$$

Localization

$$\frac{T \xrightarrow{\overline{a}c} T'}{(c)T \xrightarrow{\overline{a}(c)} T'} \qquad \frac{T \xrightarrow{\lambda} T'}{(c)T \xrightarrow{\lambda} (c)T'}\ c \notin n(\lambda)$$

Replication

$$\frac{}{!a(x).T \xrightarrow{ab} T\{b/x\} \,|\, !a(x).T} \qquad \frac{}{!\pi.T \xrightarrow{\pi} T \,|\, !\pi.T}\ \text{if } \pi \text{ is not an input.}$$

We have omitted all the symmetric rules. There is a side condition $ln(\lambda) \cap gn(T) = \emptyset$ on the first composition rule. These remarks also apply to the labeled transition systems defined later. Let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$. Let $\overset{\widehat{\lambda}}{\Longrightarrow}$ be $\Longrightarrow$ if $\lambda = \tau$ and $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$ otherwise. These notations allow us to define Milner and Park's bisimulation equality [Mil89].

Definition 1. *A symmetric relation $\mathcal{R}$ on $\mathcal{P}$ is a* weak bisimulation *if $Q \overset{\widehat{\lambda}}{\Longrightarrow} Q'\mathcal{R}P'$ whenever $Q\mathcal{R}P \xrightarrow{\lambda} P'$. The* weak bisimilarity $\simeq$ *is the largest weak bisimulation.*

We write $S \simeq T$ if $S\rho \simeq T\rho$ for every assignment ρ whose domain of definition is disjoint from $bv(S \,|\, T)$. An example of bisimulation equality is $!\pi.T \simeq \pi.(T \,|\, !\pi.T)$. For the particular π-calculus of this paper, $\simeq$ is closed under all the operators.

Theorem 1. *The relation $\simeq$ is both an equivalence and a congruence on $\mathcal{T}$.*

Theorem 1 relies on the fact that in the guarded choice $\sum_{i\in I}\varphi_i\lambda_i.T_i$ the operator is $\sum_{i\in I}\varphi_i\lambda_i.\_$. From $\tau \simeq [x{=}y]\tau$ we may derive $\overline{a}a + \tau.\tau \simeq \overline{a}a + \tau.[x{=}y]\tau$; but we may not derive $\overline{a}a + \tau \simeq \overline{a}a + [x{=}y]\tau$.

A process P is *well-founded* if there is no infinite action sequence $P \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_i} \ldots$ starting from P. It is a process with *finite interactability* if it is well-founded and there is a number $k \neq 0$ such that no action sequence $P \xrightarrow{\lambda_1} P_1 \ldots \xrightarrow{\lambda_i} P_i$ of P contains more than k non-τ actions. A process P is *functional* if every maximal action sequence of P is of the form $P \xrightarrow{\lambda}\Longrightarrow\xrightarrow{\lambda'} P'$, where λ is an input action and λ' an output action.

3 Theory

Upon request, a service provider should deliver the service in a short time. In a similar token, a proposition should have a finite description so that its validity can be verified in a finite number of steps. These observations lead to the following definition.

Definition 2. *A* theory $\mathbf{A}$ *is a nonempty set of processes of the form* $\pi.T$ *with finite interactability. These processes are called* axioms*.*

The intuition behind a theory is that it provides a set of eternal truths within a closed world. The finite interactability condition ensures that a service must be delivered within an expected number of interactions. Operationally an axiom A can be identified to the process $!A$. Every proof in the closed world can make inquiry into these laws. Propositions valid in the model are all relative to the set of the eternal truths. By definition every axiom A is nontrivial in the sense that $A \not\simeq \mathbf{0}$. A theory $\mathbf{A}$ is *finite* if it is a finite set. It is *finitely presentable* if it can be generated from a finite theory. By $\mathbf{A}$ being generated from $\mathbf{B}$, we mean that

$$\mathbf{A} = \{B\alpha \mid B \in \mathbf{B}, \text{ and } \alpha \text{ is an injective function from } \mathcal{N} \text{ to } \mathcal{N}\}.$$

A theory is *functional* if all its axioms are functional. A theory is *recursive* if it is a recursive set of finite processes.

Let's see some examples.

Example 1. The asynchronous theory $\mathbf{Asy}$ is defined by the finitely presentable theory $\{a(x).\overline{a}x \mid a \in \mathcal{N}\}$. In the presence of $\mathbf{Asy}$ communications are asynchronous. An output process $\overline{a}c.P$ does not have to interact with the target process. It could interact with the axiom $a(x).\overline{a}x$ and let the latter pass the information to the target. Using the same idea, one may define the finite theory $\mathbf{AB} = \{a(x).\overline{b}x, b(x).\overline{a}x\}$ that essentially identifies the names a and b.

Example 2. The natural numbers can be coded up in the following fashion:

$$\begin{aligned} [\![\underline{0}]\!]_p &\stackrel{\text{def}}{=} \overline{p}\bot, \\ [\![\underline{i{+}1}]\!]_p &\stackrel{\text{def}}{=} (q)(\overline{p}q \mid [\![\underline{i}]\!]_q). \end{aligned}$$

Here p is the access name for a number. The notation $\bot$ is a name that denotes false; similarly the name $\top$ denotes true. The natural numbers are underlined to avoid any confusion. For simplification the following derived prefix is introduced:

$$\overline{a}(\underline{i}).T \stackrel{\text{def}}{=} \overline{a}(p).(T \mid [\![\underline{i}]\!]_p). \tag{1}$$

It is routine to define processes $SUCC_a$, ADD_a, MUL_a, EQ_a and L_a that implement the successor function, the addition, the multiplication, the equality predicate and the linear order predicate on the natural numbers. The process L_a for example inputs a local name at a; and then uses that local name to get three more local names, say b, c, d. It continues to input a number $\underline{i}$ at b and a number $\underline{j}$ at c. Finally it returns $\top$ at d if $i < j$; otherwise it returns $\bot$ at d. Let **Pa** be the following theory

$$\{\overline{p_0}(\underline{0}), \ldots, \overline{p_i}(\underline{i}), \ldots\} \cup \{SUCC_a, ADD_a, MUL_a, EQ_a, L_a \mid a \in \mathcal{N}\}.$$

It is an implementation of the Peano arithmetic in π-calculus. For details of the encodings the reader could consult [FZ10].

Example 3. The theory **Crp** is given by the union of **Pa** with the following set

$$\{e(v).v(y).Enc, d(v).v(z).Dec\}.$$

The action sequences of the encryption function $e(v).v(y).Enc$ are all of the form

$$e(v).v(y).Enc \xrightarrow{ec}\xrightarrow{cf}\overset{\tau}{\Longrightarrow}\xrightarrow{\overline{c}(g)}\simeq [\![\underline{i}]\!]_g$$

for some number i. After receiving a local name c, it inputs at c a name that points to a number, the plain text, and then outputs at c a name that points to the number $\underline{i}$, the encrypted text, after completing a sequence of internal computations. The decryption function $d(v).v(z).Dec$ has the dual semantics. The theory **Crp** provides an encryption/decryption facility every process can make use of. A complete specification of **Crp** depends on the choice of the encryption and decryption functions.

Example 4. Suppose that we would like to define a random number generator that provides perpetual service on network. It appears at first sight that the theory **Ran** can be defined by extending **Pa** with the process

$$a(v).(c)(\overline{c}(\underline{0}) \mid !c(x).(\overline{c}(d).\overline{d}x + \overline{v}x)). \tag{2}$$

Upon receiving a private channel provided by a user, the process (2) randomly generates a number and sends it to the user through the private channel. However process (2) may diverge. So it is not an axiom according to our definition. The theory **Ran** can be defined by the finite set $\{\overline{g}(\underline{0}), g(x).\overline{g}(p).\overline{p}x)\}$. It can also be defined by the infinite set $\{\overline{g}(\underline{0}), \overline{g}(\underline{1}), \ldots, \overline{g}(\underline{i}), \ldots\}$. It is worth remarking that the randomness is achieved by the nondeterminism. There is no other way.

Example 5. The π-calculus has been used both as a specification language and a machine language. The rational behind these practices is that π is expressive enough to qualify for a machine model. Now if we think of π-calculus as a machine model, we can talk about programming in π-calculus. This is precisely what is done in [Wal95]. Formally what is then an interpreter of a higher order programming language on π? Whatever the interpreter is, it must give an account of the standard routines and packages supplied by the programming language. In our opinion these routines and packages are best interpreted as a theory **Prg**. Two programs defined according to the grammar of the language are equivalent if they are so in the presence of **Prg**. Let O be a program that invokes a system routine and P be a user defined program that achieves the same functionality. Conceptually O and P are equivalent. But they are not bisimilar since the former may interact at a name which P does not know. The notion of theory is a starting point to address issues of this kind.

3.1 Semantics

To investigate the algebraic properties of the theories, we need to define the operational semantics of the theories first. The power of a theory $\mathbf{A}$ is duly exhibited by the 'process'

$$\prod_{A\in\mathbf{A}} !A,$$

which is not always admissible at the syntactical level since it makes use of a possibly infinite composition. A process P under theory $\mathbf{A}$ can be imagined as a fixpoint in the following sense:

Operationally P is the same as $P \,|\, \prod_{A\in\mathbf{A}} !A$.

But notice that $P \,|\, A$, for each $A \in \mathbf{A}$, is also a fixpoint of the same nature. By exploring the fact that $\prod_{A\in\mathbf{A}} !A$ is equivalent to $A \,|\, \prod_{A\in\mathbf{A}} !A$, one sees that $P \,|\, A$ is operationally the same as P. The semantics of the theory $\mathbf{A}$, or the semantics of the $\pi_{\mathbf{A}}$-calculus, extends the operational semantics of the π-calculus with the following rule:

$$\frac{P \,|\, A \xrightarrow{\lambda} P'}{P \xrightarrow{\lambda} P'} \; A \in \mathbf{A}. \tag{3}$$

Definition 1 can be immediately applied to the $\pi_{\mathbf{A}}$-calculus.

Definition 3. *A symmetric relation $\mathcal{R}$ on $\mathcal{P}$ is a* weak $\mathbf{A}$-bisimulation *if $Q \overset{\widehat{\lambda}}{\Longrightarrow} Q'\mathcal{R}P'$ in $\pi_{\mathbf{A}}$ whenever $Q\mathcal{R}P \xrightarrow{\lambda} P'$ in $\pi_{\mathbf{A}}$. The* weak $\mathbf{A}$-bisimilarity $\simeq_{\mathbf{A}}$ *is the largest weak $\mathbf{A}$-bisimulation.*

The proof of Theorem 1 can be repeated to show that $\simeq_{\mathbf{A}}$ is both an equivalence and a congruence on $\mathcal{T}$.

Since every axiom in a theory is nontrivial, the fact stated in the next proposition is apparent.

Proposition 1. *The strict inclusion $\simeq \subset \simeq_{\mathbf{A}}$ holds for every theory $\mathbf{A}$.*

Proof. Using the fact that $\simeq$ is closed under composition, it is easy to show that $\simeq$ is an $\mathbf{A}$-bisimulation. The inclusion is strict since $\mathbf{A}$ is nonempty. □

The next lemma is a generalization of Proposition 1.

Lemma 1. *If $\mathbf{A} \subseteq \mathbf{B}$ then $\simeq_{\mathbf{A}} \subseteq \simeq_{\mathbf{B}}$.*

By abusing the notation again, one could describe the relationship between $\mathbf{0}$ and the theory $\mathbf{A}$ by the following statement:

Operationally $\mathbf{0}$ is the same as $\prod_{A\in\mathbf{A}} !A$.

The equivalence has been exploited to define the semantics of the asynchronous π-calculus. Honda and Tokoro introduce the following rule in [HT91a,HT91b].

$$\frac{}{\mathbf{0} \xrightarrow{ac} \overline{a}c} \tag{4}$$

It is evident that (4) is essentially (3) applied to $\mathbf{Asy}$.

3.2 Kernel

A theory $\mathbf{A}$ is *consistent* if $\simeq_{\mathbf{A}}$ is not $\mathcal{P}\times\mathcal{P}$; it is inconsistent otherwise. The next proposition is useful.

Proposition 2. *The following statements are equivalent:*
(i) $\mathbf{A}$ *is consistent;*
(ii) $\exists P{\in}\mathcal{P}.P \not\simeq_{\mathbf{A}} \mathbf{0}$*;*
(iii) $\exists P{\in}\mathcal{P}.\forall A{\in}\mathbf{A}.P \not\simeq_{\mathbf{A}} A$*.*

Proof. If (ii) did not hold, then every process would be equated by theory $\mathbf{A}$, contradicting (i). Hence (i) implies (ii). If $\forall P{\in}\mathcal{P}.\exists A{\in}\mathbf{A}.P \simeq_{\mathbf{A}} A$, then every process is equated to $\mathbf{0}$. So (ii) implies (iii). Finally (iii) trivially implies (i). □

The above proposition indicates that there is a distinguishing line between the processes equal to $\mathbf{0}$ in the equational theory of $\mathbf{A}$ and those that are not. This motivates the following definition: The *kernel* $\mathbf{A}_{ker}$ of the theory $\mathbf{A}$ is the set of the processes equal to $\mathbf{0}$ under the theory $\mathbf{A}$, i.e.

$$\mathbf{A}_{ker} = \{A \mid A \simeq_{\mathbf{A}} \mathbf{0}\}.$$

By Proposition 2, a theory is consistent if and only if its kernel is not $\mathcal{P}$.

Proposition 3. *The* $\mathbf{A}$*-bisimilarity equals the* $\mathbf{B}$*-bisimilarity iff* $\mathbf{A}_{ker} = \mathbf{B}_{ker}$*.*

Proof. It is clear that $\mathbf{A}_{ker} \subseteq \mathbf{B}_{ker}$ if and only if $\simeq_{\mathbf{A}}\subseteq\simeq_{\mathbf{B}}$. □

A corollary of Proposition 3 is that the power of a theory is essentially determined by its kernel. One could define for instance that $\mathbf{A}$ is a subtheory of $\mathbf{B}$ if $\mathbf{A} \subseteq \mathbf{B}_{ker}$, and that A is essentially in $\mathbf{A}$ if $A \in \mathbf{A}_{ker}$.

Although it is easy to see that **Asy** is consistent, it is generally a tricky job to establish the consistency of a theory. Let $\mathbf{F}$ be the recursive theory consisting of all the finite processes. For each process P, let $P^{\neg(!)}$ denote the process obtained from P by removing all the occurrences of the replication operator and the localization operator. It is not difficult to see that $P \simeq_{\mathbf{F}} P^{\neg(!)} \simeq_{\mathbf{F}} \mathbf{0}$. So $\mathbf{F}_{ker} = \mathcal{P}$. Therefore $\mathbf{F}$ is inconsistent. For a positive result, we remark that all functional theories are consistent. In a functional theory the process $\overline{a}a$ is never equal to $\mathbf{0}$.

4 Asynchronous Theory and Asynchronous $\boldsymbol{\pi}$

We prove in this section that **Asy** provides a faithful account of the asynchronous π-calculus. We adopt the following grammar for the asynchronous π-calculus, which summarizes the essential feathers of the calculi defined in literature [Bou92,HT91a,HT91b,HY95,ACS98].

$$T := \mathbf{0} \mid \overline{n}m \mid \sum_{i\in I} n_i(x).T_i \mid T \,|\, T \mid (c)T \mid !n(x).T.$$

Notice that the above grammar maintains a distinction between the names and the name variables. There are basically two ways to formulate the semantics of the asynchronous π-calculus. Honda and Tokoro's semantics makes use of the rule (4). The notion of theory is lurking in their framework. Amadio, Castellani and Sangiorgi's approach takes a more traditional view on the asynchronous π. Their operational semantics is defined by the following rules.

Action

$$\dfrac{}{\overline{a}b \xrightarrow{\overline{a}b} \mathbf{0}} \qquad \dfrac{}{\sum_{i\in I} a_i(x).T_i \xrightarrow{a_i c} T_i\{c/x\}}$$

Composition

$$\dfrac{T \xrightarrow{\lambda} T'}{S\,|\,T \xrightarrow{\lambda} S\,|\,T'} \qquad \dfrac{S \xrightarrow{ab} S' \qquad T \xrightarrow{\overline{a}b} T'}{S\,|\,T \xrightarrow{\tau} S'\,|\,T'} \qquad \dfrac{S \xrightarrow{ac} S' \qquad T \xrightarrow{\overline{a}(c)} T'}{S\,|\,T \xrightarrow{\tau} (c)(S'\,|\,T')}$$

Localization

$$\dfrac{T \xrightarrow{\overline{a}c} T'}{(c)T \xrightarrow{\overline{a}(c)} T'} \qquad \dfrac{T \xrightarrow{\lambda} T'}{(c)T \xrightarrow{\lambda} (c)T'}\ c \notin n(\lambda)$$

Replication

$$\dfrac{}{!a(x).T \xrightarrow{ac} T\{c/x\}\,|\,!a(x).T}$$

In Honda and Tokoro's treatment the asynchronous π differs from the synchronous π at the operational level, whereas in Amadio, Castellani and Sangiorgi's approach it is at the observational level. The definition of the asynchronous bisimilarity [ACS98] appears odd from the point of view of interaction.

Definition 4. *A symmetric relation $\mathcal{R}$ on the asynchronous π-processes is an* asynchronous bisimulation *if the following statements are valid whenever $P\mathcal{R}Q$:*

1. *If $Q \xrightarrow{\tau} Q'$ then $P \Longrightarrow P'\mathcal{R}Q'$ for some P'.*
2. *If $Q \xrightarrow{\overline{a}b} Q'$ then $P \overset{\overline{a}b}{\Longrightarrow} P'\mathcal{R}Q'$ for some P'.*
3. *If $Q \xrightarrow{\overline{a}(b)} Q'$ then $P \overset{\overline{a}(b)}{\Longrightarrow} P'\mathcal{R}Q'$ for some P'.*
4. *If $Q \xrightarrow{ab} Q'$ then either $P \overset{ab}{\Longrightarrow} P'\mathcal{R}Q'$ for some P' or $P \Longrightarrow P'$ for some P' such that $P'\,|\,\overline{a}b\ \mathcal{R}\ Q'$.*

The asynchronous bisimilarity $\simeq_a$ is the largest asynchronous bisimulation.

The asynchronous π is a syntactic subcalculus of π. It is also an operational variant of π according to Honda and Tokoro's formulation. Amadio, Castellani and Sangiorgi have proved that $\simeq_a$ coincides with Honda and Tokoro's bisimulation equivalence, called HT-bisimilarity in [ACS98]. Their proof can be extended to produce a proof of the following theorem.

Theorem 2. *Let S, T be asynchronous π-terms. Then $S \simeq_a T$ iff $S \simeq_{\mathbf{Asy}} T$.*

Theorem 2 can be interpreted as saying that the asynchronous π is a syntactical simplification of $\pi_{\mathbf{Asy}}$. It perceives $\pi_{\mathbf{Asy}}$ as a submodel of the π-calculus. It can also be seen as a justification of the asynchronous π as defined by Honda and Tokoro, as well as the variant defined by Amadio, Castellani and Sangiorgi.

L1	$(c)\mathbf{0}$	$=$	$\mathbf{0}$	
L2	$(c)(d)T$	$=$	$(d)(c)T$	
L3	$(c)([x{=}c]\varphi\lambda.T + \sum)$	$=$	$(c)\sum$	
L4	$(c)([x{\neq}c]\varphi\lambda.T + \sum)$	$=$	$(c)(\varphi\lambda.T + \sum)$	
L5	$(c)(\varphi\lambda.T + \sum)$	$=$	$(c)\sum$	if $\exists d \in \mathcal{N}.\lambda = \overline{c}d \vee \lambda = \overline{c}(d)$
L6	$(c)(\varphi\overline{n}c.T + \sum)$	$=$	$(c)(\varphi\overline{n}(c).T + \sum)$	if $c \notin gn(\varphi) \wedge c \neq n$
L7	$(c)\sum_{i\in I}\varphi_i\lambda_i.T_i$	$=$	$\sum_{i\in I}\varphi_i\lambda_i.(c)T_i$	if $\forall i \in I.c \notin gn(\varphi_i, \lambda_i)$
M1	$[\bot]\lambda.T + \sum$	$=$	$\sum$	
M2	$\varphi\lambda.T + \sum$	$=$	$\psi\lambda.T + \sum$	if $\varphi \Leftrightarrow \psi$
M3	$[x{=}p]\varphi\lambda.T + \sum$	$=$	$[x{=}p](\varphi\lambda.T)\{p/x\} + \sum$	
M4	$[x{\neq}p]\varphi\lambda.\sum' + \sum$	$=$	$[x{\neq}p]\varphi\lambda.[x{\neq}p]\sum' + \sum$	
S1	$\varphi\lambda.T + \sum$	$=$	$\varphi\lambda.T + \varphi\lambda.T + \sum$	
S2	$\varphi\lambda.T + \sum$	$=$	$[x{=}p]\varphi\lambda.T + [x{\neq}p]\varphi\lambda.T + \sum$	
S3	$\varphi n(x).S + \varphi n(x).T + \sum$	$=$	$\varphi n(x).S + \varphi n(x).T + \varphi n(x).([x{=}p]\tau.S + [x{\neq}p]\tau.T) + \sum$	
T1	$\varphi\tau.\sum$	$=$	$\varphi\sum$	
T2	$\sum + \varphi\tau.\sum$	$=$	$\sum$	
T3	$\phi\lambda.(\varphi\tau.T + \sum) + \sum'$	$=$	$\phi\lambda.(\varphi\tau.T + \sum) + \phi\varphi\lambda.T + \sum'$	

Fig. 1. Axioms for the Weak Bisimilarity.

5 Proof System

A complete equational system for the strong asynchronous bisimilarity is given in [ACS98]. Such a system for the weak asynchronous bisimilarity has not been available. The problem in generalizing a result from the strong case to the weak case could be an indication that something is not quite right. The difficulty in designing an equational system for the weak asynchronous bisimilarity is due to the lack of the output prefix operator. This is unfortunate since the role of the output prefix operation is to impose orders on interactions. Its relationship to asynchrony, or synchrony for that matter, was not intended. Our approach disowns this problem.

The expansion law plays a crucial role in proof systems. It is about how to convert two concurrent choice terms to one choice term. Suppose S, T are respectively the guarded choices $\sum_{i\in I}\varphi_i\lambda_i.S_i$ and $\sum_{j\in J}\psi_j\lambda_j.T_j$. Then

$$S \,|\, T = \sum_{i\in I}\varphi_i\lambda_i.(S_i \,|\, T) + \sum_{i\in I, j\in J}^{\lambda_i=m(x),\lambda_j=\overline{n}p}\varphi_i\psi_j[m{=}n]\tau.(S_i\{p/x\} \,|\, T_j)$$
$$+ \sum_{j\in J}\psi_j\lambda_j.(S \,|\, T_j) + \sum_{i\in I, j\in J}^{\lambda_j=m(x),\lambda_i=\overline{n}p}\varphi_i\psi_j[m{=}n]\tau.(S_i \,|\, T_j\{p/x\}).$$

Let AS be the equational system defined in Figure 1 plus the expansion law. In Figure 1 the notation $\sum$ stands for $\sum_{i\in I}\varphi_i\lambda_i.T_i$ and $\sum'$ for $\sum_{j\in J}\psi_j\lambda_j.T_j$. Accordingly $\varphi\sum$ should be understood as $\sum_{i\in I}\varphi\varphi_i\lambda_i.T_i$. Our axiomatic system differs from the standard one in that it is defined in terms of the guarded choice operator rather than the unguarded choice operator.

In AS we may rewrite terms to normal forms, whose definition is given next.

Definition 5. *Let $\mathcal{F}$ be $gn(T) \cup fv(T)$. A finite π-term T is a* normal form *on $\mathcal{F}$ if $T \equiv \sum_{i\in I} \varphi_i \lambda_i.T_i$ such that for each $i \in I$ one of the followings holds.*

1. *If $\lambda_i = \tau$ then T_i is a normal form on $\mathcal{F}$.*
2. *If $\lambda_i = \overline{n}m$ then T_i is a normal form on $\mathcal{F}$.*
3. *If $\lambda_i = \overline{n}(c)$ then $T_i \equiv [c{\notin}\mathcal{F}]T_i^c$ for some normal form T_i^c on $\mathcal{F} \cup \{c\}$.*
4. *If $\lambda_i = n(x)$ then T_i is of the form*
$$[x{\notin}\mathcal{F}]T_i^{\neq} + \sum_{m\in\mathcal{F}} [x{=}m]T_i^m$$
such that $T_i^{\neq}$ is a normal form on $\mathcal{F} \cup \{x\}$ and, for each $m \in \mathcal{F}$, $x \notin fv(T_i^m)$ and T_i^m is a normal form on $\mathcal{F}$.

For the motivation of the above definition and the proof of the next lemma, the reader is referred to [FZ10].

Lemma 2. *If T is finite, then a normal form T' exists such that $AS \vdash T = T'$.*

AS is sound and complete for the weak bisimilarity on the finite π-terms.

Theorem 3. *Suppose S, T are finite. Then $S \simeq T$ iff $AS \vdash S = T$.*

Complete systems have been discussed in literature [MPW92,PS95,Lin95,FY03]. A recent account that fits more into the present context can be found in [FZ10]. Notice that our formulation of T2 is crucial for the completeness proof.

We now turn to $\simeq_{\mathbf{Asy}}$. Let $AS_{\mathbf{Asy}}$ be AS together with the following law
$$a(x).\overline{a}x = \mathbf{0}. \tag{5}$$
Apparently $AS_{\mathbf{Asy}}$ is sound for $\simeq_{\mathbf{Asy}}$. The first indication that (5) is complete is the validity of the saturation property.

Lemma 3 (saturation). *Suppose that T is a normal form. The following statements are valid in the $\pi_{\mathbf{Asy}}$-calculus for every assignment ρ whose domain of definition is disjoint from $bv(T)$.*

1. *If $T\rho \overset{\lambda}{\Longrightarrow} T'$ and λ is not an input action, then $AS_{\mathbf{Asy}} \vdash T = T + \varphi\lambda'.T'$ for some φ, λ' such that $\varphi\rho \Leftrightarrow \top$ and $\lambda'\rho = \lambda$.*
2. *If $T\rho \overset{ae}{\Longrightarrow} T'$, $e \notin n(T\rho)$ and $z \notin v(t)$, then $AS_{\mathbf{Asy}} \vdash T = T + \varphi n(z).T'\{z/e\}$ for some φ, n such that $\varphi\rho \Leftrightarrow \top$ and $\rho(n) = a$.*

Proof. If $T \overset{\lambda}{\Longrightarrow} T'$ makes use of the rule (3) k times, then it is easy to see that $T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x \overset{\lambda}{\Longrightarrow} T'$. By Lemma 2 there is some normal form T_1 such that $AS \vdash T_1 = T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x$. By the standard approach it is easy to establish that $AS_{\mathbf{Asy}} \vdash T_1 = T_1 + \lambda.T'$. Thus $AS_{\mathbf{Asy}} \vdash T = T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x = T_1 = T_1 + \lambda.T' = T + \lambda.T'$. Notice that according to (5) the equality $T = T \,|\, a(x).\overline{a}x$ follows from $T = T \,|\, \mathbf{0}$, which in turn follows from the expansion law.

This simple argument should be enough for an informed reader. □

The proof of the completeness theorem is an induction on the complexity of the normal forms. The depth $dep(T)$ of a normal form T is defined as follows:

$$\begin{aligned} dep(\mathbf{0}) &\stackrel{\mathrm{def}}{=} 0, \\ dep(\varphi n(x).T) &\stackrel{\mathrm{def}}{=} dep(T) + 2, \\ dep(\varphi\lambda.T) &\stackrel{\mathrm{def}}{=} dep(T) + 1, \text{ if } \lambda \text{ is not an input}, \\ dep(\sum_{i\in I} \varphi_i\lambda_i.T_i) &\stackrel{\mathrm{def}}{=} \max\{dep(\varphi_i\lambda_i.T_i)\}_{i\in I}. \end{aligned}$$

For a finite term T, $dep(T)$ is defined by $dep(T')$, where T' is a normal form of T. It is worth remarking that the depth for input prefix is greater than that for output, bounded output and tau prefixes. This is important to the next proof.

Theorem 4 (completeness). *For finite S,T, $S \simeq_{\mathbf{Asy}} T$ iff $AS_{\mathbf{Asy}} \vdash S = T$.*

Proof. Suppose $P \equiv \sum_{i\in I} \varphi_i\lambda_i.S_i$ and $Q \equiv \sum_{j\in J} \psi_j\lambda_j.T_j$ are normal forms such that $P \simeq_{\mathbf{Asy}} Q$. If $a(x).S_i$ is a summand of P, then $P \xrightarrow{ac} S_i\{c/x\}$. The process Q has to simulate this action in the following manner $Q \Longrightarrow Q_1 \xrightarrow{ac} Q_2 \Longrightarrow Q'$. It is easy to see that $dep(Q_1) \leq dep(Q)$ and $dep(Q') \leq dep(Q_2)$. If $Q_1 \xrightarrow{ac} Q_2$ is not derived from the rule (3), then clearly $dep(Q_2) < dep(Q_1)$. If it is derived from the rule (3), then $AS \vdash Q_2 = Q_2'$ for some normal form Q_2', using the expansion law. It is obvious that $dep(Q_2) = dep(Q_2') \leq dep(Q_1) + 1$ by the definition of the depth function. But $dep(S_i\{c/x\}) \leq dep(P) - 2$ by definition. Hence $dep(Q') + dep(S_i\{c/x\}) < dep(Q) + dep(P)$. So $AS_{\mathbf{Asy}} \vdash Q' = S_i\{c/x\}$ by induction hypothesis.

The above oversimplified account is meant to bring out the fact that the depth function $dep(\_)$ does allow the standard inductive proof to go through. Consult [FZ10] for the general idea of the completeness proof. □

6 Conclusion

The notion of theory probably should have been introduced long time ago. A theory is essentially an implementation. In practice it is one of the most important concepts to start with. Theories can be defined for process calculi other than the π-calculus. The most interesting theories are defined in complete process calculi [Fu10]. In a complete model the Peano arithmetic can be defined in a robust manner. Example 2 through Example 5 of Section 3 would not make a lot of sense for incomplete models like CCS (the incompleteness of CCS is established in [Fu10]).

There are several directions one can pursue to further the study of theories. Let's mention a couple of them. Firstly it is worthwhile to carry out a comparative research into theories. Honda and Yoshida [HY95] have introduced a different notion of theory. They extended the notion of λ-theory [Bar84] to the framework of process models. A theory in their sense is a set of equalities closed under the process operations of the frame calculus. Their starting point is more algebraic than logical, whereas our motivation is less algebraic than logical. It is apparent that a theory in the present setting is a theory of Honda and Yoshida. Unlike in their treatment, the algebraic property of our theories comes for free. It would be interesting to investigate the relationship between these two approaches.

Secondly it is interesting to look for complete proof systems for general theories. Suppose $\mathbf{A}$ is a theory. Let $AS_{\mathbf{A}}$ be obtained by combining AS with the following laws for $\mathbf{A}$.

$$A = \mathbf{0}, \text{ for every } A \in \mathbf{A}. \tag{6}$$

For which recursive theories for instance is $AS_{\mathbf{A}}$ complete? It is easy to come up with theories for which $AS_{\mathbf{A}}$ does not appear to be complete. Take for instance the theory $\mathbf{A_a} = \{\overline{a}(c).\overline{a}(c)\}$. It is unlikely that $AS_{\mathbf{A}_a} \vdash \overline{a}(c) = \mathbf{0}$. In fact it follows from Sewell's nonaxiomatisability result [Sew97] that it must be incomplete if the frame calculus is CCS. It remains to check if this negative result is also valid when the frame calculus is π. A more difficult task is to answer the question if there exists a recursive theory $\mathbf{A}$ such that $\simeq_{\mathbf{A}}$, when restricted to the finite terms, has no complete recursive equational systems whatsoever. These are issues to be investigated in future.

Acknowledgments

This work has been supported by NSFC (60873034). The author is grateful to the anonymous referees for their comments and suggestions.

References

[Abr93] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[ACS98] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195:291–324, 1998.

[Bar84] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, 1984.

[Bou92] G. Boudol. Asynchrony and the π-calculus. Technical Report RR-1702, INRIA Sophia-Antipolis, 1992.

[CF10] X. Cai and Y. Fu. The λ-calculus in the π-calculus, 2010.

[FL10] Y. Fu and H. Lu. On the expressiveness of interaction. *Theoretical Computer Science*, 411:1387–1451, 2010.

[Fu10] Y. Fu. Theory of interaction, 2010.

[FY03] Y. Fu and Z. Yang. Tau laws for pi calculus. *Theoretical Computer Science*, 308:55–130, 2003.

[FZ10] Y. Fu and H. Zhu. The name-passing calculus, 2010.

[HO95] M. Hyland and L. Ong. Pi-calculus, dialogue games and pcf. In *Proc. 7th ACM Conference on Functional Programming Languages and Computer Architecture (FPCA 1995)*, pages 96–107, 1995.

[HT91a] K. Honda and M. Tokoro. An object calculus for asynchronous communications. In *Proc. ECOOP'91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147, Geneva, Switzerland, 1991.

[HT91b] K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proc. Workshop on Object-Based Concurrent Computing*, volume 615 of *Lecture Notes in Computer Science*, pages 21–51, 1991.

[HY95] K. Honda and M. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151:437–486, 1995.

[Lin95] H. Lin. Complete inference systems for weak bisimulation equivalences in the π-calculus. In *Proceedings of Sixth International Joint Conference on the Theory and Practice of Software Development*, volume 915 of *Lecture Notes in Computer Science*, pages 187–201, 1995.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mil92] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–146, 1992.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–40 (Part I), 41–77 (Part II), 1992.

[PS95] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120:174–197, 1995.

[Sew97] P. Sewell. Nonaxiomatisability of equivalence over finite state processes. *Annals of Pure and Applied Logic*, 90:163–191, 1997.

[Wal95] D. Walker. Objects in the π-calculus. *Information and Computation*, 116:253–271, 1995.