# Theory Defined by Process

Yuxi Fu

BASICS, Department of Computer Science
Shanghai Jiaotong University, Shanghai 200240, China

**Abstract.** Theories defined in a process model are formalized and studied. A theory in a process calculus is a set of well-founded processes that are perpetually available, 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 $\pi$-calculus as a theory, the asynchronous theory, in the $\pi$-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 $\pi$.

## 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. 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. The issue to address is this: A model of interaction, say the $\pi$-calculus, is a *closed* system of interactants. A $\pi$-process only interacts within the model. It does not interact with anything that is not a $\pi$-process. In this closed world view, how and in what fashion are the network services interpreted? Our basic idea is to regard the kind of service 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. 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 $P$ siting among $!S_1, !S_2, \ldots$ can always be inspected by considering the finite action sequences of the shape

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

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 $P \,|\, S' \xrightarrow{\lambda} P'$ for $S' \in \mathcal{S}$. Now the crucial point is that the services really stay invisibly in the background. So from $P \,|\, S' \xrightarrow{\lambda} P'$ we shall be able to deduce $P \xrightarrow{\lambda} P'$ and the equivalence between $P \,|\, S'$ and $P$. We are on our way to a theory by process.

From a model theoretical point of view, sometimes we need to assume that we are working with open systems within a closed world. In a real world, there are already a set of processes in operation. These processes could be the standard functionalities provided by the hardware platform, or the procedures of an operating system. As practitioners, we are not very much interested in the background functionalities or the operating procedures. Most of time we are working with a subsystem enjoying a certain degree of closedness. In other words, we focus on a variant (subsystem) of a calculus (frame system) defined within the calculus, not outside the calculus. Such a variant is both a subcalculus and an internal calculus. It is a subcalculus because the objects of study form a proper subset of the objects of the frame calculus. It is also an internal calculus because it does not make the best of a sense when viewed outside the frame calculus. Some of the process calculi proposed in literature are actually internal calculi. One such calculus is discussed next.

The asynchronous $\pi$-calculus [Bou92,HT91a,HT91b,HY95,ACS96] is obtained from the synchronous $\pi$-calculus by dropping the continuations for output actions. At first look this appears a bit simplistic since asynchrony can not be a syntactical issue. The idea is however that the output process $\overline{a}c.P$ no longer engages in any interactions. The asynchronous emission of $c$ at $a$ is perceived as an internal action of the process. Hence the following semantic definition

$$\overline{a}c.P \xrightarrow{\tau} \overline{a}c \,|\, P$$

Under the new semantics, it is easily seen that $\overline{a}c.P$ is observationally equivalent to $\overline{a}c \,|\, P$. So in the asynchronous $\pi$-calculus the continuation $P$ of the output prefix process $\overline{a}c.P$ is completely dropped. The observational theory of the asynchronous $\pi$-calculus has been based on the following understanding:

> Since an observer does not know when an output primitive is consumed by the observee, it might as well assume that it does not know at all if the process $\overline{a}c$ has acted or not.

This leads to an equational theory in which only the output actions of the observees are observable. In such a theory we have typically the equivalence

$$a(x).(\overline{a}x \,|\, T) \simeq_a T$$

where $\simeq_a$ denotes the asynchronous bisimilarity. But the idea that an observer is not aware of the action of $\overline{a}c$ simply because it has no continuation is not without debate. Why is it that in the synchronous $\pi$-calculus, the action of $\overline{a}c.\mathbf{0}$ is observable? Well it is observable because it can engage in an interaction. In the observational theory of process an observation is nothing more than an interaction. Everything that participates in an interaction is deemed observable. If we apply this fundamental view to the asynchronous $\pi$-calculus, we are forced to conclude that in $a(z).T \,|\, (c)(\overline{a}c \,|\, Q) \xrightarrow{\tau} (c)(T\{c/z\} \,|\, Q)$, both $a(z).T$ and $(c)(\overline{a}c \,|\, Q)$ participate in the interaction and therefore both are observed during the procedure. If an interaction is a cooperation between an input process and an output process, it does not seem to make too much sense to say that one is observable and the other is not. There are asynchronous communications, there are nonetheless no asynchronous interactions. Observations are about interactions; they are not about communications.

There ought to be an alternative explanation for the observational theory of the asynchronous $\pi$-calculus. 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 through channel $a$ in a later time. What is then the background environment? A process calculus is a closed model in the sense that a process of the model only interacts with processes that stay inside the same model. The background environment has to be interpreted as a set of processes within the model. The asynchronous $\pi$-calculus is better seen as an internal calculus of the $\pi$-calculus. The asynchronous communications are achieved by running a set of processes of the form $!a(x).\overline{a}x$ in the background environment.

One may also study proof theory in process models [Abr93]. A particular logical system can be formulated as a theory in a process model. An element of the theory codes up either an axiom or a rule of the logic. A process of the form $!A$ can be seen as a proposition. The process $A$, which must be well-founded, is a proof of $!A$. A logical reasoning making use of the proposition must interact with a copy of $A$ to verify the proof. According to the well-known Curry-Howard correspondence, a proposition $!A$ can be interpreted as a type, or a class in an object oriented paradigm [Wal95]. A class has a set of methods of the form $!d(x).D$. The method $!d(x).D$ can be invoked by an object of the form $\overline{d}e.O$ that supplies the method with the parameter $e$. The implementation of a programming language amounts to defining a theory that code up the system functions or methods provided by the language. The equivalence of two pieces of programs written in the language are judged in the presence of the theory.

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. Section 2 defines the $\pi$-calculus, the frame calculus used in this paper. Section 3 gives a formal treatment of the theories. Section 4 relates the asynchronous theory to the asynchronous $\pi$-calculus. Section 5 provides a complete equational system for the asynchronous theory. Section 6 concludes.

## 2 Pi Calculus

The $\pi$-calculus of Milner, Parrow and Walker [MPW92] has been widely studied and used in both theory and in practice. Our presentation of the calculus 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 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 $l, m, n, o, p, q$.

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 $\phi, \varphi, \psi$. We write $\varphi \Leftrightarrow \top$ ($\varphi \Leftrightarrow \bot$) if $\varphi$ is evaluated to the truth value $\top$ (the false value $\bot$) no matter how the name variables appeared in the condition are instantiated. Similarly we can define $\varphi \Rightarrow \psi$ and $\varphi \Leftrightarrow \psi$.

Our definition of the $\pi$-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 the *terms* is inductively constructed from the following grammar:

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

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 $\tau$ 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. According to our definition it does not make sense to write for example $T + \mathbf{0}$. 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 $c$ appeared underneath the localization operator $(c)$ or a bound output prefix $\overline{a}(c)$ is local. It is global if it is not local. A name variable $x$ is bound if it is underneath an input prefix say $n(x)$. It is free if it is not bound. Both bound names and local names are subject to $\alpha$-conversion. We will use the functions $gn(\_), ln(\_), fv(\_), bv(\_)$ with the obvious meanings. A process is a term in which all 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 *regular*.

The operational semantics of this $\pi$-calculus is defined by the following labeled transition system.

*Action*

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

*Composition*

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

*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*

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

Let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$. Let $\overset{\widehat{\lambda}}{\Longrightarrow}$ be $\Longrightarrow$ if $\lambda = \tau$ and be $\Longrightarrow\xrightarrow{\lambda}\Longrightarrow$ 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.*

An example of bisimulation equality is $!\pi.T \simeq \pi.(T\,|\,!\pi.T)$. We write $S \simeq T$ if $S\{a_1/x_1,\ldots,a_n/x_n\} \simeq T\{a_1/x_1,\ldots,a_n/x_n\}$ for every substitution whose domain of definition $\{x_1,\ldots,x_n\}$ is $fv(S\,|\,T)$. For the particular $\pi$-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-$\tau$ 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 $\lambda$ is an input action and $\lambda'$ 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 **A** *is a 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 model. Operationally an axiom $A$ can be identified to the process $!A$. Every proof in the model 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 **A** is *finite* if it is a finite set. It is *finitely presentable* if it can be generated from a finite theory. By **A** being generated from **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 regular (functional) if all its axioms are regular (functional).

Let's see some examples.

*Example 1.* The asynchronous theory **Asy** is defined by the finitely presentable theory $\{a(x).\overline{a}x \mid a \in \mathcal{N}\}$. In the presence of **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 $a, b$.

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

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

Here $p$ is the access name for the numbers. The notation $\bot$ is a name that denotes false (we may use a special name $\top$ to denote truth). The natural numbers are underlined to avoid any confusion. We write $\overline{a}(\underline{i})$ for $(p)(\overline{a}p \,|\, [\![\underline{i}]\!]_p)$. It is routine to define processes $SUCC_a$, $EQ_a$ and $L_a$ that implement the successor function, 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 using 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 and only if $i < j$; otherwise it returns $\bot$ at $d$. Let **Pa** be the theory

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

It is an implementation of the Peano arithmetic in $\pi$-calculus. For detail the reader could consult [FZ10] for the encodings of these processes.

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

$$\{e(v).v(y).(\ldots\overline{v}\_\ldots), d(v).v(z).(\ldots\overline{v}\_\ldots)\},$$

in which $e(v).v(y).(\ldots\overline{v}\_\ldots)$ is the encryption function whose action sequences are all of the form

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

for some number $i$ and $d(v).v(z).(\ldots\overline{v}\_\ldots)$ is the decryption function whose action sequences are all of the form

$$d(v).v(z).(\ldots\overline{v}\_\ldots) \xrightarrow{dc} \xrightarrow{cf} \xRightarrow{\tau} \xrightarrow{\overline{c}(g)} \simeq [\![\underline{j}]\!]_g$$

for some number $j$. The theory **Crp** provides the encryption and decryption functionalities every process can make use of.

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

$$a(v).(c)(\overline{c}(\underline{0}) \mid !c(x).(\overline{c}(d).\overline{d}x + \overline{v}x)). \tag{1}$$

Upon receiving a private channel provided by a user, the process (1) randomly generates a number and sends it to the user through the private channel. However process (1) may diverge. So it is not an axiom according to our definition. The theory **Ran** can be defined by the finite set $\{\overline{g}(\underline{0}), g(x).\overline{g}(p).(q)(\overline{p}q \mid \overline{q}x))\}$. It can also be defined by the infinite set

$$\{\overline{g}(\underline{0}), \overline{g}(\underline{1}), \ldots, \overline{g}(\underline{i}), \ldots\}.$$

So we are faced with choices of theories that provide equivalent functionality.

*Example 5.* The $\pi$-calculus has been used both as a specification language and a machine language. The rational behind these practices is that $\pi$ is expressive enough to qualify for a machine model. Now if we think of $\pi$-calculus as a machine model, we can talk about implementations of programming languages on $\pi$-calculus. This is precisely what is done in [Wal95]. Formally what is then an interpreter of a higher order programming language on $\pi$? 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 resolve issues of this kind.

### 3.1 Operational Semantics

To investigate the algebraic properties of the theories, we need to define the operational semantics of the theories first. Given a theory $\mathbf{A}$, a process $P$ under the theory $\mathbf{A}$ can be imagined as the least fixpoint of the following equation

$$P = P \mid \prod_{A \in \mathbf{A}} !A.$$

This recursive definition is not always attainable at the syntactical level since it makes use of a possibly infinite composition. But notice that $P \mid A$, for each $A \in \mathbf{A}$, is also the least fixpoint of the equation. This observation motivates the treatment of the operational semantics of $\mathbf{A}$ by extending that of $\pi$. The semantics of the theory $\mathbf{A}$, or the $\pi_{\mathbf{A}}$-calculus, extends the operational semantics of $\pi$ with the following rule:

$$\frac{P \mid A \xrightarrow{\lambda} P'}{P \xrightarrow{\lambda} P'} \; A \in \mathbf{A}. \tag{2}$$

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

**Definition 3.** *A symmetric relation $\mathcal{R}$ on $\mathcal{P}$ is a* weak $\mathbf{A}$-bisimulation *if $Q \stackrel{\widehat{\lambda}}{\Longrightarrow} Q'\mathcal{R}P'$ 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.* By definition $\simeq$ is an $\mathbf{A}$-bisimulation. The strictness is witnessed by the equivalence $A \simeq_{\mathbf{A}} \mathbf{0}$ whenever $A \in \mathbf{A}$. □

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 equation:

$$\mathbf{0} = \prod_{A \in \mathbf{A}} !A. \tag{3}$$

Equality (3) has been exploited to define the semantics of the asynchronous $\pi$-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 (2) applied to **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 $\forall P{\in}\mathcal{P}.\exists A{\in}\mathbf{A}.P = A$, 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.** $\simeq_{\mathbf{A}}$ *equals* $\simeq_{\mathbf{B}}$ *if and only if* $\mathbf{A}_{ker} = \mathbf{B}_{ker}$*.*

*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 a theory is completely 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 tricky to establish the consistency of a theory. Let $\mathbf{F}$ be the theory consists of all the regular 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 $\pi$

We prove in this section that **Asy** provides a faithful account of the asynchronous $\pi$-calculus. We adopt the following grammar for the asynchronous $\pi$-calculus, which summarizes the essential feathers of the calculi defined in literature [Bou92,HT91a,HT91b,HY95,ACS96].

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

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 $\pi$-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 $\pi$. Their operational semantics is defined by the following rules.

*Action*

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

*Composition*

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

*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*

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

In Honda and Tokoro's treatment the asynchronous $\pi$ differs from the synchronous $\pi$ at the operational level, whereas in Amadio, Castellani and Sangiorgi's treatment it is at the observational level. The definition of the asynchronous bisimilarity [ACS96] appears odd from the point of view of interaction.

**Definition 4.** *A symmetric relation $\mathcal{R}$ on the asynchronous $\pi$-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 $Q \xrightarrow{ab} Q'$ 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 $\pi$ is a syntactic subcalculus of $\pi$. It is also an operational variant of $\pi$ 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 [ACS96]. Their proof can be extended to produce a proof of the following theorem.

**Theorem 2.** *Suppose $S, T$ are asynchronous $\pi$-terms. Then $S \simeq_a T$ if and only if $S \simeq_{\mathbf{Asy}} T$.*

Theorem 2 can be interpreted as saying that the asynchronous $\pi$ is a syntactical simplification of $\pi_{\mathbf{Asy}}$. It perceives $\pi_{\mathbf{Asy}}$ as a self-contained closed model. It can also be seen as a justification of the asynchronous $\pi$ 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 $c = subj(\lambda)$ |
| L6 | $(c)(\varphi\overline{a}c.T + \sum) = (c)(\varphi\overline{a}(c).T + \sum)$ | | if $c \notin gn(\varphi)$ |
| 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 | $n(x).S + n(x).T + \sum = n(x).S + n(x).T + n(x).([x{=}p]\tau.S + [x{\neq}p]\tau.T) + \sum$ | | |
| 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 [ACS96]. 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. It has nothing to do with asynchronous communications. 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

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

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 ours 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 $gn(T) \cup fv(T)$. The $\pi$-term $T$ is a* normal form *if it is of the form $\sum_{i\in I} \lambda_i.T_i$ 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 regular, then a normal form $T'$ exists such that $AS \vdash T = T'$.*

$AS$ is sound and complete for the weak bisimilarity on the regular $\pi$-terms.

**Theorem 3.** *Suppose $S, T$ are regular. 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 $T$ is regular. If $T \stackrel{\lambda}{\Longrightarrow} T'$ in $\pi_{\mathbf{Asy}}$ and $\lambda$ is not an input action then $AS_{\mathbf{Asy}} \vdash T = T + \lambda.T'$.*

*Proof.* If $T \stackrel{\lambda}{\Longrightarrow} T'$ makes use of the rule (2) $k$ times, then it is easy to see that $T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x \stackrel{\lambda}{\Longrightarrow} T'$. By Lemma 2 there is some normal form $T_1, T_1'$ such that $AS \vdash T_1 = T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x$ and $AS \vdash T_1' = T'$. By the standard approach it is easy to establish $AS_{\mathbf{Asy}} \vdash T_1 = T_1 + \lambda.T_1'$. Thus $AS_{\mathbf{Asy}} \vdash T = T \,|\, a_1(x).\overline{a_1}x \,|\, \ldots \,|\, a_k(x).\overline{a_k}x = T_1 = T_1 + \lambda.T_1' = T + \lambda.T'$. 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. □

Our statement of the saturation property is greatly simplified. Normally it should also deal with the input actions. For the $\pi$-calculus it should also deal with the action sequences of the form $T\sigma \stackrel{\lambda}{\Longrightarrow} T'$ in which the action is enabled by

the substitution. But the simple statement of Lemma 3 suffices 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:

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

For a regular 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 regular $S, T$, $S \simeq_{\mathbf{Asy}} T$ 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 (2), then clearly $dep(Q_2) < dep(Q_1)$. If it is derived from the rule (2), 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. □

Can we generalize Theorem 4 to an arbitrary regular theory $\mathbf{A}$? Let $AS_{\mathbf{A}}$ be obtained by combining $AS$ with the following law for the regular theory $\mathbf{A}$.

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

The system $AS_{\mathbf{A}}$ is interesting in view of the following result.

**Lemma 4.** *$AS_{\mathbf{A}}$ is complete iff $AS_{\mathbf{A}} \vdash A = \mathbf{0}$ for every regular process in $\mathbf{A}_{ker}$.*

*Proof.* The axiom (6) allows us to expand a term of the form $T \,|\, A$, where $T$ is finite and $A \in \mathbf{A}$, to a normal term. So we may establish the saturation property (Lemma 3). The inductive proof of completeness makes use of a lexicographical order defined as follows: The pair $\langle S', T'\rangle$ is order less than $\langle S, T\rangle$ if the maximal number of the nested depth of the prefixes of $S'$ is strictly less than the maximal number of the nested depth of the prefixes of $S$. □

## 6 Conclusion

The notion of theory probably should have been introduced long time ago. It is in essence an application oriented concept. In practice it is one of the most important concepts to start with. There are several directions one can pursue concerning the notion of theory. Let's mention a couple of them. Firstly it is important to be able to prove that a given theory is consistent. One may look for a useful general result stating the consistency of a theory under certain properties. Secondly it is interesting to look for complete proof systems for general theories. For which regular theories for instance is $AS_{\mathbf{A}}$ complete? These are issues to be investigated in future.

## References


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

[ACS96] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous $\pi$ calculus. In *Proc. CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 1996.

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

[FL10] Y. Fu and H. Lu. On the expressiveness of interaction. *Theoretical Computer Science*, 411:1387–1451, 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. Name-passing calculus (`http://basics.sjtu.edu.cn/~yuxi/papers/`), 2010.

[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 $\pi$-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.

[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.

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