The Name-Passing Calculus

YUXI FU
and
HAN ZHU
Shanghai Jiao Tong University

Name-passing calculi are foundational models for mobile computing. Research into these models has produced a wealth of results ranging from relative expressiveness to programming pragmatics. The diversity of these results call for clarification and reorganization. This paper applies a model independent approach to the study of the name-passing calculi, leading to a uniform presentation and simplification. The technical tools and the results studied in the paper form the foundation for a theory of name-passing calculus.

Categories and Subject Descriptors: ... [...]: ...

General Terms: Theory, Languages

Additional Key Words and Phrases: Bisimulation, pi calculus, process calculus

Corresponding Author's postal address: BASICS, Department of Computer Science, Shanghai Jiao Tong University, 800 Dong Chuan Road, Shanghai 200240, China.
Corresponding Author's email address: fu-yx@cs.sjtu.edu.cn.

Contents

1. MOBILITY IN PRACTICE AND IN THEORY

Mobile calculi feature the ability to pass around objects that contain channel names. Higher order CCS [Thomsen 1989; 1990; 1993; 1995] for instance, is a calculus with a certain degree of mobility. In a mobile calculus, a process that receives an object may well make use of the names appeared in the object to engage in further interactions. It is in this sense that the communication topology is dynamic. It was soon realized that the communication mechanism that restricts the contents of communications to the channel names gives rise to a simple yet versatile model that is more powerful than the process-passing calculi [Sangiorgi 1992; 1993; 1996b; 1996a]. This is the π-calculus of Milner, Parrow and Walker [Milner et al. 1992]. See [Parrow 2001] for a gentle introduction to the model and the history of the name-passing calculus and [Sangiorgi and Walker 2001b] for a broader coverage. A seemingly innocent design decision of the π-calculus is to admit a uniform treatment of the names. This decision is however not supported by the semantics of the mobile calculi. From a process term T one could construct the input prefix term

$$a(x).T \tag{1}$$

and the localization term

$$(x)T. \tag{2}$$

According to the definition of π-calculus, the semantics of x appeared in (1) is far different from that of x in (2). In the former x is a name variable, or a dummy name, that can be instantiated by an arbitrary name when the prefix engages in an interaction. In the latter x is a local name that can never be confused with another name. The input prefix forces the unbound name x in T to be a name variable, whereas the localization operator forces the unbound name x in T to be a constant name. This apparent contradiction is the reason for all the semantic complications of π-calculus. And nothing has been gained by these complications. In what follows we take a look at some of the problems and inconveniences caused by the confusion.

To begin with, the standard operational semantics of π-calculus has been flawed from the outset. An extremely useful command in both practice and theory is the two leg if-statement *if* φ *then* S *else* T. In mobile calculi this can be defined by introducing the conditional terms $[x{=}y]T$ and $[x{\neq}y]T$. The semantics of these terms have been defined respectively by the match rule

$$\frac{T \xrightarrow{\lambda} T'}{[x{=}x]T \xrightarrow{\lambda} T'} \tag{3}$$

and the mismatch rule

$$\frac{T \xrightarrow{\lambda} T'}{[x{\neq}y]T \xrightarrow{\lambda} T'}\ x \neq y. \tag{4}$$

Rule (4) is unusual since it has an unusual side condition. How should we understand the side condition $x \neq y$? If x, y were constant names, the side condition of (4) would be pointless because $x \neq y$ would be evaluated to logical truth. The reason that (4) is necessary is precisely because x, y cannot be understood as constant

names in the uniform treatment of the names. The correct reading of (4) is that "$[x{\neq}y]T \xrightarrow{\lambda} T'$ is admissible under the logical assumption $x \neq y$". It should now be clear that the popular semantics fails to support the following equivalence

$$T = [x{=}y]T + [x{\neq}y]T. \tag{5}$$

To see this, let $=$ be $\sim$, the strong early bisimilarity of [Milner et al. 1992]. According to the definition, $T \sim [x{=}y]T + [x{\neq}y]T$ if and only if $T\sigma\dot{\sim}([x{=}y]T + [x{\neq}y]T)\sigma$ for every substitution σ, where $\dot{\sim}$ is the strong ground bisimulation equivalence. If σ is the identity substitution, it boils down to establishing the following equivalence

$$T \dot{\sim} [x{=}y]T + [x{\neq}y]T. \tag{6}$$

We may prove (6) under the assumption $x \neq y$. But we cannot prove (6) under the assumption $x = y$ since rule (3) does not allow us to do that. The failure of (5) necessarily implies that a number of the axiomatic systems for the π-calculus with the mismatch operator studied in literature are actually *not* sound since equality (5) is either an axiom or a derivable law in all these systems. A related mistake is to introduce a boolean evaluation function $beval(\_)$ whose inductive definition includes following clauses:

$$beval(x{=}y) \stackrel{\text{def}}{=} \bot,$$
$$beval(x{\neq}y) \stackrel{\text{def}}{=} \top.$$

This would lead to the axioms

$$[x{=}y]T \;=\; \mathbf{0},$$
$$[x{\neq}y]T \;=\; T,$$

which are ridiculous if observational equivalences are closed under prefix operations. Even without mismatch operator, the standard definition of the operational semantics is problematic. Take for example the following instance of the expansion law

$$\overline{x}x \,|\, y(v) = \overline{x}x.y(v) + y(v).\overline{x}x + [x{=}y]\tau. \tag{7}$$

The right hand side of (7) can do a τ-action under the assumption $x = y$. However the operational semantics of π-calculus does not admit a τ-action of the left hand side of (7) even if the logical assumption $x = y$ is made.

The correct formulation of the operational semantics of the π-calculus is given by Lin [1995a; 1995b; 1996; 1998; 2003], using the symbolic approach developed by Hennessy and Lin [1995]. In the symbolic semantics, one has that $[x{\neq}y]T \overset{x\neq y,\lambda}{\longrightarrow} T'$, meaning that the action is admissible under the logical assumption $x \neq y$. Similarly one has $[x{=}y]T \overset{x=y,\lambda}{\longrightarrow} T'$. Notice that this transition is very different from the transition $[x{=}x]T \xrightarrow{\top,\lambda} T'$. In the symbolic approach the action $T \xrightarrow{\top,\lambda} T'$ is simulated by the combined effect of $[x{\neq}y]T \overset{x\neq y,\lambda}{\longrightarrow} T'$ and $[x{=}y]T \overset{x=y,\lambda}{\longrightarrow} T'$, not by any single action sequence of $[x{=}y]T + [x{\neq}y]T$. Similarly $\overline{x}x \,|\, y(v) \overset{x=y,\tau}{\longrightarrow} \mathbf{0} \,|\, \mathbf{0}$ can be derived from $\overline{x}x \xrightarrow{\top,\overline{x}x} \mathbf{0}$ and $y(v) \xrightarrow{\top,yx} \mathbf{0}$. If we think of it, the symbolic semantics not only provides the correct labeled transition semantics upon which we may study

the observation theory of the π-calculus, but also makes clear the problem caused by the confusion of the name variables and the names.

Secondly the observational theory of the mobile processes is made more complex than it is. One well-known phenomenon is that, unlike in CCS [Milner 1989a; Milner and Sangiorgi 1992], some standard definitions of process equivalence lead to different equality relations. The standard definition of bisimulation gives rise to ground bisimilarity that is not closed under input prefix operation. The solution proposed in [Milner et al. 1992] is to take the substitution closure. The resulting relation is the early equivalence. If substitution closure is required in every bisimulation step, one obtains Sangiorgi's open bisimilarity [Sangiorgi 1996c]. The open bisimilarity is strictly finer than the early equivalence, which is in turn much finer than the ground bisimilarity. The open bisimilarity can be further improved to quasi open bisimilarity [Sangiorgi and Walker 2001a], which lies nontrivially between the open bisimilarity and the early equivalence. The barbed equivalence can be defined by placing substitution closure at the beginning of bisimulations, which gives rise to the equivalence studied by Milner and Sangiorgi [1992]. It can also be defined by requiring that every bisimulation step should be closed under substitutions of names. It is shown by Sangiorgi and Walker [2001a] that the latter coincides with the quasi open bisimilarity. It is easy to see that the barbed equivalence is weaker than the early equivalence. It is not yet clear however if it is subsumed by the early equivalence. Putting aside the issue of which of these equivalences is more authoritative than the others, we would like to point out that the substitution closure requirement is an algebraic requirement rather than an observational requirement. From the true spirit of the observation theory, an environment can never detect any difference between $a(x).\overline{b}c + \overline{b}c.a(x)$ and $a(x) \,|\, \overline{b}c$, since it can never force the distinct names a, b to be equal. This issue of reconciling the inconsistency between the observational view and the algebraic view must be addressed to achieve a better theory of the mobile processes.

The algebraic requirement also makes testing theory of mobile processes hard to comprehend. In the testing theory developed by De Nicola and Hennessy [1984], the behaviors of a process are judged by testers. Two processes are testing equivalent if no testing can detect any behavioral difference between them. Like the bisimulation approach, the testing approach fails to give rise to a reasonable equivalence on the mobile processes. In order to respect the name uniformity and obtain a useful equivalence at the same time, the algebraic condition must be imposed. See [Boreale and De Nicola 1995] for more on this issue. In some sense the substitution closure condition completely defeats the philosophy of the testing theory.

In retrospect, the confusion of the names and the name variables is not out of the desire to model mobility, since mobility can be achieved by using the name variables any way. If channels have a physical existence, computations or interactions really should not manipulate channels. What they are supposed to do is to make use of the channels for the purpose of interaction. According to this interpretation, all channel names ought to be constant. To model mobility, the introduction of a dichotomy between the names and the name variables is not only an obvious choice, it is the only choice. The variables are there for mobility.

In theory of expressiveness, the name dichotomy provides a basis for comparing

the relative expressive powers of calculi. The straightforward translation from CCS to π-calculus for instance is fully abstract if in π-calculus a line is drawn between the names and the name variables. The translation takes the equivalent CCS processes, say $a \,|\, \overline{b}$ and $a.\overline{b} + \overline{b}.a$, to the equivalent π-processes $a(x) \,|\, \overline{b}(y)$ and $a(x).\overline{b}(y) + \overline{b}(y).a(x)$. If the names are treated uniformly, the target model would have a much stronger process equality than the source model. In such a framework it is not even clear if a reasonably good fully abstract translation from CCS to π-calculus exists. Other expressiveness results can also be best interpreted using the name dichotomy. Sangiorgi-Thomsen's encoding of the higher order CCS in the π-calculus is another example. The process variables of the higher order CCS are translated to the name variables of the π-calculus, while the names of the former *are* the names of the latter. This encoding is shown to be fully abstract by Sangiorgi [1992; Sangiorgi [1993]. Again if the names of the π-calculus are treated uniformly, the encoding would not even be sound. We could give more examples to support the proposition that a dichotomic understanding should be preferred. But the point is already made. The names play a universal role in process theory. Without the assumption that all names are constant, expressiveness results about process calculi are bound to be chaotic [Nestmann 2006].

When applying the mobile calculi to interpret programming phenomena, the name dichotomy has always been enforced. It is sufficient to give just one example. An early work was done by Walker [1991; Walker [1995], who defines the operational semantics of an object oriented language in terms of the operational semantics of the π-calculus. The idea of the interpretation can be summarized as follows. An object is modeled by a prefix process of the form $objn(x).O$, where $objn$ is the name of the object. A method is interpreted as a replicated process of the form $!mthd(z).M$, where $mthd$ is the method name. The method can be invoked by a process of the form $\overline{mthd}(v).P$ that supplies the value v to the method parameter. Without going into details, it is already obvious that for this interpretation to make sense, it is important to maintain a distinction between the names and the name variables. We could give many other applications of the mobile calculi. But it suffices to say that in all these applications, there is a clear cut distinction between the names and the name variables.

The above discussions lead to the conclusion that, for both theoretical and practical reasons, the π-calculus should be defined using the name dichotomy. The dichotomy has been introduced in literature using type systems. If one thinks of the type of a channel name as defining the interface property of the channel, then the type theoretical solution does not seem appropriate since the difference between a name and a name variable is not about interface property. It is our view that the issue should be treated at a more fundamental level.

This paper presents a simple theory of the π-calculus. The simplification is achieved in a number of ways. Apart from the consideration on the issue about the names, a general motivation for what are to be presented in the paper is that all proper process calculi ought to share a core theory that is model independent. By developing such a core theory, a lot of the investigations into a particular process calculus are rendered unnecessary and the comparisons of different process calculi and variants are made possible. A study into the core theory is carried out in [Fu

2010b]. The present paper is an application of that general theory to the name-passing calculi. The following three goals will be achieved:

—We show that a concise operational semantics of the π-calculus is available.

—We demonstrate that the observational theory of the π-calculus is far less diverse than it has been perceived.

—We confirm that the algebraic theory of the π-calculus is simpler than it has been suggested.

The above claims are supported by the technical contributions summarized as follows:

—A general model independent process equality, the absolute equality, is applied to the π-calculus. It is proved that the well known bisimulation equivalences of the π-calculus, mentioned in this introduction, either coincide with a weak version of the absolute equality or should be ignored.

—A model independent equivalence, the box equality, is defined and applied to the π-calculus. It is demonstrated that this new equivalence coincides with a well known rectification of the testing equivalence in the π-calculus.

—Two complete proof systems for the set of the finite π-processes are presented, one for the absolute equality, the other for the box equality.

The rest of the paper is organized into five sections. Section 2 defines our version of the π-calculus. Section 3 studies the model independent observation theory of the π-calculus. Section 4 discusses the relative expressiveness of some well known variants of the π-calculus. Section 5 presents a uniform account of the proof systems for the finite π-processes. Section 6 points out how a theory of π-calculus can be developed using the framework set up in this paper.

In order to make the paper self-contained, some of the definitions and motivating remarks from [Fu 2010b] are reiterated in this paper. Most of the lammas are stated without proofs. A well-informed reader would have no problems in supplying the proofs by oneself.

2. PI CALCULUS

We assume that there is an infinite countable set $\mathcal{N}$ of *names*, an infinite countable set $\mathcal{N}_v$ of *name variables*. These sets will be ranged over by different lower case letters. Throughout the paper the following conventions will be enforced:

—The set $\mathcal{N}$ is ranged over by a, b, c, d, e, f, g, h.

—The set $\mathcal{N}_v$ 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 name variable acts as a place holder that need be substantiated by a name. By its very nature, a name variable can not be used as a channel for interaction. Similarly it can not be used as a message passed around in a communication.

2.1 Process

To give a structural definition of processes, we need to introduce terms. The set $\mathcal{T}$ of π-*terms* is inductively generated by the following BNF:

$$T \;:=\; \mathbf{0} \mid \sum_{i\in I} n(x).T_i \mid \sum_{i\in I} \overline{n}m_i.T_i \mid T \mid T' \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid\, !\pi.T,$$

where I is a finite nonempty indexing set and

$$\pi \;:=\; n(x) \mid \overline{n}m.$$

Here $n(x)$ is an input prefix and $\overline{n}m$ an output prefix. The *nil* process $\mathbf{0}$ cannot do anything in any environment. For each $i \leq n$, the component $n(x).T_i$ is a *summand* of the *input choice* term $\sum_{i\in I} n(x).T_i$, where the name variable x is *bound*. A name variable is *free* if it is not bound. Similarly the component $\overline{n}m_i.T_i$ is a summand of the *output choice* term $\sum_{i\in I} \overline{n}m_i.T_i$. Notice that input and output choices are syntactically simpler than the separated choices [Palamidessi 2003]. The term $T \mid T'$ is a concurrent *composition*. The restriction $(c)T$ is in *localization* form, where the name c is *local*. A name is *global* if it is not local. The following functions will be used.

—$gn(\_)$ returns the set of the global names.

—$ln(\_)$ returns the set of the local names.

—$n(\_)$ returns the set of the names.

—$fv(\_)$ returns the set of the free name variables.

—$bv(\_)$ returns the set of the bound name variables.

—$v(\_)$ returns the set of the name variables.

The guard $[p{=}q]$ is a *match* and $[p{\neq}q]$ a *mismatch*. The term $!\pi.T$ is a *guarded replication* and '!' a replication operator. The guarded replication is equivalent to the general replication of the form $!T$. The transformation from the general replication to the guarded replication makes use of an auxiliary function $(\_)^c$ defined on the replication free terms. The structural definition is as follows.

$$\begin{aligned}
(\mathbf{0})^c &\stackrel{\text{def}}{=} \mathbf{0},\\
(\pi.T)^c &\stackrel{\text{def}}{=} \pi.(\overline{c}c \mid T^c),\\
(T_1 \mid T_2)^c &\stackrel{\text{def}}{=} T_1^c \mid T_2^c,\\
((a)T)^c &\stackrel{\text{def}}{=} (a)T^c,\\
([p{=}q]T)^c &\stackrel{\text{def}}{=} [p{=}q]T^c,\\
([p{\neq}q]T)^c &\stackrel{\text{def}}{=} [p{\neq}q]T^c.
\end{aligned}$$

If neither c nor z appears in T, then we may define $(!T)^c$ by the process $(c)(\overline{c}c \mid !c(z).T^c)$. It is clear that there would be no loss of expressive power if guarded replication is further restrained to the form $!p(x).T$ or $!\overline{p}q.T$.

A *finite* π-term is one that does not contain any replication operator. A π-term is *open* if it contains free name variables; it is *closed* otherwise. A closed π-term is

also called a *π-process*. We write $\mathcal{P}$ for the set of the π-processes, ranged over by L, M, N, O, P, Q. Some derived prefix operators are defined as follows.

$$\begin{aligned}
\overline{n}(c).T &\stackrel{\text{def}}{=} (c)\overline{n}c.T,\\
n.T &\stackrel{\text{def}}{=} n(z).T \ \text{ for some } z \notin fv(T),\\
\overline{n}.T &\stackrel{\text{def}}{=} \overline{n}(c).T \ \text{ for some } c \notin gn(T),\\
\tau.T &\stackrel{\text{def}}{=} (c)(c.T \,|\, \overline{c}) \ \text{ for some } c \notin gn(T).
\end{aligned}$$

Furthermore we introduce the following polyadic prefixes:

$$\begin{aligned}
n(x_1,\ldots,x_n).T &\stackrel{\text{def}}{=} n(z).z(x_1).\cdots.z(x_n).T \ \text{ for some } z \notin fv(T),\\
\overline{n}\langle p_1,\ldots,p_n\rangle.T &\stackrel{\text{def}}{=} \overline{n}(c).\overline{c}p_1.\cdots.\overline{c}p_n.T \ \text{ for some } c \notin gn(T),
\end{aligned}$$

where $n > 1$. These two derived operator makes it clear how to simulate the polyadic π-calculus [Milner 1993b] in the (monadic) π-calculus.

Both bound name variables and local names are subject to α-conversion. Throughout the paper, it is assumed that α-conversion is applied whenever it is necessary to avoid confusion. This will be called *α-convention*. In for example the structural composition rule to be defined later, the side conditions are redundant in the presence of α-convention.

A *condition*, denoted by ϕ, φ, ψ, is a finite concatenation of matches and/or mismatches. The concatenation of zero match/mismatch is denoted by $\top$, and its negation is denoted by $\bot$. We identify syntactically $(\top)T$ with T and $(\bot)T$ with $\mathbf{0}$. If $\mathcal{F}$ is the finite name set $\{n_1,\ldots,n_i\}$, the notation $[p{\notin}\mathcal{F}]T$ stands for $[p{\neq}n_1]\ldots[p{\neq}n_i]T$.

A *renaming* is a partial injective map $\alpha : \mathcal{N} \rightharpoonup \mathcal{N}$ whose domain of definition $dom(\alpha)$ is finite. A *substitution* is a partial map $\sigma : \mathcal{N}_v \rightharpoonup \mathcal{N} \cup \mathcal{N}_v$ whose domain of definition $dom(\sigma)$ is finite. An *assignment* is a partial map $\rho : \mathcal{N}_v \rightharpoonup \mathcal{N}$ that associates a name to a name variable in the domain of ρ. It is convenient to extend the definition of an assignment ρ by declaring $\rho(a) = a$ for all $a \in \mathcal{N}$. Renaming, substitution and assignment are used in postfix. We often write $\{n_1/x_1,\ldots,n_i/x_i\}$ for a substitution, and similar notation is used for renaming. The notation $\rho[x \mapsto a]$ stands for the assignment that differs from ρ only in that $\rho[x \mapsto a]$ always maps x onto a whereas $\rho(x)$ might be different from a or undefined. Whenever we write $\rho(x)$ we always assume that ρ is defined on x.

An assignment ρ satisfies a condition ψ, denoted by $\rho \models \psi$, if $\rho(m) = \rho(n)$ for every $[m{=}n]$ in ψ, and $\rho(m){\neq}\rho(n)$ for every $[m{\neq}n]$ in ψ. We write $\rho \models \psi \Rightarrow \phi$ to mean that $\rho \models \phi$ whenever $\rho \models \psi$, and $\rho \models \psi \Leftrightarrow \phi$ if both $\rho \models \psi \Rightarrow \phi$ and $\rho \models \phi \Rightarrow \psi$. We say that ψ is *valid*, notation $\models \psi$ (or simply ψ), if $\rho \models \psi$ for every assignment ρ. A useful valid condition is the following one.

$$(x{\notin}\mathcal{F}) \vee \bigvee_{n\in\mathcal{F}} (x{=}n), \tag{8}$$

where $\mathcal{F}$ is a finite subset of $\mathcal{N} \cup \mathcal{N}_v$. Given a condition φ, we write $\varphi^{=}$ and $\varphi^{\neq}$ respectively for the condition $\bigwedge\{m{=}n \mid m,n \in n(\varphi) \cup v(\varphi) \text{ and } \varphi \Rightarrow m{=}n\}$ and the condition $\bigwedge\{m{\neq}n \mid m,n \in n(\varphi) \cup v(\varphi) \text{ and } \varphi \Rightarrow m{\neq}n\}$.

2.2 Semantics

The semantics is defined by a labeled transition system structurally generated by a set of rules. The set $\mathcal{L}$ of *labels* for π-terms, ranged over by ℓ, is

$$\{ab, \overline{a}b, \overline{a}(c) \mid a, b, c \in \mathcal{N}\}$$

where $ab, \overline{a}b, \overline{a}(c)$ denote respectively an input action, an output action and a bound output action. The set $\mathcal{L}^*$ of the finite strings of $\mathcal{L}$ is ranged over by ℓ^*. The empty string is denoted by ϵ. The set $\mathcal{A} = \mathcal{L} \cup \{\tau\}$ of actions is ranged over by λ and its decorated versions. The set $\mathcal{A}^*$ of the finite strings of $\mathcal{A}$ is ranged over by λ^*. With the help of the action set, we can define the operational semantics of π-calculus by the following labeled transition system.

Action

$$\frac{}{\sum_{i\in I} a(x).T_i \xrightarrow{ac} T_i\{c/x\}}\ i \in I \qquad \frac{}{\sum_{i\in I} \overline{a}c_i.T_i \xrightarrow{\overline{a}c_i} T_i}\ i \in I$$

Composition

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

Localization

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

Condition

$$\frac{T \xrightarrow{\lambda} T'}{[a{=}a]T \xrightarrow{\lambda} T'} \qquad \frac{T \xrightarrow{\lambda} T'}{[a{\neq}b]T \xrightarrow{\lambda} T'}$$

Replication

$$\frac{}{!\overline{a}b.T \xrightarrow{\overline{a}b} T \mid !\overline{a}b.T} \qquad \frac{}{!a(x).T \xrightarrow{ab} T\{b/x\} \mid !a(x).T}$$

The first composition rule takes care of the structural property. The second and the third define interactions. Their symmetric versions have been omitted. Particular attention should be paid to the action rules. An input action may receive a name from another term. It is not supposed to accept a name variable. This is because an output action is allowed to release a name, not a name variable. To go along with this semantics of interaction, the rule for mismatch operator is defined accordingly. Since distinct names are different constant names, the condition $a{\neq}b$ is equivalent to $\top$. The transition $[x{\neq}c]\overline{a}b.T \xrightarrow{\overline{a}b} T$ for instance is not admitted since the name variable x need be instantiated before the mismatch can be evaluated. One advantage of this semantics is the validity of the following lemma.

LEMMA 2.1. *The following statements are valid whenever* $S \xrightarrow{\lambda} T$.

(1) $S\alpha \xrightarrow{\lambda\alpha} T\alpha$ *for every renaming* α.

(2) $S\sigma \xrightarrow{\lambda} T\sigma$ *for every substitution* σ.

(3) $S\rho \xrightarrow{\lambda} T\rho$ *for every assignment* ρ.

The operational semantics of a process calculus draws a sharp line between internal actions (τ-actions) and external actions (non-τ-actions). From the point of view of interaction, the former is unobservable and the latter is observable. A *complete internal action sequence* of a process P is either an infinite τ-action sequence

$$P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_i \xrightarrow{\tau} \ldots$$

or a finite τ-action sequence $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n$, where P_n cannot perform any τ-action. A process P is *divergent* if it has an infinite internal action sequence; it is *terminating* otherwise.

In the π-calculus with the uniform treatment of names, *if_then_else_* command is defined with the help of the unguarded choice operator. A nice thing about the present semantics is that one may define *if_then_else_* command in the following manner.

$$\mathit{if}\, m{=}n\, \mathit{then}\, S\, \mathit{else}\, T \stackrel{\text{def}}{=} [m{=}n]S \,|\, [m{\neq}n]T.$$

The idea can be generalized. Suppose $\{\varphi_i\}_{1\leq i\leq n}$ is a finite collection of *disjoint conditions*, meaning that $\models \varphi_i \wedge \varphi_j \Leftrightarrow \bot$ for all $i, j \leq n$ such that $i \neq j$. The *conditional choice* $\sum_{i\in\{1,..,n\}} \varphi_i T_i$ is defined in the following fashion.

$$\sum_{1\leq i\leq n} \varphi_i T_i \stackrel{\text{def}}{=} \varphi_1 T_1 \,|\, \ldots \,|\, \varphi_n T_n. \tag{9}$$

In practice most choice operations are actually conditional choices [Cai and Fu 2010a]. Another form of choice is the so-called *internal choice*, defined as follows:

$$\sum_{1\leq i\leq n} \psi_i \tau.T_i \stackrel{\text{def}}{=} (c)(\psi_1 c.T_1 \,|\, \ldots \,|\, \psi_n c.T_n \,|\, \overline{c}), \tag{10}$$

where $\{\psi_i\}_{1\leq i\leq n}$ is a collection of conditions and c appears in none of $T_1, \ldots, T_n$.

In [Fu and Lu 2010] it is shown that the replication, the fixpoint operation and the parametric definition are equivalent in π-calculus, as long as all the choices are guarded. The fixpoint operator plays an indispensable role in proof systems for regular processes. The parametric definition is strictly more powerful in some variants of the π-calculus. Proof systems for regular π-processes will not be a major concern of this paper. In all the qualified name-passing calculi studied in the present paper the parametric definition is equivalent to the replicator. We shall therefore ignore both the fixpoint and the parametric definition in the rest of the paper.

A few more notations and terminologies need be fixed. Given an action λ, we may define $\overline{\lambda}$ as follows:

$$\overline{\lambda} \stackrel{\text{def}}{=} \begin{cases} \overline{a}b, & \text{if } \lambda = ab, \\ ab, & \text{if } \lambda = \overline{a}b, \\ ab, & \text{if } \lambda = \overline{a}(b), \\ \tau, & \text{if } \lambda = \tau. \end{cases}$$

The notation $\overline{\ell}$ should be understood accordingly. We shall write $\tilde{}$ for a finite sequence of something of same kind. For example a finite sequence of the names

$c_1, \ldots, c_n$ can be abbreviated to $\widetilde{c}$. Let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$. We write $\xrightarrow{\widehat{\lambda}}$ for $\equiv$ if $\lambda = \tau$ and for $\xrightarrow{\lambda}$ otherwise. The notation $\stackrel{\widehat{\lambda}}{\Longrightarrow}$ stands for $\Longrightarrow\xrightarrow{\widehat{\lambda}}\Longrightarrow$. If $\lambda^* = \lambda_1 \ldots \lambda_n$, we write $P \stackrel{\lambda^*}{\Longrightarrow}$ if $P \stackrel{\lambda_1}{\Longrightarrow} P_1 \ldots \stackrel{\lambda_n}{\Longrightarrow} P_n$ for some $P_1, \ldots, P_n$. If $P \stackrel{\lambda^*}{\Longrightarrow} P'$, we say that P' is a *descendant* of P. Notice that P is a descendant of itself.

In sequel a relation always means a binary relation on the processes of π-calculus or one of its variants. If $\mathcal{R}$ is a relation, $\mathcal{R}^{-1}$ is the reverse of $\mathcal{R}$ and $P\mathcal{R}Q$ for membership assertion. If $\mathcal{R}'$ is another relation, the composition $\mathcal{R};\mathcal{R}'$ is the relation $\{(P,Q) \mid \exists L.P\mathcal{R}L \wedge L\mathcal{R}'Q\}$.

2.3 Variants

A number of 'subcalculi' of π have been studied. These variants are obtained by omitting some operators. They can also be obtained by restricting the use of received names, or the use of continuation, or the forms of prefix etc.. In this section we give a brief summary of some of the variants. Our definitions of the variants are slightly different from the popular ones, since we attempt to give a systematic classification of the π-variants.

The guarded choice is a useful operator in encoding [Cai and Fu 2010b]. It is also a basic operator in proof systems. The independence of the choice operator from the other operators of π-calculus is established in for example [Palamidessi 2003; Fu and Lu 2010]. A lot of programming can be carried out in π-calculus using prefix terms rather than the guarded choice terms [Walker 1991; 1995]. We will write π^- for the subcalculus of π obtained by replacing the guarded choice terms by the prefix terms of the form $\pi.T$. In many aspects π^- is just as good as π. It is for example complete in the sense that one may embed the recursion theory [Rogers 1987] in π^-. See [Fu 2010b] for detail. The calculus π^- can be further slimmed down by removing the match and the mismatch operators. We shall call it the *minimal π-calculus* and shall denote it by π^M. It is conjectured that π^M is considerably weaker than π^-. The intuition behind the conjecture is that the if-command can not be faithfully encoded in π^M.

Some of the syntactical simplifications of π-calculus have been studied in literature. Here are some of them:

—Merro and Sangiorgi's local π-calculus forbids the use of a received name as an input channel [Merro 2000; Merro and Sangiorgi 2004]. The word 'local' refers to the fact that a receiving party may only use a received local name to call upon subroutines staying within the sending party. The calculus can be seen as a theoretical foundation of the concurrent and distributed languages *Pict* [Pierce and Turner 2000] and *Join* [Fournet and Gonthier 1996]. The local variants we introduce in this paper are obtained from π-calculus by restricting the use of the received names. A received name may be used as an output channel or an input channel. It may also appear in the object position. This suggests the following three variants. In π^L the grammar for prefix is

$$\pi \;:=\; a(x) \mid \overline{n}m.$$

In π^R the syntax of prefix is

$$\pi \;:=\; n(x) \mid \overline{a}m,$$

and in π^1 it is

$$\pi \;:=\; n(x) \mid \overline{n}c.$$

In π^1 only control information may be communicated, data information may not be passed around. A piece of control information may be sent once, it may not be resent.

Other well known variants are syntactical simplifications of π^-. Let's see a couple of them.

—The π-process $!a(x).T$ can be seen as a method that can be invoked by a process of the form $\overline{a}c.S$. This object oriented programming style is typical of the name-passing calculi. The object may invoke the method several times. It follows that the method must be perpetually available. The minimal modification of π that embodies this idea is π^O-calculus with following grammar:

$$T \;:=\; \mathbf{0} \mid \overline{n}m.T \mid T \,|\, T' \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid !n(x).T.$$

The superscript O refers to the fact that π^O-calculus has only proper output prefix; it also reminds of the object oriented programming style. The dual of π^O-calculus, π^I-calculus, is defined by the following grammar.

$$T \;:=\; \mathbf{0} \mid n(x).T \mid T \,|\, T' \mid (c)T \mid [p{=}q]T \mid [p{\neq}q]T \mid !\overline{n}m.T.$$

The relationship between π^I and π^O is intriguing. There is a straightforward encoding $[\![\_]\!]^\iota$ from the former to the latter defined as follows:

$$[\![n(x).T]\!]^\iota \;\stackrel{\text{def}}{=}\; \overline{n}(c).!c(x).[\![T]\!]^\iota,$$
$$[\![!\overline{n}m.T]\!]^\iota \;\stackrel{\text{def}}{=}\; !n(x).\overline{x}m.[\![T]\!]^\iota.$$

The soundness of this encoding is unknown. Both π^O and π^I have too weak control power to be considered as proper models. In π^O for example, a name can not be used to input two ordered pieces of information since a sending party would get confused. There is no way to define for example a polyadic prefix term like $a(x,y).T$. We shall not consider these two variants in the rest of the paper.

—Honda and Tokoro's object calculus [Honda and Tokoro 1991a; 1991b; Honda and Yoshida 1995] and Boudol's asynchronous π-calculus [Boudol 1992] are based on the idea that an output prefix with a continuation does not interact with any environment. Instead it evolves into a process representing the continuation and an atomic output process whose sole function is to send a name to an environment. This is captured by the following transitions.

$$\overline{a}c.T \;\stackrel{\tau}{\longrightarrow}\; \overline{a}c \,|\, T, \tag{11}$$
$$\overline{a}c \,|\, a(x).T \;\stackrel{\tau}{\longrightarrow}\; \mathbf{0} \,|\, T\{c/x\}. \tag{12}$$

(12) is absolutely necessary, whereas (11) can be avoided in view of the fact that $\overline{a}c.T$ must be equal to $\overline{a}c \,|\, T$. Moreover if we think asynchronously the replication,

match and mismatch operators should not apply to the processes of the form $\overline{n}m$. This explains the following grammar of π^A:

$$T := \mathbf{0} \mid \overline{n}m \mid \psi_i n_i(x).T \mid T \mid T' \mid (c)T \mid !n(x).T,$$

where ψ is a finite sequence of match/mismatch operations. The standard prefix operators can be mimicked in π^A in the following way [Boudol 1992]:

$$[\![p(x).S]\!] \stackrel{\text{def}}{=} p(u).(d)(\overline{u}d \mid d(x).[\![S]\!]), \tag{13}$$

$$[\![\overline{p}q.T]\!] \stackrel{\text{def}}{=} (c)(\overline{p}c \mid c(v).(\overline{v}q \mid [\![T]\!])). \tag{14}$$

This encoding of the output prefix is however not very robust from the observational point of view. In fact Cacciagrano et al. [2006] have proved that no encodings of the output prefix exist that preserve must equivalence. If divergence is not seen as an important issue, then there are interesting encodings into the asynchronous π-calculus as shown in [Nestmann and Pierce 1996] and in [Nestmann 2000]. The asynchronous calculi have a very different flavor from the synchronous calculi. We will not discuss π^A in the rest of the paper. But see [Fu 2010a] for an alternative approach to the asynchronous process calculi.

—A more distant relative is Sangiorgi's *private π-calculus* [Sangiorgi and Walker 2001b], initially called πI-calculus [Sangiorgi 1996b]. The grammar of π^P differs from that of π in the definition of prefix. In π^P it is given by the following BNF.

$$\pi := n(x) \mid \overline{n}(c).$$

Unlike all the above variants, the π^P-calculus has a different action set than π-calculus. There are no free output actions. One has typically that

$$a(x).S \mid \overline{a}(c).T \stackrel{\tau}{\longrightarrow} (c)(S\{c/x\} \mid T).$$

In π^P the name emitted by an output action is always local. It is worth remarking that this particular π^P-calculus is not equivalent to the version in which replications are abolished in favor of parametric definitions [Sangiorgi 1996b; Fu and Lu 2010]. It is interesting to compare π^P-calculus with π^1-calculus. In both models the messages communicated in run time are foreordained at compile time. The difference is that in π^P-calculus only local messages can be released, whereas in π^1-calculus global messages may also be transmitted. Despite of results established in [Boreale 1996], the variant π^P appears much less expressive than π^1. For one thing the match and mismatch operators in π^P are redundant. So it is more precise to say that π^P is a variant of π^M. Since π^P has a different action set than π^M, and it appears too weak, we shall not discuss it in this paper.

One may combine the restrictions introduced in the above variants to produce even more restricted 'subcalculi'. Some of these 'subcalculi' can be rejected right away. For example the variant in which the received names can only be used as data is equivalent to CCS, which is not Turing complete [Fu and Lu 2010]. Another counter example is the variant in which both the input capability and the output capability are perpetual. This calculus is utterly useless since no computations in this model ever terminate. It simply does not make sense to talk about Turing completeness for this particular calculus. Merro and Sangiorgi's local π-calculus is the

asynchronous version of π^L without the match/mismatch operators. In [Palamiddessi et al. 2006; Cacciagrano et al. 2008] the authors look at some asynchronous versions of π^I and π^O. Their work emphasizes more on the perpetual availability of the input/output actions.

A crucial question can be asked about each of the variants: which can be seen as an extension of the recursion theory? This is very much a question about the legitimacy of the variants. At the moment we only know that π, π^L, π^R are legal in this sense. See [Fu 2010b] for a formal account of this issue.

3. EQUALITY

The starting point of observational theory is to do with the definition of process equality. It was realized from the very beginning [Milner 1980; Sangiorgi 2009] that the equalities for processes ought to be observational since it is the effects that processes place on the environments that really matter. Different application platforms may demand different observing powers, giving rise to different process equalities. This interactive viewpoint [Milner 1993a] lies at the heart of the theory of equality. In a distributed environment for example, a process is subject to perpetual interventions from a potentially unbounded number of observers in an interleaving manner. Here the right equivalences are bisimulation equivalences. In a one shot testing scenario, equivalent processes are indistinguishable by any single tester in an exclusive fashion. It is nobody's business what the equivalent processes would turn into after a test. This is the testing equivalence. In this section we take a look at both the bisimulation approach and the testing approach.

3.1 Absolute Equality

A reasonable criterion for the equality on the *self evolving* objects is the famous bisimulation property of Milner [1989a] and Park [1981].

Definition 3.1. A relation $\mathcal{R}$ is a *weak bisimulation* if the following statements are valid.

(1) If $Q\mathcal{R}^{-1}P \xrightarrow{\tau} P'$ then $Q \Longrightarrow Q'\mathcal{R}^{-1}P'$ for some Q'.

(2) If $P\mathcal{R}Q \xrightarrow{\tau} Q'$ then $P \Longrightarrow P'\mathcal{R}Q'$ for some P'.

The bisimulation property captures the intuition that an equivalence for self evolving objects should be *maintainable* by the equivalent objects themselves when left alone. It is no good if two self evolving objects were equivalent one minute ago and will have to evolve into inequivalent objects one minute later. Bisimulation is about self evolving activities, not about interactive activities. What more can be said about self evolution? An evolution path may experience a series of state changes. At any particular time of the evolution, the process may turn into an equivalent state, and it may also move to an inequivalent state. Inequivalent states generally exert different effects on environments, while equivalent states always have the same interactive behaviors. Definition 3.1 can be criticized in that it overlooks the important difference between these two kinds of state transitions. The rectification of Definition 3.1 is achieved in van Glabbeek and Weijland [1989] by the branching bisimulation. The following particular formulation is proposed by [Baeten 1996].

Definition 3.2. A binary relation $\mathcal{R}$ is a *bisimulation* if it validates the following *bisimulation property*.

(1) If $Q\mathcal{R}^{-1}P \xrightarrow{\tau} P'$ then one of the following statements is valid.
   (a) $Q \Longrightarrow Q'$ for some Q' such that $Q'\mathcal{R}^{-1}P$ and $Q'\mathcal{R}^{-1}P'$.
   (b) $Q \Longrightarrow Q''\mathcal{R}^{-1}P$ for some Q'' such that $Q'' \xrightarrow{\tau} Q'\mathcal{R}^{-1}P'$ for some Q'.

(2) If $P\mathcal{R}Q \xrightarrow{\tau} Q'$ then one of the following statements is valid.
   (a) $P \Longrightarrow P'$ for some P' such that $P'\mathcal{R}Q$ and $P'\mathcal{R}Q'$.
   (b) $P \Longrightarrow P''\mathcal{R}Q$ for some P'' such that $P'' \xrightarrow{\tau} P'\mathcal{R}Q'$ for some P'.

Since we take the branching bisimulations as *the* bisimulations, we leave out the adjective.

A process not only evolves on its own, it also interacts with other processes. A plain criterion for the equalities of *interacting* objects is that an equality between two objects should be *maintainable* no matter how environments may interact with them. It is extremely important to get it right what an environment can and cannot do. In a distributed scenario, an environment may interact with a process placed in the regional network. An environment is not capable of grabbing a *running* process and reprogram it as it were. So it would not be appropriate to admit, say $a(x).(\_\{x/a\} \mid O)$, as an environment.

Definition 3.3. An *environment* is a process of the form $(\widetilde{c})(\_ \mid O)$ with the specified hole indicated by '$\_$'.

A prerequisite for two processes P, Q to be observationally equivalent is that no environment can tell them apart. But saying that the environment $(\widetilde{c})(\_ \mid O)$ cannot observe any difference between P and Q is nothing but saying that $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$ are observationally equivalent. Hence the next definition.

Definition 3.4. A relation $\mathcal{R}$ is *extensional* if the following statements are valid.

(1) If $L\mathcal{R}M$ and $P\mathcal{R}Q$ then $(L \mid P)\ \mathcal{R}\ (M \mid Q)$.
(2) If $P\mathcal{R}Q$ then $(c)P\ \mathcal{R}\ (c)Q$.

Process equivalences are *observational*. A process P is *observable*, written $P\Downarrow$, if $P \Longrightarrow\!\xrightarrow{\ell} P'$ for some P'. It is *unobservable*, written $P\not\Downarrow$, if it is not observable. Occasionally we write $P\Downarrow_\ell$ for $\exists P'.P \Longrightarrow\!\xrightarrow{\ell} P'$ and similarly $P\downarrow_\ell$ for $\exists P'.P \xrightarrow{\ell} P'$. It should be apparent that two equivalent processes must be both observable or both unobservable.

Definition 3.5. A relation $\mathcal{R}$ is *equipollent* if $P\Downarrow \Leftrightarrow Q\Downarrow$ whenever $P\mathcal{R}Q$.

Equipollence is the weakest condition that an observational equivalence has to satisfy. Now suppose P and Q are observationally inequivalent. If we ignore the issue of divergence, then there must exist some ℓ such that $P\Downarrow_\ell$ and for all ℓ' such that $Q\Downarrow_{\ell'}$, the actions ℓ and ℓ' exert different effects on some environment $(\widetilde{c})(\_ \mid O)$. As we mentioned just now, what this means is that $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$ are observationally inequivalent. So we may repeat the argument for $(\widetilde{c})(P \mid O)$ and $(\widetilde{c})(Q \mid O)$. But if the calculus is strong enough, say it is complete, then the inequivalence eventually boils down to the fact that one delivers a result at some

name and the other fails to do so at any name. Notice that the localization operator plays a crucial role in this argument. So for a large number of process calculi the equipollence condition is strong enough to characterize the observability.

The theory of process calculus has been criticized for not paying enough attention to the issue of divergence. This criticism is more or less to the point. If the theory of process calculus is meant to be an extension of the theory of computation, a divergent computation must be treated in a different way than a terminating computation. How should we formulate the requirement that process equivalences should be divergence respecting. A property that is not only divergence preserving but also consistent with the idea of bisimulation is codivergence.

Definition 3.6. A *relation* is *codivergent* if $P\mathcal{R}Q$ implies the following property.

(1) If $Q \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_n \xrightarrow{\tau} \cdots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and P' such that $P \stackrel{\tau}{\Longrightarrow} P' \; \mathcal{R} \; Q_k$.
(2) If $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_n \xrightarrow{\tau} \cdots$ is an infinite internal action sequence, then there must be some $k \geq 1$ and Q' such that $Q \stackrel{\tau}{\Longrightarrow} Q' \; \mathcal{R}^{-1} \; P_k$.

Using the codivergence condition, one may define an equivalence that is advocated in [Fu 2010b] as *the* equality for process calculi.

Definition 3.7. The *absolute equality* $=$ is the largest relation such that the following statements are valid.

(1) It is reflexive.
(2) It is equipollent, extensional, codivergent and bisimilar.

Alternatively we could define the absolute equality as the largest equipollent codivergent bisimulation that is closed under the environments.

The virtue of Definition 3.7 is that it is completely model independent, as long as we take the view that the composition operator and the localization operator are present in all process calculi. From the point of interaction, there cannot be any argument against equipollence and extensionality. From the point of view of computation, there is no question about codivergence and bisimulation. The only doubt one may raise is if Definition 3.7 is strong enough. We shall demonstrate in this paper that as far as π-calculus is concerned, the absolute equality is the most appropriate equivalence relation.

This is the right place to state an extremely useful technical lemma [Fu 1999], the Bisimulation Lemma. Although worded for $=$, Bisimulation Lemma, called X-property by De Nicola et al. [1990], is actually valid for all the observational equivalences of this paper except for the strong equality.

LEMMA 3.8. *If* $P \Longrightarrow= Q$ *and* $Q \Longrightarrow= P$, *then* $P = Q$.

A distinguished property about the absolute equality is stated in the next lemma. It is discovered by van Glabbeek and Weijland [1989] for the branching bisimilarity.

LEMMA 3.9. *If* $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n = P_0$ *then* $P_0 = P_1 = \ldots = P_n$.

A consequence of Lemma 3.9 is that if $P = Q \xrightarrow{\lambda} Q'$, then any simulation $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_n \xrightarrow{\lambda} P'$ of $Q \xrightarrow{\lambda} Q'$ by P must satisfy the property that $P = P_1 = \ldots = P_n$. This phenomenon motivates the following terminologies.

(1) $P \xrightarrow{\tau} P'$ is a *computation* if $P = P'$. In this case we write $P \to P'$.

(2) $P \xrightarrow{\tau} P'$ is a *change-of-state* if $P \neq P'$. In this case we write $P \xrightarrow{\iota} P'$.

A computation can be ignored. A change-of-state has to be properly simulated. The notation $\to^*$ stands for the reflexive and transitive closure of $\to$ and $\to^+$ for the transitive closure. We will write $P \not\to$ to indicate that $P \to P'$ for no P'.

Since the absolute equality applies not just to π and its variants but to all process calculi, the notation '=' could be confusing when more than one models are dealt with. To remove the ambiguity, we sometimes write $=^{\mathbb{M}}$ for the absolute equality of model $\mathbb{M}$. The same notations will be applied to other process equivalences. In this paper, we write $\mathbb{V}$ an arbitrary π-variant.

3.1.1 *External Bisimilarity.* The external bisimilarity requires that all observable actions are explicitly simulated. This is commonly referred to as branching bisimilarity if the codivergence is dropped [van Glabbeek and Weijland 1989].

Definition 3.10. A codivergent bisimulation of $\mathbb{V}$ is a $\mathbb{V}$-*bisimulation* if the following statements are valid for every $\ell \in \mathcal{L}$.

(1) If $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'\mathcal{R}^{-1}P'$ and $P\mathcal{R}Q''$ for some Q', Q''.

(2) If $P\mathcal{R}Q \xrightarrow{\ell} Q'$ then $P \Longrightarrow P'' \xrightarrow{\ell} P'\mathcal{R}Q'$ and $P''\mathcal{R}Q$ for some P', P''.

The $\mathbb{V}$-*bisimilarity* $\simeq_{\mathbb{V}}$ is the largest $\mathbb{V}$-bisimulation.

$\mathbb{V}$-bisimilarity is a more effective counterpart of the absolute equality $=_{\mathbb{V}}$. The latter provides the intuition, while the former offers a tool for establishing equational properties. The proof of following fact is a standard textbook application of the bisimulation argument.

Fact 3.11. *The equivalence $\simeq_{\mathbb{V}}$ is closed under input choice, output choice, composition, localization, match, mismatch, and guarded replication.*

An immediate consequence of Fact 3.11 is the inclusion described in the following lemma.

Lemma 3.12. $\simeq_{\mathbb{V}} \subseteq =_{\mathbb{V}}$.

If the processes of a model has strong enough observing power, then the absolute equality is as strong as the external bisimilarity. This is the case for the π-calculus. The proof of the following theorem resembles a proof in [Fu 2005].

Theorem 3.13. *The π-bisimilarity $\simeq_\pi$ coincides with the absolute equality $=_\pi$.*

Proof. In view of Lemma 3.12, we only have to prove $= \subseteq \simeq$. Let $\mathcal{R}$ be the relation

$$\left\{ (P,Q) \;\middle|\; \begin{array}{l} (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P) = \\ (c_1,\ldots,c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q), \\ \{a_1,\ldots,a_n\} \cap gn(P \mid Q) = \emptyset,\ n \geq 0 \end{array} \right\}.$$

To appreciate the relation notice that $(c'c'')(\overline{a'}c' \mid \overline{a''}c'' \mid P) \neq (c)(\overline{a'}c \mid \overline{a''}c \mid Q)$ for all P, Q such that $\{a', a''\} \cap gn(P \mid Q) = \emptyset$. We prove that $\mathcal{R}$ is an external bisimulation

up to $\sim$, where $\sim$ is defined in Section 3.1.2. Now suppose $A = B$ where

$$A \stackrel{\text{def}}{=} (c_1, \ldots, c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P),$$
$$B \stackrel{\text{def}}{=} (c_1, \ldots, c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q),$$

such that $\{a_1, \ldots, a_n\} \cap gn(P \mid Q) = \emptyset$ and $n \geq 0$. Consider an action $A \xrightarrow{\ell} A'$ of A. There are five situations in which the action may be induced.

(1) $\ell = ab$. Let C be $\overline{a}b + \overline{a}c.d$ for some fresh names c, d. By equipollence and extensionality, $A \mid C \xrightarrow{\tau} A' \mid \mathbf{0}$ must be matched up by $B \mid C \Longrightarrow B'' \mid C \xrightarrow{\tau} B' \mid \mathbf{0}$ for some B', B''. Since $B'' \mid C \xrightarrow{\tau} B' \mid \mathbf{0}$ must be a change of state, it must be the case that $A \mid C = B'' \mid C$. It follows easily from Bisimulation Lemma that $B \rightarrow^* B'' \xrightarrow{ab} B' = A'$. Clearly

$$A' \equiv (c_1, \ldots, c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid P'),$$
$$B' \equiv (c_1, \ldots, c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q'),$$
$$B'' \equiv (c_1, \ldots, c_n)(\overline{a_1}c_1 \mid \ldots \mid \overline{a_n}c_n \mid Q''),$$

for some P', Q', Q''. It follows from $B \rightarrow^* B'' \xrightarrow{ab} B'$ that $Q \Longrightarrow Q'' \xrightarrow{ab} Q'$. Moreover $P \mathcal{R} Q''$ and $P' \mathcal{R} Q'$ by definition.

(2) $\ell = \overline{a}b$ for some $b \notin \{c_1, \ldots, c_n\}$. Let c, d be fresh and let D be defined as follows:

$$D \stackrel{\text{def}}{=} a(x).([x{=}b]c \mid [x{=}c]d).$$

Now $A \mid D \xrightarrow{\tau} A' \mid [b{=}b]c \mid [b{=}c]d$ must be simulated by

$$B \mid D \Longrightarrow B'' \mid D \xrightarrow{\tau} B' \mid [b{=}b]c \mid [b{=}c]d.$$

The rest of the argument is the same as in the previous case.

(3) $\ell = \overline{a}(c)$ and $c \notin \{c_1, \ldots, c_n\}$. Let E be the following process

$$a(x).(\overline{a_{n+1}}x \mid [x{=}d]e \mid [x \notin gn(P \mid Q)]f),$$

where d, e, f, a_{n+1} are fresh. The rest of the argument is similar.

(4) $\ell = \overline{a}(c_i)$ for some $i \in \{1, \ldots, n\}$. Let F be the process

$$a(x).([x{=}d]e \mid a_i(y).(\overline{a_i'}y \mid [x{=}y]f)),$$

where d, e, f, a_i' are fresh. The rest of the argument is again similar.

(5) $\ell = \overline{a_i}(c_i)$ for some $i \in \{1, \ldots, n\}$. Let G be the process $a_i(x).\overline{a_i'}x$, where a_i' is fresh. This is a simpler case.

Additionally we need to consider the external actions of P, Q that communicate through one of the names $c_1, \ldots, c_n$. There are seven cases. In each case we are content with defining an environment that forces external bisimulation.

(1) $P \xrightarrow{c_i b} P'$. Let H be the following process

$$a_i(z).((\overline{z}b + \overline{z}c.\overline{d}e) \mid \overline{a_i'}z)$$

for some fresh c, d, e, a_i'.

π	π^-	π^M	π^L	π^R	π^1
$\surd$	?	?	?	?	?

Fig. 1. Coincidence of $\simeq_{\mathbb{V}}$ and $=_{\mathbb{V}}$.

(2) $P \xrightarrow{c_i c_j} P'$ such that $i \neq j$. Let I be the following process

$$a_i(z).a_j(y).((\overline{z}y + \overline{z}c.\overline{d}e) \mid \overline{a'_j}y \mid \overline{a'_i}z)$$

for some fresh c, d, e, a'_i, a'_j.

(3) $P \xrightarrow{c_i c_i} P'$. Let J be the following process

$$a_i(z).((\overline{z}z + \overline{z}c.\overline{d}e) \mid \overline{a'_i}z)$$

for some fresh c, d, e, a'_i.

(4) $P \xrightarrow{\overline{c_i}b} P'$. Let K be the following process

$$a_i(z).(z(x).([x{=}b]c \mid [x{=}c]d) \mid \overline{a'_i}z)$$

for some fresh c, d, a'_i.

(5) $P \xrightarrow{\overline{c_i}(c)} P'$. Let L be the following process

$$a_i(z).(z(x).(\overline{a_{n+1}}x \mid [x{=}d]e \mid [x \notin gn(P \mid Q)]f) \mid \overline{a'_i}z),$$

where d, e, f, a_{n+1}, a'_i are fresh.

(6) $P \xrightarrow{\overline{c_i}c_j} P'$ such that $i \neq j$. Let M be the following process

$$a_i(z).(z(x).([x{=}d]e \mid a_j(y).[x{=}y]\overline{a'_j}y) \mid \overline{a'_i}z)$$

for some fresh d, e, a'_i, a'_j.

(7) $P \xrightarrow{\overline{c_i}c_i} P'$. Let N be the following process

$$a_i(z).(z(x).([x{=}d]e \mid [x{\neq}z]d) \mid \overline{a'_i}z)$$

for some fresh d, e, a'_i.

We are done. □

The reader must have noticed that the output choice is crucial in the above proof, but the input choice is unnecessary. At the moment we do not see how the theorem can be established without using the output choice.

The external characterization of the absolute equality for a particular model is an important issue. It is definitely a technically interesting but probably tricky exercise to remove the question marks in Fig. 1.

Problem 3.14. What are the answers to the questions raised in Fig. 1?

There is a standard way to extend the absolute equality from processes to terms. The following definition is well-known.

Definition 3.15. Let $\asymp$ be a process equivalence. Then $S \asymp T$ if $S\rho \asymp T\rho$ for every assignment ρ such that $fv(S \mid T) \subseteq dom(\rho)$.

The equivalence $=$ on π-terms satisfies two crucial equalities. One is

$$T = [x{=}y]T \mid [x{\neq}y]T. \tag{15}$$

The other is

$$[x{\neq}y]\lambda.T = [x{\neq}y]\lambda.[x{\neq}y]T. \tag{16}$$

Both equalities explain the role of the name variables. It is the viewpoint of this paper that all process equivalences should validate both (15) and (16).

3.1.2 *Strong Equality and Weak Equality.* The only way to refine the absolute equality in a model independent way is to strengthen the bisimulation property. Among all the refinements of Definition 3.2, the most well known one is Milner and Park's strong bisimulation [Park 1981; Milner 1989a].

Definition 3.16. A symmetric relation $\mathcal{R}$ is a *strong bisimulation* if $P \xrightarrow{\tau} P'\mathcal{R}Q'$ for some P' whenever $P\mathcal{R}Q \xrightarrow{\tau} Q'$.

It is clear that a strong bisimulation is automatically codivergent. Hence the next definition.

Definition 3.17. The *strong equality* $=^s$ is the largest reflexive, equipollent, extensional, strong bisimulation.

The strong equality is the same as the strong bisimilarity of Milner [1989a].

Definition 3.18. A strong bisimulation is an *external strong bisimulation* if the following statements are valid for every $\ell \in \mathcal{L}$.

(1) If $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \xrightarrow{\ell} Q'\mathcal{R}^{-1}P'$ for some Q'.

(2) If $P\mathcal{R}Q \xrightarrow{\ell} Q'$ then $P \xrightarrow{\ell} P'\mathcal{R}Q'$ for some P'.

The *external strong bisimilarity* $\sim$ is the largest external strong bisimulation.

One finds a very useful application of the strong equality in the 'bisimulation up to $\sim$' technique [Sangiorgi and Milner 1992]. The strong equality equates all the structurally equivalent processes. Some well known equalities are stated below.

Fact 3.19. *The following equalities are valid.*

(1) $P \mid \mathbf{0} \sim P$; $P \mid Q \sim Q \mid P$; $P \mid (Q \mid R) \sim (P \mid Q) \mid R$.
(2) $(c)\mathbf{0} \sim \mathbf{0}$; $(c)(d)P \sim (d)(c)P$.
(3) $(c)(P \mid Q) \sim P \mid (c)Q$ *if* $c \notin gn(P)$.
(4) $[a{=}a]P \sim [a{\neq}b]P \sim P$; $[a{\neq}a]P \sim [a{=}b]P \sim \mathbf{0}$.
(5) $!\pi.P \sim \pi.P \mid !\pi.P$.

In the other direction we may weaken the absolute equality by relaxing the bisimulation property *a la* Definition 3.1.

Definition 3.20. The *weak equality* $=^w$ is the largest reflexive, equipollent, extensional, codivergent, weak bisimulation.

The external characterization of the weak equality is Milner's weak bisimilarity [Milner 1989a] enhanced with the codivergence condition.

Definition 3.21. A codivergent bisimulation is an *external weak bisimulation* if the following statements are valid for every $\ell \in \mathcal{L}$.

(1) If $Q\mathcal{R}^{-1}P \xrightarrow{\ell} P'$ then $Q \stackrel{\ell}{\Longrightarrow} Q'\mathcal{R}^{-1}P'$ for some Q'.

(2) If $P\mathcal{R}Q \xrightarrow{\ell} Q'$ then $P \stackrel{\ell}{\Longrightarrow} P'\mathcal{R}Q'$ for some P'.

The *external weak bisimilarity* $\simeq$ is the largest external weak bisimulation.

The proof of Theorem 3.13 can be extended to a proof of the following coincidence result.

PROPOSITION 3.22. *The equivalences* $=^w, \simeq$ *coincide in* π^M, π^- *and* π.

The relationships between the strong equality, the absolute equality and the weak equality are stated in the next proposition.

PROPOSITION 3.23. *The inclusions* $=^s \subsetneq = \subsetneq =^w$ *are strict.*

PROOF. It is well known that $a.(\tau.b + \tau.c) =^w a.(\tau.b + \tau.c) + a.c$ holds but $a.(\tau.b + \tau.c) = a.(\tau.b + \tau.c) + a.c$ is not valid. □

3.1.3 *Remark.* In the theory of bisimulation semantics, the three major contributions are the bisimulation of Milner [1989a] and Park [1981], the branching bisimulation of van Glabbeek and Weijland [1989], and the barbed bisimulation of Milner and Sangiorgi [1992]. Despite of its nice properties [De Nicola et al. 1990; van Glabbeek 1994; De Nicola and Vaandrager 1995; Baier and Hermanns 1997], the branching bisimulation is not as widely appreciated as it deserves. The absolute equality is introduced in [Fu 2010b] with three points of view. The first is that bisimulation is a defining property for the internal actions, it is a derived property for the external actions. This is very much the motivation for the barbed bisimulation. The second is that not all internal actions are computations. Only those that evolve to equivalent states are computations. This is in our opinion the philosophy of the branching bisimulation. The third is that bisimilar objects should also bisimulate each other's infinite computations. Codivergence and bisimulation complements to each other. The point is that computations are ignorable locally but not globally. The notion of codivergence is formulated by Priese [1978]. The correlation of the codivergence property to the bisimulation property is emphasized independently by Fu [2010b] and by van Glabbeek et al. [2009].

Several other bisimulation based equivalences have been proposed for π-calculus. First of all let's take a look at the first equivalence proposed for π-calculus [Milner et al. 1992], the early equivalence. For the sake of illustration let's denote by $\dot{\approx}$ the largest equipollent codivergent weak bisimulation. We say that P, Q are early equivalent, notation $P \approx_e Q$, if $(\widetilde{c})(P \,|\, O)\dot{\approx}(\widetilde{c})(Q \,|\, O)$ for every environment $(\widetilde{c})(\_ \,|\, O)$. Despite the result in [Sangiorgi 1992], it is still an open problem if $\approx_e$ coincides with the weak bisimilarity $\simeq$. But the issue is not worth any serious effort in view of the following contradiction one finds in the definition of $\approx_e$.

—The bisimulation property together with the equipollence property assume that environments are dynamically changing.

—The closure under environments in the beginning of the observation says the opposite.

The so-called late equivalence [Milner et al. 1992] has also been studied in literature, especially when one constructs proof systems. If we think of it, the late equivalence is not really an observational equivalence. As long as the one step interactions are atomic actions, there is no way to tell apart by a third party the difference between $a(x).P + a(x).Q$ and $a(x).P + a(x).Q + a(x).([x{=}d]P + [x{\neq}d]Q)$. Even if the late equivalence is useful when names are treated uniformly, it is a dubious equivalence when name dichotomy is applied.

Another well known equivalence for π-calculus is the open bisimilarity [Sangiorgi 1996c]. The open approach was meant to deal with the open π-terms head-on. A closely related equivalence is the quasi open bisimilarity [Sangiorgi and Walker 2001a; Fu 2005]. An open bisimulation is a distinction indexed set of relations. In the presence of the name dichotomy, an indexed family of relations is no longer necessary. The following is the standard definition of the open bisimilarity iterated in the present framework.

> A codivergent bisimulation $\mathcal{R}$ on $\mathcal{T}$ is an *open bisimulation* if the following statements are valid.
> (1) If $S\mathcal{R}T$ then $S\sigma\ \mathcal{R}\ T\sigma$ for every substitution σ.
> (2) If $T\mathcal{R}^{-1}S \xrightarrow{\ell} S'$ then $T \stackrel{\ell}{\Longrightarrow} T'\mathcal{R}^{-1}S'$ for some T'.
> (3) If $S\mathcal{R}T \xrightarrow{\ell} T'$ then $S \stackrel{\ell}{\Longrightarrow} S'\mathcal{R}T'$ for some S'.
> The *open bisimilarity* $\approx_o$ is the largest open bisimulation.

The relation $\approx_o$ should definitely be given up. For one thing it fails (15). The term $\overline{a}a$ for example cannot be simulated by $[x{=}y]\overline{a}a \,|\, [x{\neq}y]\overline{a}a$. If one tries to correct the above definition, one soon realizes that what one gets is the weak equality.

3.1.4 *Algebraic Property.* This section takes a closer look at the absolute equality on π-terms. The goal is to reduce the proof of an equality between π-terms to the proof of an equality between π-processes. To describe the reductions, we need to fix some terminologies and notations. We will write $\subseteq_f$ for the finite subset relationship. If $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, then $\mathcal{F}_c$ stands for $\mathcal{F} \cap \mathcal{N}$ and $\mathcal{F}_v$ for $\mathcal{F} \cap \mathcal{N}_v$. Given such a finite set $\mathcal{F}$, a condition ψ may completely characterize the relationships between the elements of $\mathcal{F}$. This intuition is formalized in the next definition.

Definition 3.24. Suppose $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$. We say that a satisfiable condition ψ is *complete on* $\mathcal{F}$, if $\mathcal{F}_c = n(\psi)$, $\mathcal{F}_v = v(\psi)$ and for every $x \in \mathcal{F}_v$ and every $q \in \mathcal{F}$ it holds that either $\psi \Rightarrow x = q$ or $\psi \Rightarrow x \neq q$. A condition ψ is *complete* if it is complete on $n(\psi) \cup v(\psi)$.

The defining property of Definition 3.24 immediately implies the following lemma.

Lemma 3.25. *Suppose φ is complete on $\mathcal{F}$ and $n(\psi) \cup v(\psi) \subseteq \mathcal{F}$. Then either $\varphi \Rightarrow \psi$ or $\varphi\psi \Rightarrow \bot$.*

It follows from Lemma 3.25 that if both φ, ψ are complete on $\mathcal{F}$ then either $\varphi \Leftrightarrow \psi$ or $\varphi\psi \Leftrightarrow \bot$. This fact indicates that we may apply complete conditions to carry out case analysis when reasoning about term equality. A case analysis is based on a complete disjoint partition.

Definition 3.26. Suppose $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$. A finite set $\{\varphi_i\}_{i\in I}$ is a *complete disjoint partition* of $\mathcal{F}$ if the following conditions are met.

(1) φ_i is complete on $\mathcal{F}$ for every $i \in I$.
(2) $\bigvee_{i\in I} \varphi_i \Leftrightarrow \top$.
(3) $\varphi_i \wedge \varphi_j \Rightarrow \bot$ for all i, j such that $i \neq j$.

Given a set $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, there is always a complete disjoint partition on $\mathcal{F}$. So one can always apply a partition to a π-term.

LEMMA 3.27. *$T \sim \sum_{i\in I} \varphi_i T$, where $\{\varphi_i\}_{i\in I}$ is a complete disjoint partition of $gn(T) \cup fv(T)$.*

Now suppose φ is complete on $gn(S \,|\, T) \cup fv(S \,|\, T)$. Usually it is much easier to prove $\varphi S \simeq \varphi T$ than to prove $S \simeq T$. So there is a strong interest in the result stated next.

LEMMA 3.28. *Let $\{\varphi_i\}_{i\in I}$ be a complete disjoint partition of $gn(S \,|\, T) \cup fv(S \,|\, T)$. Then $S \simeq T$ if and only if $\varphi_i S \simeq \varphi_i T$ for every $i \in I$.*

How can we reduce the equality proof of $\varphi S \simeq \varphi T$ further if φ is complete on $gn(S \,|\, T) \cup fv(S \,|\, T)$? Consider $[x{=}y][z{=}a][z'{=}b]S \simeq [x{=}y][z{=}a][z'{=}b]T$. This equality is the same as

$$[x{=}y][z{=}a][z'{=}b]S\{y/y, y/x, a/z, b/z'\} \simeq [x{=}y][z{=}a][z'{=}b]T\{y/y, y/x, a/z, b/z'\},$$

which can be reduced to

$$S\{y/y, y/x, a/z, b/z'\} \simeq T\{y/y, y/x, a/z, b/z'\}.$$

The substitution $\{y/y, y/x, a/z, b/z'\}$ has the property that it agrees with the condition $x{=}y \wedge z{=}a \wedge z'{=}b$ and its domain set and range set are subsets of $\{x, y, z, a, z', b\}$. Substitutions of this kind are very useful when rewriting π-terms.

Definition 3.29. An assignment ρ *agrees with* ψ, and ψ agrees with ρ, if the following statements are valid.

(1) $v(\psi) \subseteq dom(\rho)$.
(2) For each $x \in v(\psi)$ and each $a \in n(\psi)$, $\psi \Rightarrow x = a$ if and only if $\rho(x) = a$.
(3) For all $x, y \in v(\psi)$, $\psi \Rightarrow x = y$ if and only if $\rho(x) = \rho(y)$.

A substitution σ is *induced* by ψ if the following statements are valid.

(1) $n(\sigma) \subseteq n(\psi)$, $v(\sigma) \subseteq v(\psi)$ and $dom(\sigma) = v(\psi)$.
(2) $\psi \Rightarrow x = a$ if and only if $\sigma(x) = a$.
(3) $\sigma(x) = \sigma(y)$ if and only if $\psi \Rightarrow x = y$.
(4) For each $x \in v(\psi)$, $\sigma(x)$ is some name c if and only if $\psi \Rightarrow x = c$.

There could be many assignments that agree with ψ and many substitutions that are induced by ψ. We shall write ρ_ψ for some assignment that agrees with ψ and σ_ψ for some substitution that is induced by ψ. It should be clear that if ψ is complete, then $\sigma_\psi(x) \neq \sigma_\psi(y)$ if and only if $\psi \Rightarrow x \neq y$ if and only if $\rho_\psi(x) \neq \rho_\psi(y)$. The validity of the next lemma is obvious.

LEMMA 3.30. $\varphi T \sim \varphi T \sigma_\varphi$.

So we have reduced the proof of $S \simeq T$ to the proof of $\varphi S\sigma_\varphi \simeq \varphi T\sigma_\varphi$ for some φ complete on $gn(S\,|\,T) \cup fv(S\,|\,T)$. Now the match conditions appeared in φ are inessential. We may apply the following lemma to remove all the match conditions.

LEMMA 3.31. *The following statements are valid.*

(1) *If* $n \notin gn(S\,|\,T) \cup fv(S\,|\,T)$, *then* $[x{\neq}n]S \simeq [x{\neq}n]T$ *if and only if* $S \simeq T$.

(2) *If* $x \notin fv(S\,|\,T)$, *then* $[x{=}n]S \simeq [x{=}n]T$ *if and only if* $S \simeq T$ *if and only if* $[x{\neq}n]S \simeq [x{\neq}n]T$.

After Lemma 3.31 we may assume that an equality to be proved is of the form $\delta_{\mathcal{F}}S \simeq \delta_{\mathcal{F}}T$, where $\mathcal{F} = gn(S\,|\,T) \cup fv(S\,|\,T)$ and $\delta_{\mathcal{F}}$ is the conjunction

$$\bigwedge\{x \neq n \mid x \in \mathcal{F}_v,\ n \in \mathcal{F} \text{ and } x, n \text{ are distinct}\}.$$

In the last step of the reduction we apply the following result.

LEMMA 3.32. $\delta_{\mathcal{F}}S \simeq \delta_{\mathcal{F}}T$ *if and only if* $S\rho_{\delta_{\mathcal{F}}} \simeq T\rho_{\delta_{\mathcal{F}}}$, *where* $\mathcal{F} = gn(S\,|\,T) \cup fv(S\,|\,T)$.

It should be remarked that $\delta_{\mathcal{F}}S \simeq \delta_{\mathcal{F}}T$ is an equality between two open π-terms, whereas $S\rho_{\delta_{\mathcal{F}}} \simeq T\rho_{\delta_{\mathcal{F}}}$ is an equality between two π-processes. The proof of Lemma 3.32 is immediately from the following fact.

LEMMA 3.33. *If* $P \simeq Q$ *then* $P\alpha \simeq Q\alpha$ *for every renaming* α.

We conclude that all equality proofs for π-terms can be transformed to equality proofs for π-processes.

Lemma 3.28, Lemma 3.31, Lemma 3.32 and Lemma 3.33 are also valid for $\sim$.

It is difficult to attribute the above lemmas to the original authors. It would be fair to say that most early researchers on π-calculus were aware of the properties described in these lemmas. See for example the work of Lin [2003] and Parrow and Sangiorgi [1995].

3.2 Testing Equivalence

The testing equivalence of De Nicola and Hennessy [De Nicola and Hennessy 1984; Hennessy 1988] is probably the best known nonbisimulation equivalence for processes. In the testing approach, one only cares about the *result* of observation, not the *course* of observation. To define a testing equivalence, one needs to explain what the observers are, how the observers carry out tests, and what counts as the result of a test. In the spirit of observation theory, there is not much room for variation. The observers are the environments. A test by the observer $(\widetilde{c})(\_\,|\,O)$ on a process P is a *complete* internal action sequence of $(\widetilde{c})(P\,|\,O)$. The result of a test can be either successful or unsuccessful. From the point of view of observation, the success of a test can only be indicated by an observable action. The only variations that may arise are concerned with the way the successes are reported.

This subsection serves two purposes. One is to demonstrate that the testing theory for π-calculus is just as simple as that for CCS. The second is to give a model independent characterization of the testing equivalence.

3.2.1 *May Equivalence and Must Equivalence.* The testing machinery of De Nicola and Hennessy [1984] can be summarized as follows.

—Success is indicated by the special action ω.

—In the presence of ω, the observers can be simplified to the environments of the form $\_ \mid O$. There is no need for the localization operator. An observer O is obtained from some process by replacing some occurrences of $\mathbf{0}$ by ω.

—A test of P by O is a complete internal action sequence of $P \mid O$.

—A test of P by O is *DH-successful* if at some state of the test, the ω action is *immediately* fireable. It is unsuccessful otherwise.

—A binary relation $\mathcal{R}$ on processes satisfies *may predicate* if $P\mathcal{R}Q$ implies that, for every observer O, some test of $P \mid O$ is DH-successful if and only if some test of $Q \mid O$ is DH-successful.

—A binary relation $\mathcal{R}$ on processes satisfies *must predicate* if $P\mathcal{R}Q$ implies that, for every observer O, all tests of $P \mid O$ are DH-successful if and only if all tests of $Q \mid O$ are DH-successful.

Using the above terminologies, we may define the may equivalence $\approx_{may}$, respectively the must equivalence $\approx_{must}$, as the largest relation on the π-processes that satisfies the may predicate, respectively the must predicate. Let's see some illustrating examples.

—$A \mid !\tau \approx_{may} A$. The may equivalence ignores divergence.

—$A \mid !\tau \not\approx_{must} A$ if A is terminating. The must equivalence is discriminating against divergent processes.

—$A \mid !\tau \approx_{must} B \mid !\tau$ even if $A \not\approx_{must} B$. It is often said that divergence is catastrophic for must equivalence.

It is obvious from these examples that $\approx_{must}$ is incompatible to both $\approx_{may}$ and Milner's weak bisimilarity, which is really undesirable. The may equivalence behaves better. It is well known that $\approx_{may}$ is nothing but the so called trace equivalence. A proof of this coincidence, Lemma 3.34, can be found in [De Nicola and Hennessy 1984].

LEMMA 3.34. *$P \approx_{may} Q$ if and only if $(P \stackrel{\ell^*}{\Longrightarrow}) \Leftrightarrow (Q \stackrel{\ell^*}{\Longrightarrow})$ for all $\ell^* \in \mathcal{L}^*$.*

All observational equivalences should subsume the trace equivalence [van Glabbeek 1990]. For the must equivalence this is true only for a set of hereditarily terminating processes. The following definition and the next lemma are from [De Nicola and Hennessy 1984].

Definition 3.35. A process is *strongly convergent* if all the descendants of the process are terminating.

LEMMA 3.36. *$P \approx_{must} Q$ implies $P \approx_{may} Q$ if P, Q are strongly convergent.*

PROOF. We start by defining the *trace observer* generated by some $\ell^* \in \mathcal{L}^*$ and some finite subset $\mathcal{F}$ of $\mathcal{N}$. The structural definition is given in Figu. 2. Notice that the input choice and the output choice are necessary to define the trace observer.

$$
\begin{array}{rcl}
O^{\mathcal{F}}_{\epsilon} & \stackrel{\text{def}}{=} & \tau.\mathbf{0} \\
O^{\mathcal{F}}_{ab,\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + \overline{a}b.O^{\mathcal{F}}_{\ell^*} \\
O^{\mathcal{F}}_{\overline{a}b,\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + a(z).(\tau.\omega + [z{=}b]\tau.O^{\mathcal{F}\cup\{b\}}_{\ell^*}) \\
O^{\mathcal{F}}_{\overline{a}(b),\ell^*} & \stackrel{\text{def}}{=} & \tau.\omega + a(z).(\tau.\omega + [z{\notin}\mathcal{F}]\tau.O^{\mathcal{F}\cup\{z\}}_{\ell^*})
\end{array}
$$

Fig. 2. Trace Observer.

Without loss of generality assume that $P \not\approx_{may} Q$. Then there must exist some nonempty $\ell_1^* \in \mathcal{L}^*$ such that $P \stackrel{\ell_1^*}{\Longrightarrow}$ and $\neg(Q \stackrel{\ell_1^*}{\Longrightarrow})$. Let $\mathcal{F}'$ be $gn(P \mid Q)$. Obviously $P \mid O^{\mathcal{F}'}_{\ell_1^*}$ has an unsuccessful test. Since Q is strongly convergent, it does not induce any infinite computation. It follows that all the tests of $Q \mid O^{\mathcal{F}'}_{\ell_1^*}$ are successful. □

3.2.2 *Remark.* The proceeding section has pinpointed the problems with the must equivalence. Let's summarize.

I. Something must be wrong with the fact that Milner's weak bisimilarity is not a sub-relation of the testing equivalence. Both the weak bisimilarity and the testing equivalence are to blame since neither deals with divergence properly. Moreover there is no excuse for $\approx_{must} \not\subseteq \approx_{may}$.

II. It is a little discomfort to introduce a special symbol ω to indicate success. The action ω introduces an asymmetry between testers and testees. In De Nicola and Hennessy's approach, a testing equivalence is always defined for a particular process calculus. The definition may vary from one calculus to another. If the testing equivalence is really a fundamental equivalence for processes, there ought to be a model independent definition of the equivalence that applies to all calculi. The model independent characterization should be able to provide some canonicity for the testing approach.

In literature there have been several attempts to modify the definition of must equivalence in order to resolve issues I and II, see for example [Phillips 1987; Brinksma et al. 1995; Natarajan and Cleaveland 1995; Boreale et al. 1999; 2001]. Let's review some of the proposals.

I. Brinksma, Rensink and Volger's *should equivalence* [Brinksma et al. 1995] and Natarajan and Cleaveland's *fair testing equivalence* [Natarajan and Cleaveland 1995] are two modifications proposed to address issue I. These two variants are essentially defined over the same success predicate. A test of P by O is *FS-successful* if $Q \stackrel{\omega}{\Longrightarrow}$ whenever $P \mid O \Longrightarrow Q$. A binary relation $\mathcal{R}$ on processes satisfies *fair/should predicate* if $P\mathcal{R}Q$ implies that, for every observer O, all tests of $P \mid O$ are FS-successful if and only if all tests of $Q \mid O$ are FS-successful. Let $\approx_{FS}$ be the largest relation that satisfies this predicate. This equivalence stays between Milner's weak bisimilarity and trace equivalence.

II. Boreale, De Nicola and Pugliese address issue II in [Boreale et al. 1999; 2001]. Instead of using the special symbol ω, they let all the observable actions to indicate success. Their version of the fair/should predicate is based on a slightly different notion of success. A test of P by O is *BDP-successful* at ℓ if $Q \stackrel{\ell}{\Longrightarrow}$

whenever $P \,|\, O \Longrightarrow Q$. Notice that there is a guarantee at ℓ predicate for every $\ell \in \mathcal{L}$. Boreale, De Nicola and Pugliese's equivalence $\approx_{BDP}$ is the largest reflexive contextual relation satisfying all the guarantee at ℓ predicates. It is proved in [Boreale et al. 1999] that $\approx_{BDP}$ coincides with $\approx_{FS}$ for the calculus considered in [Boreale et al. 1999]. The significance of their approach is that it provides a characterization of the must equivalence without resorting to any testing machinery.

We shall cast more light on $\approx_{FS}$ and $\approx_{BDP}$ next.

3.2.3 *Testing Equivalence without Testing.* Although the study in [Boreale et al. 1999] is carried out for a particular calculus with a particular notion of context, Boreale, De Nicola and Pugliese's approach to testing equivalence is basically model independent. In this subsection we shall give an even more abstract characterization of $\approx_{BDP}$ in the style of the absolute equality. We start with a similar abstract characterization of the trace equivalence. The proof of the following lemma is essentially given in [Boreale et al. 1999].

LEMMA 3.37. *The equivalence $\approx_{may}$ is the largest reflexive, equipollent, extensional relation.*

In view of Lemma 3.37 we may introduce the following definition.

Definition 3.38. The *diamond equality* $=_\diamond$ is the largest reflexive, equipollent, extensional relation.

The diamond equality is about the existence of a successful testing. A logical dual would be about the inevitability of successful testings. For the purpose of introducing such a dual, one need to strengthen the equipollence condition.

Definition 3.39. A process P is *strongly observable*, notation $P{\downarrow\downarrow}$, if $P'{\Downarrow}$ for all P' such that $P \Longrightarrow P'$. A relation $\mathcal{R}$ is *strongly equipollent* if $P{\downarrow\downarrow} \Leftrightarrow Q{\downarrow\downarrow}$ whenever $P\mathcal{R}Q$.

After Definition 3.39, the following definition must be expected.

Definition 3.40. The *box equality* $=_\Box$ is the largest reflexive, strongly equipollent, extensional relation.

The first indication that the box equality offers a better process equivalence than the must equivalence is the property described in the next lemma.

LEMMA 3.41. *The strict inclusion $=_\Box \subsetneq =_\diamond$ holds.*

PROOF. The proof of Lemma 3.36 can be reiterated. Notice that the strongly convergent condition is not necessary in this case. The strictness is witnessed by the process pair $a.(!b \,|\, !c)$ and $a.b.(!b \,|\, !c) + a.c.(!b \,|\, !c)$. □

The next lemma confirms that the box equality captures the essence of the must semantics.

LEMMA 3.42. *The box equality and the must equivalence coincide on the strongly convergent processes.*

PROOF. Suppose $P \not\approx_{must} Q$. Without loss of generality, assume that there is some observer O such that $P \,|\, O$ has an unsuccessful test and all the tests of $Q \,|\, O$ are successful. Suppose the failure test of $P \,|\, O$ is the infinite tau action sequence

$$P \,|\, O \xrightarrow{\tau} (\widetilde{c_1})(P_1 \,|\, O_1) \ldots \xrightarrow{\tau} (\widetilde{c_i})(P_i \,|\, O_i) \ldots . \tag{17}$$

If O performs an infinite consecutive tau action sequence in (17) then it must be the case that $P \neq_\diamond Q$. If not then by the strong convergence property P must perform an infinite number of external actions in (17). It follows from the assumption that Q can not perform the same infinite number of external actions. So $P \neq_\diamond Q$ also holds in this case. If the failure test of $P \,|\, O$ is the finite internal action sequence

$$P \,|\, O \xrightarrow{\tau} (\widetilde{c_1})(P_1 \,|\, O_1) \ldots \xrightarrow{\tau} (\widetilde{c_i})(P_i \,|\, O_i),$$

then either $P \neq_\diamond Q$ or there exists some ℓ^* such that for each fresh name f one has $(\widetilde{c})(Q \,|\, \ell^* \overline{f} f) \downarrow\downarrow$ but not $(\widetilde{c})(P \,|\, \ell^* \overline{f} f) \downarrow\downarrow$, where $\{\widetilde{c}\} \subseteq gn(P \,|\, Q)$. We conclude that $=_\Box \subseteq \approx_{must}$.

Suppose $P \neq_\Box Q$ for strongly convergent processes P and Q. We prove the contrapositive of $\approx_{must} \subseteq =_\Box$. Without loss of generality, assume that there were $\widetilde{c}$ and O such that $(\widetilde{c})(P \,|\, O)\downarrow\downarrow$ holds but $(\widetilde{c})(Q \,|\, O)\downarrow\downarrow$ is not valid. Let $(\widetilde{c'})(Q' \,|\, O')$ be such that $(\widetilde{c})(Q \,|\, O) \Longrightarrow (\widetilde{c'})(Q' \,|\, O') \not\downarrow\downarrow$. Then there must be a sequence of actions ℓ_1^* such that $Q \stackrel{\ell_1^*}{\Longrightarrow} Q'$ and $O \stackrel{\overline{\ell_1^*}}{\Longrightarrow} O'$. If $\neg(P \stackrel{\ell_1^*}{\Longrightarrow})$, then let ℓ^* be ℓ_1^*. If $P \stackrel{\ell_1^*}{\Longrightarrow} P'$ for some P' then $(\widetilde{c'})(P' \,|\, O')\Downarrow$ by definition. So there is some observer $(\widetilde{c''})(\_ \,|\, O'')$ and some non-tau action λ such that $(\widetilde{c'})(P' \,|\, O') \Longrightarrow (\widetilde{c''})(P'' \,|\, O'')\downarrow_\lambda$. Assume that $P' \stackrel{\ell_2^*}{\Longrightarrow} P''$. If $Q' \stackrel{\ell_2^*}{\Longrightarrow} Q''$ for some Q'', then $(\widetilde{c''})(P'' \,|\, O'')\downarrow_\lambda$ implies $P''\downarrow_\lambda$ since $(\widetilde{c''})(Q'' \,|\, O'')\not\downarrow_\lambda$. In this case let ℓ^* be $\ell_1^*\ell_2^*\lambda$. If $\neg(Q' \stackrel{\ell_2^*}{\Longrightarrow})$, then let ℓ^* be $\ell_1^*\ell_2^*$. Let $\mathcal{F} = gn(P \,|\, Q)$. It is easy to see that $P \,|\, O_{\ell^*}^{\mathcal{F}}$ has an unsuccessful test but $Q \,|\, O_{\ell^*}^{\mathcal{F}}$ does not. This completes the proof of $\approx_{must} \subseteq =_\Box$. □

We can now summarize the main result of this section by the following theorem. It says that the box equality improves upon the testing equivalence in the best way one could expect.

THEOREM 3.43. *The following statements are valid.*

(1) *The strict inclusions* $= \subsetneq =_\Box \subsetneq =_\diamond$ *hold.*
(2) *The equivalences* $=_\Box, \approx_{must}$ *coincide on the strongly convergent processes.*

So far all the results in testing theory are model dependent. For example the coincidence between $\approx_{FS}$ and $\approx_{BDP}$ cannot be established in a model independent way. This is because to make sense of the fair/should testing one has to incorporate the special symbol ω into a calculus, which is impossible without knowing the details of the model. Similarly the coincidence of $\approx_{FS}$, $\approx_{BDP}$, $\approx_{must}$ on the strongly convergent processes is also model specific. For a particular model like π-calculus one may show that $\approx_{BDP}$ is the largest reflexive, strongly equipollent, extensional relation. But there is no way to prove the validity of this characterization for all models. Similarly the coincidence between $\approx_{BDP}$ and $=_\Box$ can only be proved for individual models. The same remark applies to the trace equivalence. There are labeled transition systems for which one might not like to talk about traces at all.

An example is a labeled transition semantics for a higher order calculus that defines transitions from processes to abstractions or concretions.

It is in the light of the above discussion that the significance of the diamond equality and the box equality emerge. The former is *the* universal definition of the trace equivalence, whereas the latter is *the* universal definition of the testing equivalence. These definitions apply to all models of interaction. For a particular model it is possible to give an explicit counterpart of $=_\Box$ or $=_\Diamond$. The explicit versions of $=_\Diamond$ and $=_\Box$ may depend heavily on the details of the model. The relationship between $=_\Box$ and its explicit characterization is like that between the absolute equality and the external bisimilarity.

The outline for a model independent testing theory is now complete.

3.3 Remark

The model independent methodology can be applied to analyze dozens of process equivalences defined in literature. In particular it can be applied to evaluate the equivalence relations for example in the linear time branching time spectrum [van Glabbeek 1990]. We have confirmed in this paper that the diamond equality is the right generalization of the trace equivalence, the bottom of the spectrum. On the top of the spectrum, the absolute equality and the weak equality enjoy the model independent characterizations. It would be interesting to take a look at the other equivalences of the spectrum from this angle. Good process equivalences are defined in a model independent way.

Having said that it must be emphasized that it is often difficult to come up with an external characterization of an equality defined in a model independent manner. Let's take a look at an interesting example. An explicit characterization of the absolute equality for π^A-calculus is tricky. The asynchronous equivalence studied by Honda and Tokoro [1991a], and by Amadio et al. [1996], satisfies the following equality, where $\simeq_b$ stands for the barbed equivalence.

$$a(x).\overline{a}x \simeq_b \mathbf{0}. \tag{18}$$

The equality (18) does not hold for the absolute equality since it violates the equipollence condition. Another equality validated by the asynchronous equivalence is (19).

$$!a(x).\overline{a}x \simeq_b a(x).\overline{a}x \tag{19}$$

Equality (19) is rejected by the absolute equality because $\overline{a}c \,|\, !a(x).\overline{a}x$ is divergent but $\overline{a}c \,|\, a(x).\overline{a}x$ is not. However the following absolute equality holds.

$$a(x).\overline{a}x \,|\, a(x).\overline{a}x =_{\pi^A} a(x).\overline{a}x \tag{20}$$

So an external characterization of $=_{\pi^A}$ is yet to be worked out. The asynchronous π-calculus suggests that there is a discrepancy between the absolute equality and the asynchronous equivalence. How should we reconcile the difference? A new approach to the asynchronous calculus that gives a satisfactory answer to the question is outlined in [Fu 2010a].

We conclude this section by remarking that the branching style bisimulation property is what makes the coincidence proofs difficult. External characterizations of the weak equalities of the π-variants are much easier to come by.

4. EXPRESSIVENESS

How do different variants of π-calculus compare? One fundamental criterion for comparison is relative expressiveness. Is a calculus $\mathbb{M}$ as powerful as another calculus $\mathbb{L}$? Or are $\mathbb{L}$ and $\mathbb{M}$ incomparable in terms of expressiveness? To answer these questions, we need a notion of relative expressiveness that does not refer to any particular model. The philosophy developed in [Fu 2010b] is that relative expressiveness is the same thing as the process equality. The latter is a relation on one calculus, whereas the former is a relation from one calculus to another. To say that $\mathbb{M}$ is at least expressive as $\mathbb{L}$ means that for every process L in $\mathbb{L}$ there is some process M in $\mathbb{M}$ such that M is somehow equal to L. It is clear from this statement that relative expressiveness must break the symmetry of the absolute equality. Condition (2) of Definition 3.7 can be safely kept when comparing two process calculi. But condition (1) has to be modified. The reader is advised to consult [Fu 2010b] for the argument why the reflexivity condition for the absolute equality turns into totality condition and soundness condition. The following definition is taken from [Fu 2010b].

Definition 4.1. Suppose $\mathbb{L}, \mathbb{M}$ are two process calculi. A binary relation $\Re$ from the set $\mathcal{P}_{\mathbb{L}}$ of $\mathbb{L}$-processes to the set $\mathcal{P}_{\mathbb{M}}$ of $\mathbb{M}$-processes is a *subbisimilarity* if the following statements are valid.

(1) $\Re$ is *reflexive from* $\mathbb{L}$ *to* $\mathbb{M}$ in the sense that the following properties hold.
   (a) $\Re$ is *total*, meaning that $\forall L \in \mathcal{P}_{\mathbb{L}}.\exists M \in \mathcal{P}_{\mathbb{M}}.L\Re M$.
   (b) $\Re$ is *sound*, meaning that $M_1\Re^{-1}L_1 =_{\mathbb{L}} L_2\Re M_2$ implies $M_1 =_{\mathbb{M}} M_2$.

(2) $\Re$ is equipollent, extensional, codivergent and bisimilar.

We say that $\mathbb{L}$ is *subbisimilar to* $\mathbb{M}$, notation $\mathbb{L} \sqsubseteq \mathbb{M}$, if there is a subbisimilarity from $\mathbb{L}$ to $\mathbb{M}$.

It is shown in [Fu 2010b] that in general there are an infinite number of pairwise incompatible subbisimilarities from $\mathbb{L}$ to $\mathbb{M}$.

An elementary requirement for the relative expressiveness relationship is transitivity. The subbisimilarity relationship is well defined in this aspect since its transitivity is apparent from the definition.

So we have formalized the intuitive notion "$\mathbb{M}$ being at least as expressive as $\mathbb{L}$" by "$\mathbb{L} \sqsubseteq \mathbb{M}$". We write $\mathbb{L} \sqsubset \mathbb{M}$ if $\mathbb{L} \sqsubseteq \mathbb{M}$ and $\mathbb{M} \not\sqsubseteq \mathbb{L}$, meaning that $\mathbb{M}$ is strictly more expressive than $\mathbb{L}$. The reader might wonder why the soundness condition of Definition 4.1 is not strengthened to the full abstraction. The truth is that the soundness condition is equivalent to the full abstraction condition as far as Definition 4.1 is concerned.

Consider the self translation $[\![\_]\!]_{\bowtie}$ from π-calculus to itself. The nontrivial part of the translation is defined as follows.

$$[\![a(x).T]\!]_{\bowtie} \stackrel{\text{def}}{=} \overline{a}(c).c(x).[\![T]\!]_{\bowtie},$$
$$[\![\overline{a}b.T]\!]_{\bowtie} \stackrel{\text{def}}{=} a(y).(\overline{y}b \mid [\![T]\!]_{\bowtie}).$$

It is structural on the non-prefix terms. Is $[\![\_]\!]_{\bowtie}; =$ a subbisimilarity? The negative answer is given in [Fu 2010b]. This is an example of a typical phenomenon. Without the soundness condition, many liberal encodings like the above one would be

$\sqsubseteq$	π	π^L	π^R	π^1
π	-	?	?	?
π^L	?	-	?	?
π^R	?	?	-	?
π^1	?	?	?	-

Fig. 3. Expressiveness Relationship

admitted. In most cases soundness is the only thing to prove when comparing a syntactical subcalculus against the super calculus. Now suppose the external bisimilarity coincides with the absolute equality in π-variants π^1 and π^2, then $\pi^1 \sqsubseteq \pi^2$ if there is a total relation $\Re$ from π^1 to π^2 such that the following statements are valid.

(1) If $Q\Re^{-1}P \xrightarrow{\ell} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'\Re^{-1}P'$ and $P\Re Q''$ for some Q', Q''.

(2) If $P\Re Q \xrightarrow{\ell} Q'$ then $P \Longrightarrow P'' \xrightarrow{\ell} P'\Re Q'$ and $P''\Re Q$ for some P', P''.

Such an external characterization is also difficult to come by. So far we have not been able to confirm that $\pi^M \sqsubseteq \pi^- \sqsubseteq \pi$, although the following negative results are easy to derive.

PROPOSITION 4.2. $\pi \not\sqsubseteq \pi^- \not\sqsubseteq \pi^M$.

PROOF. Using the same idea from [Fu 2010b] it is routine to show that the π-process $a(x).\bar{b}b + a(x).\bar{c}c$ can not be defined in π^- and the π^--process $a(x).[x{=}c]\bar{b}b$ can not be defined in π^M. □

Proposition 4.2 is all we know about the relative expressiveness of the π-variants. The relative expressiveness between π, π^L, π^R, π^1 is important enough to be stated as an open problem.

Problem 4.3. What are the answers to the questions posed in Fig. 3?

Again the branching style bisimulation is what makes things difficult.

Before ending this section, let's see a positive result. Consider the boolean expressions constructed from the binary relation $=$ and the logical operators $\wedge, \neg$. For example $\neg(\neg(x = a) \wedge \neg(x = b))$ is such an expression, which is normally abbreviated to $x = a \vee x = b$. Let $\pi^\vee$ denote the π-calculus with the match and mismatch operators replaced by the operator $[\varphi](\_)$, where φ is a boolean expression.

PROPOSITION 4.4. $\pi \sqsubseteq \pi^\vee \sqsubseteq \pi$.

PROOF. The external bisimilarity and the absolute equality coincide for $\pi^\vee$. So $\pi \sqsubseteq \pi^\vee$ is obvious. The proof of the converse is equally simple as soon as the encodings of processes of the form $[\varphi]T$ become clear. Given any φ, one may transform it into a disjunctive normal form $\bigvee_{i\in I} \varphi_i$. By applying the equivalence

$$\bigvee_{i\in I} \varphi_i \Leftrightarrow \bigvee_{i\in I} \varphi_i \wedge (u = v) \vee \bigvee_{i\in I} \varphi_i \wedge (u \neq v),$$

one gets a complete disjoint partition $\bigvee_{j\in J} \psi_j$ of φ on $fv([\varphi]T) \cup gn([\varphi]T)$. It follows that $[\varphi]T = \prod_{j\in J}[\psi_j]T$ holds in $\pi^\vee$. Therefore $[\varphi]T$ can be bisimulated by $\prod_{j\in J}[\psi_j]T$. □

5. PROOF SYSTEM

An important issue for a process calculus is to design algorithms for the decidable fragments of the calculus. The behavior of a finite term can be obviously examined by a terminating procedure [Milner 1989a]. More generally if a term that has only a finite number of descendants and is finite branching, then there is an algorithm that generates its transition graph [Lin 1996], which can be seen as the abstract representation of the term. An equivalence checker for the terms now works on the transitions graphs. To help transform the terms to their underlying transition graphs, a recursive enumerable equational rewrite system would be helpful.

In this section we construct two equational systems of the finite π-terms, one for the absolute equality and the other for the box equality. Our prime objective is to demonstrate how the name dichotomy simplifies equational reasoning.

5.1 Normal Form

To construct an equational system, it is always necessary to introduce a combinator that is capable of describing the nondeterministic choices inherent in concurrent systems. This is the *general choice* operator '+'. The semantics of this operator is defined by the following rules.

$$\frac{S \xrightarrow{\lambda} S'}{S+T \xrightarrow{\lambda} S'} \qquad \frac{T \xrightarrow{\lambda} T'}{S+T \xrightarrow{\lambda} T'}$$

The choice operation is both commutative and transitive. We will write $T_1+\ldots+T_n$ or $\sum_{i\in I} T_i$, called a *summation*, without further comment. Here T_i is a summand. Occasionally this notation is confused with the one defined in (9) or (10). When this happens, the confusions are harmless. Using the unguarded choice operator, one may define for example $[p \notin \mathcal{F}]T$ by

$$\sum_{m\in\mathcal{F}} [p=m]T.$$

In general one could define $(\varphi)T$ for a boolean expression constructed from $=, \wedge, \neg$. It follows that *if* φ *then* S *else* T is definable using the general choice operator. The choice operator is a debatable operator since it destroys the congruence property of the absolute equality. In most of the axiomatic treatment of this operator an induced congruence relation is introduced. This is avoided in this paper since we see the choice operator not as a proper process operator but as an auxiliary one used in the proof systems.

One criterion for an equational system is if it allows one to reduce every term to some normal form. The exercise carried out in Section 3.1.4 suggests the following definitions.

Definition 5.1. Let $\mathcal{F}$ be $gn(T) \cup fv(T)$. The π-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 $\mathcal{F}$.

L1	$(c)\mathbf{0}$	$=$	$\mathbf{0}$	
L2	$(c)(d)T$	$=$	$(d)(c)T$	
L3	$(c)\pi.T$	$=$	$\pi.(c)T$	if $c \notin n(\pi)$
L4	$(c)\pi.T$	$=$	$\mathbf{0}$	if $c = subj(\pi)$
L5	$(c)\varphi T$	$=$	$\varphi(c)T$	if $c \notin n(\varphi)$
L6	$(c)[x{=}c]T$	$=$	$\mathbf{0}$	
L7	$(c)(S+T)$	$=$	$(c)S + (c)T$	
M1	$(\top)T$	$=$	T	
M2	$(\bot)T$	$=$	$\mathbf{0}$	
M3	φT	$=$	ψT	if $\varphi \Leftrightarrow \psi$
M4	$[x{=}p]T$	$=$	$[x{=}p]T\{p/x\}$	
M5	$[x{\neq}p]\pi.T$	$=$	$[x{\neq}p]\pi.[x{\neq}p]T$	if $x \notin bv(\pi) \not\ni p$
M6	$\varphi(S+T)$	$=$	$\varphi S + \varphi T$	
S1	$T + \mathbf{0}$	$=$	T	
S2	$S + T$	$=$	$T + S$	
S3	$R + (S+T)$	$=$	$(R+S) + T$	
S4	$T + T$	$=$	T	
S5	T	$=$	$[x{=}p]T + [x{\neq}p]T$	
S6	$n(x).S + n(x).T$	$=$	$n(x).S + n(x).T + n(x).([x{=}p]S + [x{\neq}p]T)$	
C1	$\lambda.\tau.T$	$=$	$\lambda.T$	
C2	$\tau.(S+T)$	$=$	$\tau.(\tau.(S+T)+T)$	

Fig. 4. Axioms for Absolute Equality.

(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 $\mathcal{F}$.

Definition 5.2. T is a *complete normal form* on $\mathcal{F}$ if it is of the form $\delta_{\mathcal{F}}T'$ for some normal form T' such that $gn(T') \cup fv(T') \subseteq \mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$.

5.2 Axiom for Absolute Equality

The equational system for the absolute equality on the finite π-terms is given in Fig. 4. The axioms C1 and C2 are *computation laws*. Notice that C2 implies the following equality

$$\tau.T = \tau.(T + \tau.T).$$

We write $AS \vdash S = T$ if $S = T$ can be derived from the axioms and the rules in Fig. 4 and the equivalence and congruence rules. Some derived axioms are summarized in the next lemma. The proofs of these derived axioms can be found in for example [Fu and Yang 2003].

Lemma 5.3. *The following propositions are valid.*

(1) $AS \vdash T = T + \varphi T$.
(2) $AS \vdash \varphi\pi.T = \varphi\pi.\varphi T$.

(3) $AS \vdash (c)[x{\neq}c]T = (c)T$ *and* $(c)[x{=}c]T = \mathbf{0}$.

(4) $AS \vdash \varphi T = \varphi T\sigma_\varphi$.

(5) $AS \vdash \lambda.\varphi\tau.T = \lambda.\varphi T$.

(6) $AS \vdash \sum_{i=1}^{k} m(x).T_i = \sum_{i=1}^{k} m(x).T_i + m(x).([x \notin \mathcal{F}]T_{k+1} + \sum_{i=1}^{k}[x{=}n_i]T_i)$, *where* $\mathcal{F}$ *is the set* $\{n_1, \ldots, n_k\}$.

Using these derived axioms it is easy to prove the following lemma by structural induction.

LEMMA 5.4. *Suppose T is a finite π-term and φ is complete on a finite set $\mathcal{F} \supseteq gn(T) \cup fv(T)$. Then $AS \vdash \varphi T = \varphi^{=}\delta T'$ for some complete normal form $\delta T'$ on $gn(\delta T') \cup fv(\delta T')$.*

PROOF. To start with, $AS \vdash \varphi T = \varphi^{=}\varphi^{\neq}T = \varphi^{=}(\varphi^{\neq}T)\sigma_{\varphi^{=}}$. In the second step observe that within AS we can carry out the reductions, explained in Section 3.1.4, to $(\varphi^{\neq}T)\sigma_{\varphi^{=}}$. Notice that, for each $\mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$, although $\{x{\notin}\mathcal{F}\} \cup \{x{=}m\}_{m\in\mathcal{F}}$ is not a complete disjoint partition of $\mathcal{F} \cup \{x\}$, the set $\{(x{\notin}\mathcal{F}) \wedge \delta_{\mathcal{F}}\} \cup \{(x{=}m) \wedge \delta_{\mathcal{F}}\}_{m\in\mathcal{F}}$ *is* a complete disjoint partition of $\mathcal{F} \cup \{x\}$. □

Suppose we are to prove $AS \vdash \varphi S = \varphi T$ where φ is complete on $gn(S \,|\, T) \cup fv(S \,|\, T)$. According to Lemma 5.4 we only need to prove $AS \vdash \delta S' = \delta T'$ for some complete normal forms $\delta S', \delta T'$. In the light of this fact the next lemma should play a crucial role in the completeness proof.

LEMMA 5.5. *Suppose S, T are normal forms on the finite set $\mathcal{F} \supseteq gn(S \,|\, T) \cup fv(S \,|\, T)$. If $\delta_{\mathcal{F}} S \simeq \delta_{\mathcal{F}} T$ then $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.T$.*

PROOF. Assume that $S \equiv \sum_{i\in I} \lambda_i.S_i$ and $T \equiv \sum_{j\in J} \lambda_j.T_j$ and that $\delta_{\mathcal{F}} S \simeq \delta_{\mathcal{F}} T$ for some $\mathcal{F} \supseteq gn(S \,|\, T) \cup fv(S \,|\, T)$. Let ρ be an assignment that agrees with $\delta_{\mathcal{F}}$. We are going to establish by simultaneous induction on the structure of S, T the properties stated below.

(S) If $T\rho \to^{+} T'\rho \not\to$ then $AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + T') = \delta_{\mathcal{F}}\tau.T'$.

(P) If $\delta_{\mathcal{F}} S \simeq \delta_{\mathcal{F}} T$ then $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.(S + T) = \delta_{\mathcal{F}}\tau.T$.

Suppose $T\rho \to T_1\rho \to \ldots \to T_n\rho \to T'\rho \not\to$. Lemma 3.32 and Lemma 3.33 imply that $\delta_{\mathcal{F}} T_1 \simeq \ldots \simeq \delta_{\mathcal{F}} T_n \simeq \delta_{\mathcal{F}} T'$. By induction hypothesis on (P), $AS \vdash \delta_{\mathcal{F}}\tau.T_1 = \ldots = \delta_{\mathcal{F}}\tau.T_n = \delta_{\mathcal{F}}\tau.T'$. Therefore

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + \tau.T').$$

Let $\lambda_j.T_j$ be a summand of T. Consider how $T'\rho$ might bisimulate $T\rho \xrightarrow{\lambda_j} T_j\rho$. There are four cases.

(1) $\lambda_i = \tau$ and $T\rho \xrightarrow{\tau} T_j\rho$. Suppose the τ-action is bisimulated by $T'\rho \xrightarrow{\tau} T'_j\rho \simeq T_j\rho$ for some T'_j. Then $\delta_{\mathcal{F}} T'_j \simeq \delta_{\mathcal{F}} T_j$ by Lemma 3.32 and Lemma 3.33. It follows from the induction hypothesis on (P) that $AS \vdash \delta_{\mathcal{F}}\tau.T'_j = \delta_{\mathcal{F}}\tau.T_j$. Using (2) of Lemma 5.3 one gets the following inference

$$\begin{aligned} AS \vdash \delta_{\mathcal{F}}\tau.(T + \tau.T') &= \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T'_j)) \\ &= \delta_{\mathcal{F}}\tau.(T + \tau.(T' + \tau.T_j)). \end{aligned}$$

If $T\rho \xrightarrow{\tau} T_j\rho$ is bisimulated vacuously by $T'\rho$. Then $\delta_{\mathcal{F}}T_j \simeq \delta_{\mathcal{F}}T'$ according to Lemma 3.32 and Lemma 3.33 and consequently $AS \vdash \delta_{\mathcal{F}}\tau.T_j = \delta_{\mathcal{F}}\tau.T'$ by the induction hypothesis on (P). It follows from C2 that

$$\begin{aligned} AS \vdash \delta_{\mathcal{F}}\tau.(T+\tau.T') &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+\tau.T')) \\ &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+\tau.T_j)). \end{aligned}$$

(2) $\lambda_i = \overline{n}m$ and $T\rho \xrightarrow{\overline{a}b} T_j\rho$ for some a, b. We can prove as in the next case that $AS \vdash \delta_{\mathcal{F}}\tau.(T+\tau.T') = \delta_{\mathcal{F}}\tau.(T+\tau.(T'+\overline{n}m.T_j))$.

(3) $\lambda_i = \overline{m}(c)$ and $T\rho \xrightarrow{\overline{a}(c)} T_j\rho$ for some a. Suppose this action is bisimulated by $T'\rho \xrightarrow{\overline{a}(c)} T'_j\rho \simeq T_j\rho$ for some T'_j. Then $\delta_{\mathcal{F}}[c\notin\mathcal{F}]T'_j \simeq \delta_{\mathcal{F}}[c\notin\mathcal{F}]T_j$ and

$$AS \vdash \delta_{\mathcal{F}}[c\notin\mathcal{F}]\tau.T'_j = \delta_{\mathcal{F}}[c\notin\mathcal{F}]\tau.T_j$$

by induction hypothesis on (P). Therefore

$$AS \vdash \delta_{\mathcal{F}}\overline{m}(c).[c\notin\mathcal{F}]T'_j = \delta_{\mathcal{F}}\overline{m}(c).[c\notin\mathcal{F}]T_j.$$

If follows from (3) of Lemma 5.3 that

$$AS \vdash \delta_{\mathcal{F}}\tau.(T+\tau.T') = \delta_{\mathcal{F}}\tau.(T+\tau.(T'+\overline{m}(c).T_j)).$$

(4) $\lambda_i = m(x)$. Assume that $\rho(m) = a$ and $gn((S \mid T)\rho) = \{b_1, \ldots, b_n\}$. Let $n_k \in \mathcal{F}$ be such that $n_k = b_k$ if $b_k \in \mathcal{F}$ and $\rho(n_k) = b_k$ if $b_k \notin \mathcal{F}$. There are two subcases.

(a) For every $k \in \{1, \ldots, n\}$, one has $T\rho \xrightarrow{ab_k} T_j\rho\{b_k/x\}$. Let this action be bisimulated by $T'\rho \xrightarrow{ab_k} T'_{n_k}\rho\{b_k/x\} \simeq T_j\rho\{b_k/x\}$. Then

$$AS \vdash \delta_{\mathcal{F}}[x{=}n_k]\tau.T'_{n_k} = \delta_{\mathcal{F}}[x{=}n_k]\tau.T_j$$

by induction hypothesis on (P). Consequently

$$\begin{aligned} \delta_{\mathcal{F}}\tau.(T+\tau.T') &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).T'_{n_k})) \\ &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).([x{\neq}n_k]\tau.T'_{n_k}+[x{=}n_k]\tau.T'_{n_k}))) \\ &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).([x{\neq}n_k]\tau.T'_{n_k}+[x{=}n_k]\tau.T_j))). \end{aligned}$$

(b) Suppose $T\rho \xrightarrow{ad} T_j\rho\{d/x\}$, where $d \notin \mathcal{F}$, is simulated by

$$T'\rho \xrightarrow{ad} T_x\rho\{d/x\} \simeq T_j\rho\{d/x\}.$$

Like in the previous subcase, we may prove that

$$AS \vdash \delta_{\mathcal{F}}\tau.(T+\tau.T') = \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).([x{\in}\mathcal{F}]\tau.T_x+[x{\notin}\mathcal{F}]\tau.T_j))).$$

Putting together the two equalities obtained in (a) and (b), we get the following equational rewriting

$$\begin{aligned} AS \vdash \delta_{\mathcal{F}}\tau.(T+\tau.T') &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).([x{\in}\mathcal{F}]\tau.T_x+[x{\notin}\mathcal{F}]\tau.T_j) \\ &\quad + \sum_{j=1}^{n} m(x).([x{\neq}n_k]\tau.T_{n_k}+[x{=}n_k]\tau.T_j))) \\ &= \delta_{\mathcal{F}}\tau.(T+\tau.(T'+m(x).T_j)), \end{aligned}$$

where the second equality holds by (6) of Lemma 5.3.

In summary we may prove by simple induction that

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + \tau.(T' + T)).$$

Resorting to the full power of C2 we get from the above equality the following.

$$AS \vdash \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.(T + T'). \tag{21}$$

If we examine the proof of (21) carefully we realize that it also establishes the following.

$$AS \vdash \delta_{\mathcal{F}}\tau.(T + T') = \delta_{\mathcal{F}}\tau.T'.$$

This finishes the inductive proof of (S).

The inductive proof of (P) is now simpler. Suppose $\delta_{\mathcal{F}}S \simeq \delta_{\mathcal{F}}T$. Let S', T' be such that $S\rho \rightarrow^* S'\rho \not\rightarrow$ and $T\rho \rightarrow^* T'\rho \not\rightarrow$. According to the induction hypothesis of (S), the followings hold.

$$AS \;\vdash\; \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.S',$$
$$AS \;\vdash\; \delta_{\mathcal{F}}\tau.T = \delta_{\mathcal{F}}\tau.T'.$$

The inductive proof of (S) can be reiterated to show $AS \vdash \delta_{\mathcal{F}}\tau.S' = \delta_{\mathcal{F}}\tau.T'$. Hence $AS \vdash \delta_{\mathcal{F}}\tau.S = \delta_{\mathcal{F}}\tau.T$. □

It is a small step from Lemma 5.5 to the completeness result.

THEOREM 5.6. *Suppose S, T are finite π-processes. Then $S \simeq T$ if and only if $AS \vdash \tau.S = \tau.T$.*

PROOF. Let $\mathcal{F}$ be $gn(S \,|\, T) \cup fv(S \,|\, T)$ and let $\{\varphi_i\}_{i\in I}$ be a complete disjoint partition $\mathcal{F}$. Then $S \simeq T$ if and only if $\varphi_i S \simeq \varphi_i T$ for every $i \in I$. For each $i \in I$, $\varphi_i S \simeq \varphi_i T$ can be turned into an equality of the form $\varphi_i^{=}(\varphi_i^{\neq} S')\sigma_{\varphi_i^{=}} \simeq \varphi_i^{=}(\varphi_i^{\neq} T')\sigma_{\varphi_i^{=}}$. Using Lemma 3.31 and Lemma 5.4, this equality can be simplified to $\delta S'' \simeq \delta T''$, where $\delta S''$ and $\delta T''$ are complete normal forms. We are done by applying Lemma 5.5. □

5.3 Axiom for Box Equality

According to Lemma 3.42, a proof system for the box equality on the finite π-terms is the same as a proof system for the testing equivalence on the finite π-terms. De Nicola and Hennessy have constructed an equational system for the testing equivalence on the finite CCS processes [De Nicola and Hennessy 1984]. Built upon that system, Boreale and De Nicola have studied the equational system for the testing equivalence on the finite π-processes [Boreale and De Nicola 1995]. So there is not much novelty about an equational proof system for the box equality on the finite π-terms. It would be however instructive in the present framework to give an outline of the proof technique that reduces the completeness for the box equality to the completeness for the absolute equality.

Since $\simeq \subseteq =_{\Box}$, one may devise an equational system for the latter by extending the system given in Fig. 4. The additional axioms are given in Fig. 5. These laws are the well known axioms for the testing equivalence of De Nicola and Hennessy [1984] adapted to the π-calculus. It is well known that they subsume Milner's τ-laws (See Fig. 6). Let AS_b be the system $AS \setminus \{C1, C2\} \cup \{N1, N2, N3, N4\}$. It is

N1	$\lambda.S + \lambda.T$	$=$	$\lambda.(\tau.S + \tau.T)$
N2	$S + \tau.T$	$=$	$\tau.(S + T) + \tau.T$
N3	$n(x).S + \tau.(n(x).T + R)$	$=$	$\tau.(n(x).S + n(x).T + R)$
N4	$\overline{n}l.S + \tau.(\overline{n}m.T + R)$	$=$	$\tau.(\overline{n}l.S + \overline{n}m.T + R)$

Fig. 5. Axioms for Box Equivalence.

T1	$\lambda.\tau.T$	$=$	$\lambda.T$
T2	$T + \tau.T$	$=$	$\tau.T$
T3	$\lambda.(S + \tau.T)$	$=$	$\lambda.(S + \tau.T) + \lambda.T$

Fig. 6. Milner's Tau Laws.

routine to check that AS_b is sound for the box equality. To prove that the system is also complete, we apply the following strategy.

> $S =_\Box T$ if and only if there exist some S', T' such that $AS_b \vdash S = S'$, $AS_b \vdash T = T'$ and $S' = T'$.

The soundness of the strategy relies on the fact that AS_b is complete for $=^w$. To make the strategy work, we need to introduce a special set of π-terms, different from the complete normal forms, so that the box equality and the absolute equality coincide on these special π-terms.

Axiom N1 suggests that a summation may be rewritten to a form in which no two summands have identical non-tau prefix. For instance

$$AS_b \vdash a(x).S + a(y).T = a(z).(\tau.S\{z/x\} + \tau.T\{z/y\})$$

and

$$AS_b \vdash \overline{a}(b).S + \overline{a}(c).T = \overline{a}(d).(\tau.S\{d/b\} + \tau.T\{d/c\}).$$

Axiom N2 implies that either all the summands of a summation are prefixed by τ, or none of them is prefixed by τ. Moreover N1 actually says that one does not have to consider any τ-prefix immediately underneath another τ-prefix. Axioms N3 and N4 can be used to expand a summation $\sum_{i\in I} \tau.T_i$ to a saturated form. We say that $\sum_{i\in I} \tau.T_i$ is *saturated* if $\tau.(\lambda_1.T_1 + T_j)$ is a summand of $\sum_{i\in I} \tau.T_i$ whenever there is a summand $\lambda_1.T_1$ of T_i such that for every summand $\lambda_2.T_2$ of T_j it holds up to α-conversion that $\lambda_2 \neq \lambda_1$. These observations lead to the following definition.

Definition 5.7. Let $\mathcal{F}$ be $gn(T) \cup fv(T)$. A finite π-term T is a *box normal form* if it is in one of the following two forms:

(1) There are $A, B, C \subseteq_f \mathcal{N}$, where $C \subseteq B$, such that T is of the shape:

$$\sum_{a\in A} a(x).\left([x\notin\mathcal{F}]T_a + \sum_{n\in\mathcal{F}} [x{=}n]T_a^n\right) + \sum_{b\in B}\sum_{n\in N_b} \overline{b}n.T_b^n + \sum_{c\in C} \overline{c}(d).T_c \qquad (22)$$

where $N_b \subseteq_f \mathcal{N} \cup \mathcal{N}_v$; and moreover the following properties hold:
(a) for all $a \in A$, T_a is a box normal form on $\mathcal{F} \cup \{x\}$;
(b) for all $a \in A$ and all $n{\in}\mathcal{F}$, $x \notin fv(T_a^n)$ and T_a^n is a box normal form on $\mathcal{F}$;
(c) for all $b \in B$ and all $n{\in}N_b$, T_b^n is a box normal form on $\mathcal{F}$;

(d) for all $c \in C$, T_c is a box normal form on $\mathcal{F} \cup \{d\}$.

(2) There is some finite set I such that T is of the following form

$$\sum_{i \in I} \tau.T_i \tag{23}$$

such that the following properties hold:
—For each $i \in I$, T_i is a box normal form on $\mathcal{F}$ of the shape (22);
—$\sum_{i \in I} \tau.T_i$ is saturated.

In the above definition we have ignored the conditionals. This is rectified in the next definition.

Definition 5.8. T is a *complete box normal form* if $T \equiv \delta_{\mathcal{F}} T'$ for some box normal form T' such that $gn(T') \cup fv(T') \subseteq \mathcal{F} \subseteq_f \mathcal{N} \cup \mathcal{N}_v$.

We can now state a lemma that correlates Lemma 5.4.

LEMMA 5.9. *Suppose T is a finite π-term and φ is complete on a finite set $\mathcal{F} \supseteq gn(T) \cup fv(T)$. Then $AS_b \vdash \varphi T = \varphi^{=} \delta T'$ for some complete box normal form $\delta T'$ on $gn(\delta T') \cup fv(\delta T')$.*

PROOF. The proof is a modification of the proof of Lemma 5.4 with additional rewriting using N-laws. □

We could have a lemma that parallels Lemma 5.5. But the following result is more revealing.

LEMMA 5.10. *Suppose S, T are complete box normal forms on the finite set $\mathcal{F} = gn(S \,|\, T) \cup fv(S \,|\, T)$. Then $S =_{\Box} T$ if and only if $S = T$.*

PROOF. Suppose $S =_{\Box} T$. By Lemma 5.9 we may assume that $S \equiv \delta S'$ and $T \equiv \delta T'$. Let ρ be an assignment that agrees with δ. Then $S'\rho =_{\Box} T'\rho$. In view of Lemma 3.32, we only need to show that $S'\rho = T'\rho$. So let $\mathcal{R}$ be the relation

$$\{(P, Q) \mid P =_{\Box} Q, \text{ and } P, Q \text{ are box normal forms}\}.$$

There are three crucial properties about this relation.

(1) If P is of type (23) and Q is of type (22), then $P' \mathcal{R} Q$ whenever $P \xrightarrow{\tau} P'$. This is so simply because Q cannot do any τ-action.

(2) If both P and Q are of type (23), then $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q' \mathcal{R} P'$.

(3) If both P and Q are of type (22), then $P \xrightarrow{\lambda} P'$ implies $Q \xrightarrow{\lambda} Q' \mathcal{R} P'$.

For the detailed proofs of these claims, the reader is advised to consult [De Nicola and Hennessy 1984; Boreale and De Nicola 1995]. So $\mathcal{R}$ is a π-bisimulation. □

Theorem 5.6, Lemma 5.10 and the fact that complete box normal forms are complete normal forms immediately imply the completeness of AS_b.

THEOREM 5.11. *Suppose S, T are finite π-processes. Then $S =_{\Box} T$ if and only if $AS_b \vdash \tau.S = \tau.T$.*

It is remarkable that the axioms N1 through N4 actually reduces the box equality on the finite terms to the absolute equality. This is yet another support to the branching style bisimulation.

5.4 Remark

In theory of CCS, Milner's equational systems for the strong and the weak congruences on the finite CCS-processes [Hennessy and Milner 1985; Milner 1989a] and De Nicola and Hennessy's system for the testing congruence are well known. A complete system for the branching congruence is given by van Glabbeek and Weijland [1989] in the first paper on branching bisimulation, in which the following single tau law is proposed.

$$\lambda.(\tau.(S+T)+T) = \lambda.(S+T). \tag{24}$$

Their proof of the completeness makes use of a graph rewriting system. The axiom (24) is clearly equivalent to the combination of C1 and C2.

The extension of these systems to the value-passing framework is not trivial, the reason being that every value-passing calculus is built on top of an oracle domain, say $\mathfrak{D}$. Early inference systems studied by Hennessy and Ingólfsdóttir [Hennessy 1991; Hennessy and Ingólfsdóttir 1993a; 1993b] are based on concrete semantics. An uncomfortable rule in all these systems, from the point of view of a proof system, is the so-called ω-data-rule (25).

$$\frac{\forall v \in \mathfrak{D}.S\{v/x\} = T\{v/x\}}{a(x).S = a(x).T} \tag{25}$$

A significant step was made by Hennessy and Lin [1995] that introduces a whole new approach to the study of the value-passing calculi. The symbolic semantics dispenses with (25) by introducing a strong logic. One could argue that this use of a logic is cheating because it essentially makes use of a universal quantification operator that ranges over the oracle domain $\mathfrak{D}$. But the virtue of the symbolic approach is that one could define a value-passing calculus using a moderate logic that makes a lot of sense from a programming point of view, and then works out the observational theory with the help of the logic. Using this idea Hennessy and Lin [1996] propose several symbolic proof systems for finite value-passing processes. A survey of the symbolic approach is given in [Ingólfsdóttir and Lin 2001].

For the name-passing calculi, the study on the proof systems was initiated in the pioneering paper of Milner et al. [1992]. The system of [Milner et al. 1992] is complete for the strong early equivalence. It contains the ω-name-rule (26), which is a variant of (25).

$$\frac{\forall n \in \mathcal{N}.S\{n/x\} = T\{n/x\}}{a(x).S = a(x).T} \tag{26}$$

In the presence of the finite branching property, (26) is not as bad as (25) since only a finite number of the premises of (26) have to be verified. A beautiful alternative to *rule* (26) is Parrow and Sangiorgi's set of *axioms* for match/mismatch [Parrow and Sangiorgi 1995], among which the following one, S5, plays an indispensable role.

$$[x{=}y]T + [x{\neq}y]T = T \tag{27}$$

Axiom (27) offers the possibility to carry out case analysis within an equational system. It is obvious from (27) that Parrow and Sangiorgi's systems are only good for the π-calculi with the mismatch operator. The mismatch is also necessary to

state the following law, S6, which first appeared in [Parrow and Sangiorgi 1995].

$$n(x).S + n(x).T = n(x).S + n(x).T + n(x).([x{=}y]S + [x{\neq}y]T) \tag{28}$$

Axiom (28) characterizes the *atomic* nature of interactions. The symbolic approach to proof systems however makes use of neither (26) nor (28). Lin's symbolic proof systems [Lin 1995a; 2003] are capable of dealing with calculi with or without mismatch operator. Instead of (28), the systems in [Lin 1995a; 2003] resort to a less attractive rule (29).

$$\frac{\sum_{i\in I}\tau.S_i = \sum_{j\in J}\tau.T_j}{\sum_{i\in I}a(x).S_i = \sum_{j\in J}a(x).T_j} \tag{29}$$

A proof system for the strong open bisimilarity is given in [Sangiorgi 1996c], in which all axioms are indexed by distinctions. The system without using distinctions is proposed in [Fu and Yang 2003]. Since the following law, M5,

$$[x{\neq}y]\pi.T = [x{\neq}y]\pi.[x{\neq}y]T \tag{30}$$

is invalid in the open semantics, axiom (31) is proposed in [Fu and Yang 2003] as a substitute for (30).

$$(a)C[[x{=}a]T] = (a)C[\mathbf{0}] \tag{31}$$

It is worth remarking that (31) is equivalent to $(a)C[[x{\neq}a]T] = (a)C[T]$ in the presence of (27). So it is enough even for the π-calculus with the mismatch operator.

The first complete proof system for the weak congruence on finite π-processes is Lin's symbolic system [Lin 1995b; 1998]. The nonsymbolic systems were proposed by Parrow [1999] using (29), and by Fu and Yang [2003] using (28). In [Fu and Yang 2003] the authors have also discussed complete equational systems for the weak open congruence. It is revealed that the open bisimilarities, as well as the quasi open bisimilarities [Fu 2005], are quite complicated in the presence of the mismatch operator. It has been wrongly suggested that the complications are due to the introduction of the mismatch operator. The real culprit is actually the confusion of the names and the name variables. Study on the weak open systems pointed out that Milner's three τ-laws are insufficient [Fu and Yang 2003]. An additional τ-law

$$\tau.T = \tau.(T + \varphi\tau.T) \tag{32}$$

is necessary. Notice that (32) is derivable from (30).

All of these tiny discrepancies appear a little confusing. What the present paper suggests is that the proof system AS is the only one we need for the finite π-processes. Moreover, since the mismatch operator comes hand in hand with the match operator, there is no real interest in systems without the mismatch operator.

A more challenging issue is to construct proof systems for the regular processes [Milner 1984]. A regular process is not necessarily finite state [Milner 1989a]; it is finite control [Lin 1998]. Milner addressed the issue in the framework of CCS. His strong complete proof system [Milner 1984] and weak complete proof system [Milner 1989b] make crucial use of the following fixpoint induction rule

$$\frac{F\{E/X\} = E}{E = \mu X.F}\ X \text{ is guarded in } F. \tag{33}$$

Milner's approach has been applied to branching congruence by van Glabbeek [1993a]. It has been extended and applied to the value-passing calculi by Hennessy, Lin and Rathke [Hennessy and Lin 1997; Rathke 1997; Hennessy et al. 1997] and to the π-calculus by Lin [Lin 1995b; 1998]. All these more complicated complete proof systems are of a symbolic nature. The side condition of the fixpoint induction (33) renders the rule truly unwelcome. But as Sewell has proved in [Sewell 1994; 1997] there is no finitely axiomatizable complete system for the finite controls. In order to derive the following equality

$$\mu X.a.X = \mu X.a.\cdots.a.X$$

from a finite system, one needs rule(s) in addition to axioms. It is possible to come up with a complete proof system in which all the rules are unconditional [Sewell 1995]. But it would probably not pay off when it comes down to implementation. Now if we have to stick to the fixpoint induction, how should we make use of it? Milner's straightforward answer [Milner 1984], adopted in almost all following-up works, is to introduce process equation systems. However process equations are not that innocent. Consider the equation system

$$X = a(y).Y + G, \tag{34}$$

$$Y = b(x).X + H. \tag{35}$$

The disassociation between the right hand side of (35) and the explicit Y in (34) is not intended. All the occurrences of y in H are bound by the prefix operation in a different equation. To avoid this embarrassment one has to resort to *abstractions*, which introduces extra complexity as can be seen from the systems studied in [Hennessy et al. 1997; Lin 1998]. Completeness proofs without using any equation systems is presented in [Fu and He 2010].

The finite states/controls raise the question of divergence. The axiomatic treatment to divergence has been borrowing ideas from domain theory [Amadio and Curien 1998]. Scott's denotational approach is too abstract to give a proper account of interactions, and in the case of divergence, non-interactions. Early treatment of divergence in process algebra is more influenced by the domain theoretical approach [De Nicola and Hennessy 1984; Walker 1990]. It is our opinion that the first successful operational approach to divergence is achieved in Lohrey, D'Argenio and Hermanns' work [Lohrey et al. 2002; 2005] on the axioms for divergence. Crucial to their approach is the observation that all divergence of a finite control is due to self-looping. The Δ-operator, defined below, is isolated to play a key role in their inference systems.

$$\Delta(T) \stackrel{\text{def}}{=} \mu X.(\tau.X + T) \tag{36}$$

Lohrey, D'Argenio and Hermanns' systems consist of the laws to convert divergence from one form to another so that the fixpoint induction can be applied without any regards to divergence. One axiom proposed in [Lohrey et al. 2002; 2005] is the following axiom

$$\Delta(\Delta(T) + T') = \tau.(\Delta(T) + T'). \tag{37}$$

The law (37) is sound for the termination preserving weak congruence. But it fails to meet the codivergence property. Based on Lohrey, D'Argenio and Hermanns'

work, Fu and He discussed the axioms for the codivergence in the framework of CCS [Fu and He 2010]. The codivergent version of (37) for example is

$$\Delta(\Delta(T)) = \Delta(T). \tag{38}$$

It is worth remarking that (38) is valid for the strong equality. The proof systems using the Δ-operator begin to unveil the rich structure of divergence.

The above discussion is meant to bring out the following point. The system AS plus the axioms of codivergence proposed in [Fu and He 2010] give rise to a complete proof system for the absolute equality on the regular π-processes, which is much more accessible than the symbolic proof system. We leave it as an exercise to the reader. A related question can be asked about the box equality. The crucial step is to prove for the regular π-processes a property corresponding to the one stated in Lemma 5.10. It could turned out that axiomatization for the box equality on the regular π-processes is far more difficult than one would have perceived. The reader is referred to the work of Rensink and Vogler [2007] for a counter example showing the failure of the fixpoint induction for the fair testing equivalence.

Unlike the pure algebraic view of process [Baeten and Weijland 1990], we tend to think of AS and AS_b as proof systems that help derive process equalities. This explains the particular statements of Theorem 5.6 and of Theorem 5.11. The emphasis on equational proof systems rather than on axiomatic systems has the advantage that the introduction of congruence relations can be avoided. What is stated in Theorem 5.6 is called promotion property in [Fu and Yang 2003]. It is pointed out by Fu and Yang [2003] that this property is absolutely necessary to prove that an algebraic system of π-calculus is complete, the reason being that Hennessy Lemma [Milner 1989a] fails for the π-calculus.

6. FUTURE WORK

We have presented a comprehensive summary of the foundational theory of the name-passing calculi. The presentation follows the general principles and methodologies of Theory of Interaction developed in [Fu 2010b]. The prime motivation of Theory of Interaction can be summarized as follows:

> There are two universal relations for all models, the equality relation (absolute equality) between the programs of a model and the expressiveness relation (subbisimilarity) between the models. By using these two fundamental relations, one may introduce a number of basic postulates that formalize the foundational assumptions widely adopted in computer science. Let $\mathfrak{M}$ be the class of all models of interaction. The first postulate has to assert that $\mathfrak{M}$ is nonempty.
>
> **Axiom of Foundation**. $\mathbb{I} \in \mathfrak{M}$.
>
> One may think of $\mathbb{I}$ as the least expressive model supported by the Turing/von Neumann architecture. It lays down the physics foundation of computer science. See [Fu 2010b] for the definition of $\mathbb{I}$. A model is interesting if it subsumes $\mathbb{I}$. This leads to the next postulate.
>
> **Axiom of Completeness**. $\forall\, \mathbb{M} \in \mathfrak{M}.\ \mathbb{I} \sqsubseteq \mathbb{M}$.
>
> Axiom of Completeness can be seen as a formalization of Church-Turing Thesis. It imposes the condition that every model is a conservative

extension of the initial model $\mathbb{I}$. The two postulates already put a considerable constraints on the world $\mathfrak{M}$ of models.

In [Fu 2010b] it is shown that π, π^-, π^L and π^R are complete in the sense that they satisfy the Axiom of Completeness. These are the basic facts that justify the present paper. We now know that not all variants of the π-calculus satisfy Axiom of Completeness, and that the completeness of some other π-variants, like π^M, is still unknown. So the general framework helps to evaluate the different variants of the π-calculus and tells us which variants are the proper ones.

The model independent approach not only cleans up the syntax of the π-calculus, it also clears up the intriguing issue about the equivalence and the expressiveness relations for mobile processes. Moreover it helps to crystalize the problems for further investigation. Problem 3.14 and Problem 4.3 for example would be much easier to resolve if the branching bisimulation condition is replaced by the weak bisimulation property. In the present framework these problems can not be solved without a deeper understanding of the models. To some extent this is precisely what is needed at the moment. Our understanding of the name-passing calculi would not advance significantly if these questions remain unanswered.

The studies in process algebra over the last thirty years largely fall into two main categories:

I. The first is to do with models. A lot of process calculi have been proposed and researched [Nestmann 2006].

II. The second is about process equivalences. Many observational equivalences and algebraic equalities have been suggested, axiomatized and compared [van Glabbeek 2001; 1993b].

In the literature on process algebra there is only a small number of papers that deal with the expressiveness of process calculi [Palamidessi 2003; Gorla 2009; Fu and Lu 2010]. If we understand the situation correctly [Abramsky 2006], the biggest problem currently facing the process algebraic community is this: "where should we go from here?". It is our personal view that we should go from the process algebra, which discusses models and algebraic properties, to process theory that studies problem solving with process calculi. Results in I and II tell us which process calculi we should be concerned with and what foundational theories are available to support problem solving. What is achieved in this paper is a condensed account of the status quo of the research on the π-models. Based on these results, the π-models can be studied in three directions.

III. A theory of π-solvability that goes beyond the traditional recursion theory should be useful to understand the power of the name-passing calculi. It can be shown that a pseudo natural number generator is solvable in π. But a genuine natural number generator is π-unsolvable. It would be useful to develop a theory of π-solvability and investigate the diagonal methods for tackling π-unsolvability. Other possible issues to look at are nondeterministic functions definable in π, π-enumerators, and recursion theorem in the π-framework.

IV. A comparative study of the complexity classes defined by the π-programs against the standard complexity classes [Papadimitriou 1994] would be instructive for a better understanding of the impact of different forms of nondetermin-

ism to computation theory. Such a study may begin with a formulation of the class P_π of the problems decidable in π in polynomial time in a way similar to the definition of NP in terms of the Nondeterministic Turing Machine Model, and then investigate the relationship between P_π and NP.

V. At a more applied level, one could try to develop a hierarchy of programming languages on top of π. Previous studies in the program theory of π have focused on the object oriented feature of π and its specification capacity. A great deal more has to be learned before we are confident of using π as a target model to interpret a full fledged typed functional language.

One may question the practical significance of these new research directions. After all the π-calculus, unlike the Deterministic Turing Machine Model, does not have a physical implementation. The rapid development of computing technology has actually provided an answer. The Internet is an approximate implementation of π-calculus! It is not completely, as one might argue, a physical implementation. But it is the kind of computing environment for which a theory of π-calculus might provide just the right foundation.

ACKNOWLEDGMENTS

This work has been supported by NSFC (60873034). The authors would like to thank the members of BASICS for their interest and feedbacks. Especially they would like to thank Xiaojuan Cai for proof reading the paper and to Chaodong He for pointing out to us that our previous proof of Lemma 5.5 was mistaken (He actually demonstrated to us how to correct the mistake).

REFERENCES

Abramsky, S. 2006. What are the Fundamental Structures of Concurrency? We still do not know. Electronic Notes in Theoretical Computer Science. 37–41.

Amadio, R., Castellani, I., and Sangiorgi, D. 1996. On bisimulations for the asynchronous π calculus. In *Proc. CONCUR'96*. Lecture Notes in Computer Science, vol. 1119. Springer, 147–162.

Amadio, R. and Curien, P. 1998. Domains and lambda-calculi. *Cambridge Tracts in Theoretical Computer Science*.

Baeten, J. 1996. Branching bisimilarity is an equivalence indeed. *Information Processing Letters 58*, 141–147.

Baeten, J. and Weijland, W. 1990. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science, vol. 18. CUP.

Baier, C. and Hermanns, H. 1997. Weak bisimulation for fully probabilistic processes. In *Proc. CAV'97*. Lecture Notes in Computer Science, vol. 1254. 119–130.

Boreale, M. 1996. On the expressiveness of internal mobility in name-passing calculi. In *Proc. CONCUR'96*. Lecture Notes in Computer Science, vol. 1119. 161–178.

Boreale, M. and De Nicola, R. 1995. Testing equivalence for mobile processes. *Information and Computation 120*, 279–303.

Boreale, M., De Nicola, R., and Pugliese, R. 1999. Basic observables for processes. *Information and Computation 149*, 77–98.

Boreale, M., De Nicola, R., and Pugliese, R. 2001. Divergence in testing and readiness semantics. *Theoretical Computer Science 266*, 237–248.

Boudol, G. 1992. Asynchrony and the π-calculus. Tech. Rep. RR-1702, INRIA Sophia-Antipolis.

Brinksma, E., Rensink, A., and Vogler, W. 1995. Fair testing. In *Proc. CONCUR'95*. Lecture Notes in Computer Science, vol. 962. 313–327.

Cacciagrano, D., Corradini, F., Aranda, J., and Valencia, F. 2008. Linearity, persistence and testing semantics in the asynchronous pi-calculus. *Electronic Notes in Theoretical Computer Science 194*, 59–84.

Cacciagrano, D., Corradini, F., and Palamidessi, C. 2006. Separation of synchronous and asynchronous communication via testing. *Electronic Notes in Theoretical Computer Science 154*, 95–108.

Cai, X. and Fu, Y. 2010a. The λ-calculus in the π-calculus, `http://basics.sjtu.edu.cn/~ yuxi/papers/`.

Cai, X. and Fu, Y. 2010b. Machine models for interaction.

De Nicola, R. and Hennessy, M. 1984. Testing equivalence for processes. *Theoretical Computer Science 34*, 83–133.

De Nicola, R., Mantanari, U., and Vaandrager, F. 1990. Back and forth bisimulations. In *Proc. CONCUR'90*. Lecture Notes in Computer Science, vol. 458. 152–165.

De Nicola, R. and Vaandrager, F. 1995. Three logics for branching bisimulation. *Journal of ACM 42*, 458–487.

Fournet, C. and Gonthier, G. 1996. The reflexive chemical abstract machines and the join calculus. In *Proc. POPL'96*. ACM Press.

Fu, Y. 1999. Variations on mobile processes. *Theoretical Computer Science 221*, 327–368.

Fu, Y. 2005. On quasi open bisimulation. *Theoretical Computer Science 338*, 96–126.

Fu, Y. 2010a. Theory defined by process, `http://basics.sjtu.edu.cn/~ yuxi/papers/`.

Fu, Y. 2010b. Theory of interaction.

Fu, Y. and He, C. 2010. Axiom for codivergence.

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

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

Gorla, D. 2009. On the relative power of calculi for mobility. In *Proc. MFPS'09*. Electronic Notes in Theoretical Computer Science, vol. 249. 269–286.

Hennessy, M. 1988. *An Algebraic Theory of Processes*. MIT Press, Cambridge, MA.

Hennessy, M. 1991. A proof system for communicating processes with value-passing. *Journal of Formal Aspects of Computer Science 3*, 346–366.

Hennessy, M. and Ingólfsdóttir, A. 1993a. Communicating processes with value-passing and assignment. *Journal of Formal Aspects of Computing 5*, 432–466.

Hennessy, M. and Ingólfsdóttir, A. 1993b. A theory of communicating processes with value-passing. *Information and Computation 107*, 202–236.

Hennessy, M. and Lin, H. 1995. Symbolic bisimulations. *Theoretical Computer Science 138*, 353–369.

Hennessy, M. and Lin, H. 1996. Proof systems for message passing process algebras. *Formal Aspects of Computing 8*, 379–407.

Hennessy, M. and Lin, H. 1997. Unique fixpoint induction for message-passing process calculi. In *Proc. Computing: Australian Theory Symposium (CAT'97)*. Vol. 8. 122–131.

Hennessy, M., Lin, H., and Rathke, J. 1997. Unique fixpoint induction for message-passing process calculi. *Science of Computer Programming 41*, 241–275.

Hennessy, M. and Milner, R. 1985. Algebraic laws for nondeterminism and concurrency. *Journal of AC 32*, 137–161.

Honda, K. and Tokoro, M. 1991a. An object calculus for asynchronous communications. In *Proc. ECOOP'91*. Lecture Notes in Computer Science, vol. 512. Geneva, Switzerland, 133–147.

Honda, K. and Tokoro, M. 1991b. On asynchronous communication semantics. In *Proc. Workshop on Object-Based Concurrent Computing*. Lecture Notes in Computer Science, vol. 615. 21–51.

Honda, k. and Yoshida, M. 1995. On reduction-based process semantics. *Theoretical Computer Science 151*, 437–486.

Ingólfsdóttir, A. and Lin, H. 2001. A symbolic approach to value-passing processes. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. North-Holland, 427–478.

Lin, H. 1995a. 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*. Lecture Notes in Computer Science, vol. 915. 187–201.

Lin, H. 1995b. Unique fixpoint induction for mobile processes. In *Proc. CONCUR '95*. Lecture Notes in Computer Science, vol. 962. 88–102.

Lin, H. 1996. Symbolic transition graphs with assignment. In *Proc. CONCUR '96*. Lecture Notes in Computer Science, vol. 1119. 50–65.

Lin, H. 1998. Complete proof systems for observation congruences in finite-control π-calculus. In *Proc. ICALP '98*. Lecture Notes in Computer Science, vol. 1443. 443–454.

Lin, H. 2003. Complete inference systems for weak bisimulation equivalences in the pi-calculus. *Information and Computation 180*, 1–29.

Lohrey, M., D'Argenio, P., and Hermanns, H. 2002. Axiomatising divergence. In *Proc. ICALP 2002*. Lecture Notes in Computer Science, vol. 2380. Springer, 585–596.

Lohrey, M., D'Argenio, P., and Hermanns, H. 2005. Axiomatising divergence. *Information and Computatio 203*, 115–144.

Merro, M. 2000. Locality in the π-calculus and applications to object-oriented languages. Ph.D. thesis, Ecole des Mines de Paris.

Merro, M. and Sangiorgi, D. 2004. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science 14*, 715–767.

Milner, R. 1980. A calculus of communicating systems. *Lecture Notes in Computer Science 92*.

Milner, R. 1984. A complete inference system for a class of regular behaviours. *Journal of Computer and System Science 28*, 439–466.

Milner, R. 1989a. *Communication and Concurrency*. Prentice Hall.

Milner, R. 1989b. A complete axiomatization system for observational congruence of finite state behaviours. *Information and Computation 81*, 227–247.

Milner, R. 1993a. Elements of interaction. *Communication of the ACM 36*, 78–89.

Milner, R. 1993b. The polyadic π-calculus: a tutorial. In *Proceedings of the 1991 Marktoberdorf Summer School on Logic and Algebra of Specification*. NATO ASI, Series F. Springer-Verlag.

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

Milner, R. and Sangiorgi, D. 1992. Barbed bisimulation. In *Proc. ICALP'92*. Lecture Notes in Computer Science, vol. 623. 685–695.

Natarajan, V. and Cleaveland, R. 1995. Divergence and fair testing. In *Proc. ICALP'95*. Lecture Notes in Computer Science, vol. 944. 648–659.

Nestmann, U. 2000. What is a good encoding of guarded choices? *Information and computation 156*, 287–319.

Nestmann, U. 2006. Welcome to the jungle: A subjective guide to mobile process calculi. In *Proc. CONCUR'06*. Lecture Notes in Computer Science, vol. 4137. 52–63.

Nestmann, U. and Pierce, B. 1996. Decoding choice encodings. In *Proc. CONCUR'96*, U. Montanari and V. Sassone, Eds. Lecture Notes in Computer Science, vol. 1119. 179–194.

Palamiddessi, C., Saraswat, V., Valencia, F., and Victor, B. 2006. On the expressiveness of linearity vs. persistence in the asynchronous pi calculus. In *Proc. LICS'06*. IEEE press, 59–68.

Palamidessi, C. 2003. Comparing the expressive power of the synchronous and the asynchronous π-calculus. *Mathematical Structures in Computer Science 13*, 685–719.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley, Reading, MA.

Park, D. 1981. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science 104*, 167–183.

Parrow, J. 1999. On the relationship between two proof systems for the pi-calculus.

Parrow, J. 2001. An introduction to the π-calculus. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. North-Holland, 478–543.

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

Phillips, I. 1987. Refusal testing. *Theoretical Computer Science 50*, 241–284.

Pierce, B. and Turner, D. 2000. Pict: A programming language based on the pi calculus. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling, and M. Tofts, Eds. MIT Press.

Priese, L. 1978. On the concept of simulation in asynchronous, concurrent systems. *Progress in Cybernatics and Systems Research 7*, 85–92.

Rathke, J. 1997. Unique fixpoint induction for value-passing processes. In *Proc. LICS '97*. IEEE Press.

Rensink, A. and Vogler, W. 2007. Fair testing. *Information and Computation 205*, 125–198.

Rogers, H. 1987. *Theory of Recursive Functions and Effective Computability*. MIT Press.

Sangiorgi, D. 1992. Expressing mobility in process algebras: First order and higher order paradigm. Ph.D. thesis, Department of Computer Science, University of Edinburgh.

Sangiorgi, D. 1993. From π-calculus to higher order π-calculus–and back. In *Proc. TAPSOFT'93*. Lecture Notes in Computer Science, vol. 668. 151–166.

Sangiorgi, D. 1996a. Bisimulation for higher order process calculi. *Information and Comptation 131*, 141–178.

Sangiorgi, D. 1996b. π-calculus, internal mobility and agent-passing calculi. *Theoretical Computer Science 167*, 235–274.

Sangiorgi, D. 1996c. A theory of bisimulation for π-calculus. *Acta Informatica 3*, 69–97.

Sangiorgi, D. 2009. On the origin of bisimulation and coinduction.

Sangiorgi, D. and Milner, R. 1992. Techniques of "weak bisimulation up to". In *Proc. CONCUR'92*. Lecture Notes in Computer Science, vol. 630. 32–46.

Sangiorgi, D. and Walker, D. 2001a. On barbed equivalence in π-calculus. In *Proc. CONCUR'01*. Lecture Notes in Computer Science, vol. 2154. 292–304.

Sangiorgi, D. and Walker, D. 2001b. *The π Calculus: A Theory of Mobile Processes*. Cambridge University Press.

Sewell, P. 1994. Bisimulation is not finitely (first order) equationally axiomatisable. In *Proc. LICS'94*. IEEE. 62–70.

Sewell, P. 1995. The algebra of finite state processes. Ph.D. thesis, The University of Edinburgh.

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

Thomsen, B. 1989. A calculus of higher order communicating systems. In *Proc. POPL'89*. 143–154.

Thomsen, B. 1990. Calculi for higher order communicating systems. Ph.D. thesis, Department of Computing, University of London.

Thomsen, B. 1993. Plain *chocs*–a second generation calculus for higher order processes. *Acta Informatica 30*, 1–59.

Thomsen, B. 1995. A theory of higher order communicating systems. *Information and Computation 116*, 38–57.

van Glabbeek, R. 1990. Linear time – branching time spectrum. In *Proc. CONCUR'90*. Lecture Notes in Computer Science, vol. 458. 278–297.

van Glabbeek, R. 1993a. A complete axiomatization for branching bisimulation congruence of finite-state behaviours. In *Proc. MFCS'93*. Lecture Notes in Computer Science, vol. 711. 473–484.

van Glabbeek, R. 1993b. Linear time – branching time spectrum (ii). In *Proc. CONCUR'93*. Lecture Notes in Computer Science, vol. 715. 66–81.

van Glabbeek, R. 1994. What is branching time semantics and why to use it? In *Current Trends in Theoretical Computer Science; Entering the 21th Century*, G. Paun, G. Rozenberg, and A. Salomaa, Eds. World Scientific. 469–479.

van Glabbeek, R. 2001. Linear time – branching time spectrum (i). In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. North-Holland, 3–99.

van Glabbeek, R., Luttik, B., and Trčka, N. 2009. Branching bisimilarity with explicit divergence. *Fundamenta Informaticae 93*, 371–392.

van Glabbeek, R. and Weijland, W. 1989. Branching time and abstraction in bisimulation semantics. In *Information Processing'89*. North-Holland, 613–618.

Walker, D. 1990. Bisimulation and divergence. *Information and Computation 85*, 202–241.

Walker, D. 1991. π-calculus semantics for object-oriented programming languages. In *Proc. TACS '91*. Lecture Notes in Computer Science, vol. 526. 532–547.

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