Theory of Interaction

Yuxi Fu

BASICS, Shanghai Jiao Tong University

Abstract

Theory of Interaction aims to provide a foundational framework for computation and interaction. It proposes four fundamental principles that characterize the common features of all models of computation and interaction. These principles suffice to support a model independent treatment of the two most important relationships in computer science, the equality between processes and the relative expressiveness between models. Based on the two relationships the theory of equality, the theory of expressiveness and the theory of completeness are developed.

Keywords: Process calculus, bisimulation, interaction, computation

Contents

1. Foundation

Modern computing is all about interaction (Milner, 1993a). This is however not to say that the traditional notion of computing is not about interaction. The difference is due to the angle of observation. In Church-Turing's models of computation, the focus is on the closed systems, systems that never interact with any other systems, and the internal actions of the systems. So the models of computation are concerned with interactions within. On the other hand, the interest of the distributed computing and the mobile computing is in the open systems. In addition to its internal interactions, an open system interacts with another open system. The complexity of an open system is often caused by the interleavings of its internal interactions with its external actions and the nondeterministic timings of these interleavings.

Models of computation were proposed and studied in 1930's. Some well-known models are the Recursive Function Model, the Turing Machine Model, the λ-calculus, and the Random Access Machine (Rogers, 1987; Hopcroft and Ullman, 1979; Barendregt, 1984; van Emde Boas, 1990). By investigating the comparative power of these models, it was soon realized that all these models are equivalent in the sense that the partial functions definable in these models are all the same. The belief that all models of computation are equivalent in this sense has been referred to as Church-Turing Thesis. The original formulation of Church-Turing Thesis emphasizes on computability. It was revealed in subsequent studies, especially in the studies of algorithms (Dasgupta et al., 2006) and computational complexity (Papadimitriou, 1994; Wegener, 2005; Arora and Barak, 2009), that the equivalence of the models of computation actually holds at the operational level. There is a translation from one model of computation to another that preserves and reflects computations. The translation is not only effective, it is actually efficient.

Milner (1980) pioneered the study to formalize the notion of interaction between open systems (processes) by his work on CCS. At about the same time, Hoare (1978) has made landmark contribution to the theory in his work on the well-known programming language CSP. Since then the theory of process calculus in particular and the concurrency theory in general have proliferated. In fact the development of concurrency theory has been so fast and the number of the proposed models has increased so dramatically that a call for a general theory of interaction has long been overdue. Looking back, one cannot help remarking that the fundamental notions like interaction, composition, localization, and interface (channel) were all present in the very beginning in CSP as well as CCS. A crucial tool, bisimulation, was introduced to concurrency by Park (1981) and Milner (1989a) in 1980's, which has greatly advanced the observation theory of processes ever since. An interesting account of the history of bisimulation in computer science can be found in (Sangiorgi, 2009).

An issue that has attracted more attention than ever rencently is the relative expressiveness of process calculi. In view of the hundreds of process calculi (Nestmann, 2006), if not more, that have been proposed, results on expressiveness are rare. The issue is complicated by the lack of a consensus on the criteria for comparing the expressiveness. As Parrow (2006) has pointed out, if we have i process calculi and j sets of criteria, we end up with $i \times j \times i$ positive or negative results on expressiveness. Worse still there may well be contradictory results since an expressiveness relationship from one calculus to another may be negative by one set of criteria and positive by another set of criteria. The truth is, as long as we are using two different sets of criteria, contradictory results will never be a surprise. This is definitely unacceptable for a theory that seeks to reveal the law of interaction. One source of the diversity is that too many process calculi have been manufactured with little industrial effort. The lack of constraints has led to a profusion of calculi that would fail any single set of criteria. If we are to look for a single set of sensible criteria applicable to some models, we will be at the same time ruling out a good few of others.

Another important issue is about what Abramsky (2006) has called expressive completeness. Should we impose a minimal expressive power on all models of interaction? A positive answer would be a very foundational assumption that provides a fundamental constraint. If a process calculus is intended to be an extension of a computation model, the expressive completeness should imply that within the calculus one should be able to code up all the computable functions. So in a unified theory for both computation models and interaction models, the expressive completeness should subsume the so-called Turing completeness discussed in literature. But how should we formalize the expressive completeness? Let's take a look at how Turing completeness is formalized in the theory of process calculus. Again different criteria have been proposed in literature (Maffeis and Phillips, 2005). The proof of Turing completeness according to these criteria typically amounts to constructing a map $[\![\_]\!]$ from the set of the computing objects of a computation model $\mathbf{L}$ to a set of processes of a process calculus $\mathbf{M}$ in such a way that a computation of say L is simulated by an internal action sequence of $[\![L]\!]$; and vice versa. There is nothing wrong with most Turing completeness proofs in process calculus literature. But since these proofs are for interaction models, one should really see them from the perspective of interaction. Some of the equalities used in the proofs are not even trace equivalent. Consequently these Turing completeness results are not really useful for the interactive processes. Another conspicuous deficiency of most proofs of Turing completeness for process calculi is the lack of something like "value supply" or "result delivery". The input values are not picked up from environments properly, and the results cannot be delivered safely to any targeted receivers. Again this level of completeness cannot be promoted to interaction level.

One could give more examples demonstrating the lack of consensus, not to mention philosophies, in other parts of the theory of process calculus. But the point has been made. As much as it has benefited from the observational approach that tries to ignore all computations, the theory of process calculus has suffered from not paying enough attention to the fundamental role of the theory of computation. The theory of process calculus and the theory of computation have been two separate developments. A sensible thing to do is to carry out an integrated study that does not favour one over the other. It is expected that both theories should benefit from such an integration.

Before we formally develop the Theory of Interaction, we have to point out that the motivation for this work is mainly theoretical. We aim to lay down a framework that is as model independent as possible and at the same time allows one to prove significant general results. For that purpose we have to make some choices along the way. Firstly we will adopt the one-one synchronous communication as the basic interaction pattern. For two processes to interact they must be connected to the two ends of a channel. There are other communication patterns like one-many communication and CSP style synchronization that are not covered by our framework. Secondly we will focus exclusively on interleaving semantics. Some of our criteria do not apply in a noninterleaving semantics. Thirdly we assume that there are a couple of operators that are universal in the sense that these operators are present in all the models we will be considering. There are models that instead of having all the universal operators, have some kind of combined versions of these operators whose semantics is model specific. A study of an even more general framework that can cover models like some variants of CSP, Join Calculus, Distributed π and the Ambient Calculus (see for example (Fournet et al., 1996; Fournet and Gonthier, 1996; Hennessy, 2007; Cardelli and Gordon, 2000)) is beyond the scope of this paper.

1.1. Theory of Interaction

The prime motivation for Theory of Interaction is to bridge the gap between the computation theory and the interaction theory. By adopting the view that interaction is computation seen from within and that computation is interaction seen from without, Theory of Interaction eliminates the distinction between these two kinds of models from the outset. The benefit is that we may extend the results and methodologies well established in the computation theory to models that account for both computation and interaction.

The ultimate goal of Theory of Interaction is to provide a framework in which one may formalize the foundational assumptions, for example the Church Turing Thesis, widely accepted in the major branches of computer science. Since postulates are formulated in terms of relations, one has to pin down the fundamental relationships in computer science before formalizing any postulates. These relationships must be about models and objects (processes, programs), there is nothing more basic than these two classes of entities, hence the following two relationships.

- The first is the equality relationship on processes. At an abstract level, one cannot think of a relationship on objects/processes/programs that is simpler than the equality relationship.

- The second is the expressiveness relationship formalizing the idea that one model is at least as expressive as another. All other relationships between models are conceptually more complex.

Both the equality relationship and the expressiveness relationship must be defined in a model independent manner, otherwise there would be no way to formalize any postulates that apply to all models. Now the only way to achieve model independence is to introduce a number of principles that prescribe the common properties of all models as well as all objects. We start with these principles.

1.2. Fundamental Principle

Four fundamental principles for Theory of Interaction will be introduced. These principles introduce a set of minimal properties enough to define the two most important relationships.

1.2.1. Principle of Object

A *model of interaction* defines interactions. All interactions are performed by *processes*. An interaction is a *cooperation* between two processes. It is synchronous and atomic. A basic assumption in Theory of Interaction is that all interactions are conducted via *interfaces*.

> Through interfaces are interactions possible; no interactions go without interfaces.

This assumption, borrowed from the theory of process calculus, is ground shaping. It forces us, from the very start, to answer some basic questions about interfaces. Do we need to assume different sets of interfaces for different models of interaction? If our answer to the question is positive, we are implicitly assuming that an interface does have a mind of its own. If that was the case, why were interfaces not processes? It makes more sense to assume that interfaces are property free. This is why they are simply formalized as *names*. A name is the name of itself. It differs from another name only in that they are distinct names. Since the names are propertyless, it does not matter which set of names a model is using. Without losing any generality we adopt the following convention:

> There is only one infinite and countable set of interfaces. All models of interaction make use of this set of interfaces.

This simple assumption suggests that the interfaces are of a physical nature, whose existence is independent of any particular model.

So there are two kinds of syntactical objects in every model of interaction. In Theory of Interaction, we confine our attention to the models in which the names and the processes are the only proper objects. Other syntactical objects are either derivable or auxiliary.

> **Principle of Object**. There are two kinds of objects; they are the names and the processes.

We will apply as much as possible the same syntactical notations to all models. We will write $\mathcal{N}$ for the set of the names, ranged over by a, b, c, d, e, f, g, h. In the models we consider in this paper, an interaction happen between two processes. When two processes are connected at the two ends of an interface, an interaction can be fired. We often write a and $\overline{a}$ to indicate the "positive" and the "negative" ends of the interface a.

When defining mobility, we need *name variables.* Let $\mathcal{V}_n$ be the countable and infinite set of the name variables, ranged over by u, v, w, x, y, z. A name variable is a place holder. It can be bound by a prefix operation, in which case we say that the name variable is *bound.* A name variable is *free* if it is not bound.

The set $\mathcal{N} \cup \mathcal{V}_n$ will be ranged over by l, m, n, o, p, q. A *renaming* α is a partial map $\alpha : \mathcal{N} \rightharpoonup \mathcal{N}$ such that the domain of definition $dom(\alpha)$ is finite and that it is injective on $dom(\alpha)$. A renaming is often written explicitly as $\{b_1/a_1, \ldots, b_k/a_k\}$. A *substitution* σ is a partial map $\sigma : \mathcal{V}_n \rightharpoonup \mathcal{N} \cup \mathcal{V}_n$ whose domain of definition $dom(\sigma)$ is finite. A substitution is often written as $\{p_1/x_1, \ldots, p_k/x_k\}$. An *assignment* ρ is a partial function of type $\mathcal{V}_n \rightharpoonup \mathcal{N}$ whose domain of definition is cofinite.

For a model of interaction $\mathbb{M}$, there are syntactical objects called *terms*, or the $\mathbb{M}$-terms. The notation $\mathcal{T}_{\mathbb{M}}$ denotes the set of the $\mathbb{M}$-terms, ranged over by R, S, T. When defining higher order calculi or some form of recursion, we need *term variables.* We write $\mathcal{V}_t$ for the infinite countable set of the term variables, ranged over by X, Y, Z. A term variable can be bound by a prefix operation.

An $\mathbb{M}$-*process* is a proper $\mathbb{M}$-term. The $\mathbb{M}$-terms are introduced in order to define $\mathbb{M}$-processes. This is often necessary in a structural definition. A basic requirement for a term to be a process is that it does not contain any free variables of any kind. The notation $\mathcal{P}_{\mathbb{M}}$ stands for the set of the $\mathbb{M}$-processes, ranged over by $A, B, C, D, L, M, N, O, P, Q$.

A *term substitution* is a partial map $\varsigma : \mathcal{V}_t \rightharpoonup \mathcal{T}_{\mathbb{M}}$ whose domain of definition $dom(\varsigma)$ is finite. We often write $\{T_1/X_1, \ldots, T_k/X_k\}$ for a term substitution. A *term assignment* ϱ is a partial function of type $\mathcal{V}_t \rightharpoonup \mathcal{P}_{\mathbb{M}}$ whose domain of definition is cofinite.

We abbreviate a finite sequence of names $c_1, \ldots, c_k$ to $\widetilde{c}$. Accordingly $(c_1) \ldots (c_k)T$ is often abbreviated to $(\widetilde{c})T$. Similarly we will write for example $\widetilde{x}$, $\widetilde{X}$, $\widetilde{T}$ and $\widetilde{P}$.

The letters i, j, k will range over the set of natural numbers.

1.2.2. Principle of Action

A process either interacts with another process or performs an action on its own. The former is an *external* action whereas the latter is an *internal* action. An internal action is either a one step *deterministic* computation, or a one step *nondeterministic* computation. An external action of a process is what a counterpart sees when the two processes are engaged in an interaction. For this reason, the external actions are *observable* whereas the internal actions are *unobservable*. From the point of view of Theory of Interaction, the external actions and the internal actions carry the same weight.

> **Principle of Action**. There are two aspects of atomic actions, the internal aspect and the external aspect.

Syntactically we shall write $P \xrightarrow{\tau} P'$ to indicate that P evolves to P' after performing an internal action. The reflexive and transitive closure of $\xrightarrow{\tau}$ is denoted by $\Longrightarrow$, and the transitive closure of $\xrightarrow{\tau}$ by $\overset{\tau}{\Longrightarrow}$. Using this notation we can describe the dichotomy between deterministic and nondeterministic computations.

- We say that $P \xrightarrow{\tau} P'$ is a one step *deterministic* computation, notation $P \rightarrow P'$, if P' and P are equal.
- We say that $P \xrightarrow{\tau} P'$ is a one step *nondeterministic* computation, notation $P \xrightarrow{\iota} P'$, if P' and P are not equal.

The precise meaning of the terminologies will be clear after the equality relation is fixed. The reflexive and transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$, and the transitive closure by $\rightarrow^+$. We write $P \not\rightarrow$ if $P \rightarrow P'$ for no P'.

Suppose $\mathbb{M}$ is a model of interaction. Let $\mathcal{L}_\mathbb{M}$, ranged over by ℓ, be the set of labels representing the external actions admitted in $\mathbb{M}$. We write

$$P \xrightarrow{\ell} P'$$

for the fact that P turns into P' after performing the action ℓ. For each $\ell \in \mathcal{L}_\mathbb{M}$ let $\overline{\ell}$ be the opposite action of ℓ. Conversely ℓ is the opposite action of $\overline{\ell}$. An action and its opposite action, attached to the opposite ends of a channel, are contributions of two processes engaged in an interaction. The notation $\overset{\ell}{\twoheadrightarrow}$ is defined as follows:

$$\overset{\ell}{\twoheadrightarrow} \overset{\mathrm{def}}{=} \rightarrow^* \xrightarrow{\ell} .$$

Let $\mathcal{L}_\mathbb{M}^*$ be the set of the finite strings of the elements of $\mathcal{L}_\mathbb{M}$. We write ℓ^* for an element of $\mathcal{L}_\mathbb{M}^*$. The empty string is denoted by ϵ. If $\ell^* = \ell_1 \ldots \ell_k$, then the notation $P \overset{\ell^*}{\twoheadrightarrow} P'$ stands for $\exists P_1, \ldots, P_{k-1}.P \overset{\ell_1}{\twoheadrightarrow} P_1 \ldots \overset{\ell_{k-1}}{\twoheadrightarrow} P_{k-1} \overset{\ell_k}{\twoheadrightarrow} P'$. If $\ell^* = \epsilon$, then $P \overset{\ell^*}{\twoheadrightarrow} P'$ is the same as $P \rightarrow^* P'$.

The set of *actions* $\mathcal{A}_\mathbb{M}$ is the union set $\mathcal{L}_\mathbb{M} \cup \{\tau\}$, ranged over by λ. We say that P' is a *descendant* of P if $P \xrightarrow{\lambda_1} \ldots \xrightarrow{\lambda_k} P'$ for some $k \geq 0$. According to the definition P is a descendant of itself.

1.2.3. Principle of Observation

The only way to make an observation on a process is to interact with it. There is no alternative. If an interaction pattern between O and P is not similar to any interaction pattern between O and Q, then in the eyes of O the processes P and Q are different. In other words, O can observe a difference between P and Q. While making an observation, an observer is simultaneously being observed by the observee. It follows that the observers and the observees must be in reciprocal positions. This reciprocality has serious implications, one of which is stated as follows:

> The observers have the same observing power as the observees, no more, no less.

In a closed model of processes, where a process may interact with another process within the model but may not interact with anything outside the model, having a stronger observing power is impossible, and having a weaker observing power is insufficient.

What makes it possible for two processes to observe each other? The observation is possible if they are composed at the same level to form a bigger system. A *composition* differs from a parallel deployment precisely in that in the former the components may interact whereas in the latter no such interactions may happen. Composition and interaction come hand in hand. In terms of observation, this relationship can be phrased as follows:

> Observation is the reason for composition; composition enables observation.

The standard notation for the *composition operator* is "|". In $P \,|\, Q$ the operator "|" connects the processes P, Q, allowing them to interact at common interfaces. Systems composed of many components admit a great deal of interactions. However interactions without any control are hardly of any use. One effective approach to increase the control on interaction is to use *localization operator*. For a process P, we write $(a)P$ for the process obtained from P by restricting the use of the name a. Here the name a is localized, meaning that in $(a)P$ the interface a must be used within P and that P cannot interact with another process through the interface a. In other words, P cannot be observed at a. Investigations conducted in the theory of process calculus have shown that the localization operator is of fundamental importance (Busi and Zavattaro, 2004; Giambiagi et al., 2004; Fu and Lu, 2010).

Without the operator most calculi would be too weak to be of any interest. The motto is stated below:

> Non-observation is the reason for localization; localization disables observation.

A mainstream practice in the theory of process calculus is that the notion of observation is assumed invariant in all process calculi. If one thinks of it, this assumption is really the very basis for the expressiveness study.

> The way of making observations is the same in all models of interaction. In other words, the notion of observation is model independent.

Technically this assumption on observation forces one to adopt the following fundamental principle.

> **Principle of Observation**. There are two universal operators, the composition operator and the localization operator.

Principle of Observation implies that the semantics of the two operators are essentially the same in all models of interaction (Gorla, 2008a,b). Using the labeled transition systems (Plotkin, 1981; Aceto et al., 2001), the operational semantics of the composition operator can be defined in the standard way. There are two structural rules:

$$\frac{S \xrightarrow{\lambda} S'}{S \,|\, T \xrightarrow{\lambda} S' \,|\, T} \qquad \frac{T \xrightarrow{\lambda} T'}{S \,|\, T \xrightarrow{\lambda} S \,|\, T'} \tag{1}$$

There are semantic rules that define the cooperations between the two components of a composition. Although they cannot be completely specified at this level of abstraction, they always take the following general form.

$$\frac{S \xrightarrow{\ell} S' \qquad T \xrightarrow{\overline{\ell}} T'}{S \,|\, T \xrightarrow{\tau} R'} \tag{2}$$

The symmetric version of (2) will be always omitted. In (2) the external actions ℓ and $\overline{\ell}$ are the contributions of S and T respectively in the cooperation, and R' is defined from $S', T', \ell, \overline{\ell}$. For the localization operator, the structural rule is defined by the following rule.

$$\frac{T \xrightarrow{\lambda} T'}{(a)T \xrightarrow{\lambda} (a)T'} \; a \text{ does not appear in } \lambda. \tag{3}$$

The semantic rules for the localization operator are of the following pattern.

$$\frac{T \xrightarrow{\ell} T'}{(a)T \xrightarrow{(a)\ell} T'} \quad a \text{ does not appear in } \ell \text{ as an interface name.} \qquad (4)$$

The precise interpretation of the side condition of (4) is model dependent.

Since the composition operator is symmetric and associative, we shall use the product notation

$$\prod_{i\in\{1,..,k\}} T_i$$

for the composite term $((\ldots(T_1 \,|\, T_2) \,|\, \ldots) \,|\, T_{k-1}) \,|\, T_k$. The composition of i copies of T is denoted by

$$\prod_i T.$$

We shall say that the name a in $(a)T$ is a *local name*. A name is a *global name* if it is not local. The localization operation introduces a hierarchical structure in a term. In $(a)(S \,|\, (a)T)$ for example, S and T cannot interact at a. We write $gn(\_)$ and $ln(\_)$ for the functions that return the set of global names, and respectively the set of local names. We also write $n(\_)$ for the function that returns the set of all names.

We shall apply the *α-conversion* to both local names and bound name variables. Based on α-conversion, we shall make it a convention that no conflict and confusion about names and name variables can ever arise.

We can now describe the observations using the universal operators. For example the following internal action sequence is an observation on P.

$$(b)(O_2 \,|\, (a)(O_1 \,|\, P)) \xrightarrow{\tau} (b)(O_2 \,|\, (a)(O_1' \,|\, P')) \xrightarrow{\tau}\xrightarrow{\tau} (b)(O_2' \,|\, (a)(O_1' \,|\, P')) \ldots.$$

The process P is observed in the environment $(b)(O_2 \,|\, (a)(O_1 \,|\, \_))$. A universal definition of environment is rendered possible by the Principle of Observation.

Definition 1. An $\mathbb{M}$-*environment* $C[\_]$ is either a hole $[\_]$, or $(c)C'[\_]$, or $P \,|\, C'[\_]$, or $C'[\_] \,|\, P$, where $c \in \mathcal{N}$, $P \in \mathcal{P}_{\mathbb{M}}$ and $C'[\_]$ is an $\mathbb{M}$-environment.

1.2.4. Principle of Consistency

Theory of Interaction is *consistent* in the sense that not all processes of a model can be identified. Since the theory is meant to provide a unification framework, the equality or inequality of two processes should be judged from the point of view of both computation and interaction. In terms of the ability to perform computations, what could be the biggest difference between two processes? The difference cannot be sharper than that

- one always terminates, and
- the other may engage in an infinite sequence of internal actions.

We say that a process P is *terminating* if it does not have any infinite sequence of internal actions $P \xrightarrow{\tau}\xrightarrow{\tau} \ldots$; it is *divergent* otherwise. From the point of view of the external actions, what could be the biggest difference between two processes? The difference cannot be sharper than that

- one can interact, and
- the other cannot interact with any process.

We say that a process P is *observable*, notation $P\Downarrow$, if $P \Longrightarrow\xrightarrow{\ell} T$ for some ℓ and T; otherwise it is *unobservable*, notation $P\not\Downarrow$.

At the concurrent level of abstraction, the consistency of a model can only mean that the most different processes should never be equated.

> **Principle of Consistency**. A terminating process is never equal to a divergent process. An observable process is never equal to an unobservable process.

In Theory of Interaction all models are assumed to have the process $\mathbf{0}$ that cannot do any internal or external actions. Principle of Consistency not only points out what processes can be distinguished, but also implies what cannot be distinguished. The following statement can be seen as a corollary of the principle: All terminating unobservable processes are equal to $\mathbf{0}$. So we might as well think of a terminating unobservable process as a terminated process.

1.3. Computability Model

From the point of view of Theory of Interaction, a computation model is a simplification of an interaction model obtained by localizing all global interfaces. Different computation models can be extended to different interaction models. Among all these extensions a minimal model that embodies the fundamental idea of computability and interactability would be very useful. From our perspective an interactional extension of the computable functions may serve as such a model.

The first step to recover the interaction model from the computable functions is to recall that what made us to introduce the channels in the first place is precisely to *internalize* actions like input-a-value and output-a-result. For the unary computable function $\mathsf{f}(x)$ the intuition is that its operational semantics should describe the following three stage activities:

1. it picks up a numeral from the environment;
2. it then carries out the computation prescribed by the function; and finally
3. it delivers the result of the computation to a targeted receiver.

Depending on the type of the ambient environment, the inputs and the outputs are done with the help of particular channels. The Computability Model, the $\mathbb{C}$-calculus, is the minimal extension of the computable function model that supports (1) through (3). In $\mathbb{C}$ the atomic processes are either functions or values with the capacity to interact. The processes are generated by the following simple grammar:

$$P \;:=\; \mathbf{0} \mid \Omega \mid F_a^b(\mathsf{f}(x)) \mid \overline{a}(\underline{i}) \mid P \,|\, P,$$

where $\mathsf{f}(x)$ is a unary computable function, and $\underline{i}$ is a numeral, underlined to avoid confusion. We will write $\underline{i}, \underline{j}, \underline{k}, \underline{l}, \underline{m}, \underline{n}$ for numerals. The *functional process* $F_a^b(\mathsf{f}(x))$ at a, b, where the names a, b must be distinct, can input a numeral, say $\underline{m}$, at channel a. It becomes $\overline{b}(\underline{n})$ if $\mathsf{f}(\underline{m}) = \underline{n}$ and Ω if $\mathsf{f}(\underline{m})$ is undefined. The semantics of $F_a^b(\mathsf{f}(x))$ is given by the following rules.

$$\frac{}{F_a^b(\mathsf{f}(x)) \xrightarrow{a(\underline{m})} \overline{b}(\underline{n})}\ \text{if } \mathsf{f}(\underline{m}) = \underline{n}. \qquad \frac{}{F_a^b(\mathsf{f}(x)) \xrightarrow{a(\underline{m})} \Omega}\ \text{if } \mathsf{f}(\underline{m}) \text{ is undefined.}$$

The sole action of the *value process* $\overline{a}(\underline{i})$ at a is to release the numeral $\underline{n}$ at the channel a. The process Ω is special in that it can only diverge. These operational behaviors are defined by the following rules.

$$\frac{}{\overline{a}(\underline{n}) \xrightarrow{\overline{a}(\underline{n})} \mathbf{0}} \qquad \frac{}{\Omega \xrightarrow{\tau} \Omega}$$

The transition rule for interaction is standard:

$$\frac{P \xrightarrow{a(\underline{n})} P' \qquad Q \xrightarrow{\overline{a}(\underline{n})} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'}$$

The structural rules are also standard:

$$\frac{P \xrightarrow{\lambda} P'}{P \,|\, Q \xrightarrow{\lambda} P' \,|\, Q} \qquad \frac{Q \xrightarrow{\lambda} Q'}{P \,|\, Q \xrightarrow{\lambda} P \,|\, Q'}$$

The definition of the Computability Model is now complete. Its fundamental role in Theory of Interaction will be revealed later.

At this point the reader must wonder why $\mathbb{C}$ does not have the localization operator. According to the principles we have described, $\mathbb{C}$ as it stands is not a legitimate model of interaction in the present framework. It turns out that for $\mathbb{C}$ to fulfil its role expected of it, the localization operator is not only unnecessary, it actually has an adversary effect on $\mathbb{C}$.

2. Model of Interaction

Let's begin with an outline of how the models of interaction are defined. Suppose $\mathbb{M}$ is a model. The set $\mathcal{T}_{\mathbb{M}}$ of the $\mathbb{M}$-terms is defined by an abstract grammar in the style of BNF. It looks typically like the following:

$$T := \mathbf{0} \mid T \,|\, T' \mid (c)T \mid T_3 \mid \ldots \mid T_k.$$

Each of $\{T_3, \ldots, T_k\}$ introduces an operator. In the spirit of Theory of Interaction, we take $\mathbf{0}$, $T \,|\, T'$ and $(c)T$ as default and will omit them from the definition of an abstract grammar. The semantics of $\mathbb{M}$ is defined by a labeled transition system, which is a tuple $(\mathcal{T}_{\mathbb{M}}, \mathcal{A}_{\mathbb{M}}, \longrightarrow_{\mathbb{M}})$ where the transition relation $\longrightarrow_{\mathbb{M}}$ is a subset of $\mathcal{T}_{\mathbb{M}} \times \mathcal{A}_{\mathbb{M}} \times \mathcal{T}_{\mathbb{M}}$. We write $S \xrightarrow{\lambda}_{\mathbb{M}} T$ for $(S, \lambda, T) \in \longrightarrow_{\mathbb{M}}$. The relation $\longrightarrow_{\mathbb{M}}$, where the subscript is almost always omitted, is generated by mutual inductions on $\mathcal{T}_{\mathbb{M}}$. The inductions are given by axioms and rules and are composed of two parts:

- the axioms and rules specific to $\mathbb{M}$;
- the rules introduced in Section 1.2.3.

When defining the semantics of a model, the structural rules (1) and (3) will always be omitted. Quite often a semantic rule like (4) is also omitted as long as the interface names of the external actions have been specified. However the interaction rules like (2) will always be given.

In this section, we review three important models of interaction. They will be the running examples throughout the paper. Our formulations of the function model and the program model are not quite the same as the standard ones. Extensive studies of these two models in a manner advocated by Theory of Interaction can be found in (Fu, 2013b; Fu and Zhu, 2015).

2.1. Machine Model

We first take a look at a machine model for interaction. Such a model is promoted from the machine models for computation. An *Atomic Interactive Machine* is basically obtained from a (deterministic) Turing Machine (Hopcroft and Ullman, 1979) by replacing the input tape, the work tapes, and the output tape by input channels, accumulator and registers, and output channels respectively. It can also be seen as an extension of a Counter Machine (van Emde Boas, 1990) with input and output channels. Formally an Atomic Interactive Machine M is a tuple $(\mathsf{I}, \mathsf{O}, \mathsf{R}_0(\underline{n_0}), \mathsf{R}_1(\underline{n_1}), \mathsf{A}(\underline{n}), \mathsf{P}, j)$ where the components (see Fig. 1) have the following interpretations:

- I is the finite set of *input channels* through each of which the machine may input one numeral at a time.
- O is the finite set of *output channels* through each of which the machine may output one numeral at a time.
- $\mathsf{R}_0, \mathsf{R}_1$ are two *registers*, either of which may store a numeral. The numerals $\underline{n_0}, \underline{n_1}$ are the current values of the registers.
- A is the *accumulator* which may hold a numeral. The numeral $\underline{n}$ is the current value of the accumulator.
- P is the *program* that consists of a finite list of *instructions*. The program appears in the following shape:

$$\begin{array}{rcl} 1 & : & I_1 \\ 2 & : & I_2 \\ & \vdots & \\ k & : & I_k \\ k+1 & : & \mathit{Stop} \end{array}$$

  where $k \geq 0$. Here $1, \ldots, k+1$ are the *addresses* of the instructions. The last instruction of the program must be *Stop*. For each $i \in \{1, \ldots, k+1\}$, we write $\mathsf{P}(i)$ for the i-th instruction of P.

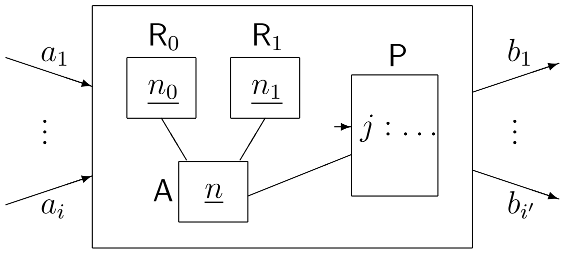

Figure 1: Atomic Interactive Machine.

- j is the address of the *current* instruction, taking value in $\{1, \dots, k+1\}$.

The instructions are classified into two groups, those that move data around between machines, and those that manipulate data within a machine. The instructions in the first group are of two types:

(i) *Input* a, where $a \in \mathsf{I}$, picks up a numeral at channel a and update the content of A by the received numeral.

(o) *Output* b, where $b \in \mathsf{O}$, fetches the numeral stored in A and delivers the numeral through channel b.

The instructions in the second group are of three types. Only one instruction is of the first type.

(s) *Stop* is the instruction that terminates the machine execution.

Two instructions are of the second type.

(r) *Read* i, where i is either 0 or 1, copies the content of R_i to A.

(w) *Write* i, where i is either 0 or 1, copies the content of A to R_i.

The instructions of the third type define arithmetic operations. There are many choices for these instructions. The reader might have his/her favorite combination. In this paper we shall not be specific about these instructions.

The semantics of the Atomic Interactive Machines is given by a labeled transition system. The rules about the input and output instructions are as follows.

$$\frac{\mathsf{P}(j) = \mathit{Input}\ a}{(\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j) \xrightarrow{a(m)} (\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{m}),\mathsf{P},j+1)}$$

$$\frac{\mathsf{P}(j) = \mathit{Output}\ b}{(\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j) \xrightarrow{\overline{b}(n)} (\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j+1)}$$

The rules about the read and write instructions are as follows:

$$\frac{\mathsf{P}(j) = \mathit{Read}\ i \text{ and } i \in \{0,1\}}{(\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j) \xrightarrow{\tau} (\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n_i}),\mathsf{P},j+1)}$$

$$\frac{\mathsf{P}(j) = \mathit{Write}\ 0}{(\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j) \xrightarrow{\tau} (\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j+1)}$$

$$\frac{\mathsf{P}(j) = \mathit{Write}\ 1}{(\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n_0}),\mathsf{R}_1(\underline{n_1}),\mathsf{A}(\underline{n}),\mathsf{P},j) \xrightarrow{\tau} (\mathsf{I},\mathsf{O},\mathsf{R}_0(\underline{n}_0),\mathsf{R}_1(\underline{n}),\mathsf{A}(\underline{n}),\mathsf{P},j+1)}$$

The Atomic Interactive Machines are so defined to facilitate exchange of information among individual machines. The general *Interactive Machines* are syntactically defined by the following grammar:

$$\mathsf{M} := \mathsf{AIM} \mid \mathsf{M} \,|\, \mathsf{M}' \mid (c)\mathsf{M}$$

where AIM's are the Atomic Interactive Machines. Let $\mathbb{IM}$ denote the Interactive Machine Model. The semantics of $\mathbb{IM}$ follows the general methodology. The interaction rule for example is defined by the following rule and its symmetric version.

$$\frac{\mathsf{M} \xrightarrow{a(n)} \mathsf{M}' \qquad \mathsf{N} \xrightarrow{\overline{a}(n)} \mathsf{N}'}{\mathsf{M} \,|\, \mathsf{N} \xrightarrow{\tau} \mathsf{M}' \,|\, \mathsf{N}'}$$

The machine model is convenient for model theoretical studies of distributed systems. In distributed computing a major concern is how interactions are implemented. Insightful discussions on this issue can be found in the pioneering work of Hoare (1978) and of Milner (1989a).

2.2. Function Model

The value-passing calculus $\mathbb{VPC}$ is the recursion theory (Rogers, 1987; Soare, 1987) reincarnated in an interactive framework. Let's pause for a minute and think about what need be introduced to define all the recursive functions. The followings ought to be clear.

1. To admit the input actions input terms "$a(x).T$" are introduced.
2. To support the output actions output terms "$\overline{a}(\_).T$" are incorporated.
3. To define minimization, conditional terms "$\mathit{if}\,\varphi\ \mathit{then}\,T$" are necessary.
4. To interpret recursion, replication is a minimal option.

The model $\mathbb{VPC}$ is defined on top of Presburger Arithmetic (Fu, 2013b). We write $\mathsf{PA} \vdash \varphi$ if φ can be derived in Presburger Arithmetic. Formally the set $\mathcal{T}_{\mathbb{VPC}}$ of the $\mathbb{VPC}$-terms is generated by the following grammar.

$$T \ := \ \sum_{1\le i\le k} a(x).T_i \mid \sum_{1\le i\le k} \overline{a}(t_i).T_i \mid \mathit{if}\,\varphi\ \mathit{then}\,T \mid !a(x).T \mid !\overline{a}(t).T.$$

In the input choice $\sum_{1\le i\le k} a(x).T_i$ and the output choice $\sum_{1\le i\le k} \overline{a}(t_i).T_i$, the guard $a(x)$ is an input prefix and the guard $\overline{a}(t_i)$ is an output prefix. The input prefix $a(x)$ binds the term variable x in the term it applies. A variable is free if it is not bound. The set $\mathcal{P}_{\mathbb{VPC}}$ of the $\mathbb{VPC}$-processes consists of those $\mathbb{VPC}$-terms that do not contain any free variables. The term $\mathit{if}\,\varphi\ \mathit{then}\,T$ is a conditional term in which φ is a boolean expression. Often we write $[\varphi]T$ as a syntactical abbreviation for $\mathit{if}\,\varphi\ \mathit{then}\,T$. A finite set of boolean expressions $\{\varphi_i\}_{i\in I}$ is mutually exclusive if $\mathsf{PA} \vdash (\varphi_i \wedge \varphi_j \Rightarrow \bot)$ for all $i, j \in I$ such that $i \neq j$. Given such a family, one could define the conditional choice $\sum_{i\in I}[\varphi_i]T_i$ as follows:

$$\sum_{i\in I}[\varphi_i]T_i \ \stackrel{\mathrm{def}}{=}\ \prod_{i\in I}[\varphi_i]T_i.$$

A special case of the conditional choice is the following *if_then_else_* term:

$$\mathit{if}\,\varphi\ \mathit{then}\,S\ \mathit{else}\,T \ \stackrel{\mathrm{def}}{=}\ [\varphi]S \mid [\neg\varphi]T.$$

It is obvious that at most one of S, T can be fired. For the same reason, at most one summand of $\sum_{i\in I}[\varphi_i]T_i$ may proceed further.

Using the label set $\mathcal{L}_{\mathbb{VPC}} = \{a(\underline{i}), \overline{a}(\underline{i}) \mid a \in \mathcal{N}, \text{ and } i \text{ is a natural number}\}$, the semantics of $\mathbb{VPC}$ can be defined by the following labeled transition system, where j ranges over $\{1, \ldots, k\}$ in the action rules.

Action

$$\frac{}{\sum_{1\le i\le k} a(x).T_i \xrightarrow{a(\underline{n})} T_j\{\underline{n}/x\}} \qquad \frac{}{\sum_{1\le i\le k} \overline{a}(t_i).T_i \xrightarrow{\overline{a}(\underline{n})} T_j} \ \mathsf{PA} \vdash t_j = \underline{n}.$$

Interaction

$$\frac{S \xrightarrow{a(\underline{n})} S' \qquad T \xrightarrow{\overline{a}(\underline{n})} T'}{S \,|\, T \xrightarrow{\tau} S' \,|\, T'}$$

Condition

$$\frac{T \xrightarrow{\lambda} T'}{\mathit{if}\, \varphi\ \mathit{then}\, T \xrightarrow{\lambda} T'} \ \mathsf{PA} \vdash \varphi.$$

Recursion

$$\frac{}{!a(x).T \xrightarrow{a(\underline{n})} T\{\underline{n}/x\} \,|\, !a(x).T} \qquad \frac{}{!\overline{a}(t).T \xrightarrow{\overline{a}(\underline{n})} T \,|\, !\overline{a}(t).T} \ \mathsf{PA} \vdash t = \underline{n}.$$

In many occasions the variant of $\mathbb{VPC}$ with the input and the output prefixes in place of the guarded choices is sufficient. We write $\mathbb{VPC}^-$ for this variant. Our treatment of $\mathbb{VPC}$ is more formal than the ones found in literature. A detailed exposure of our approach can be found in (Fu, 2013b).

2.3. Program Model

Church's λ-calculus (Barendregt, 1984) has played a key role both in mathematical logic and in understanding the operational issues concerning (functional) programming. The lazy λ-calculus (Abramsky, 1988) is the variant whose operational semantics is purely sequential. What makes the λ-calculus so successful in modeling programming is its use of a basic operator, the functional application, and its exclusive focus on functions. This is very much like in the axiomatic set theory, widely accepted as a foundation of mathematics, where one has the membership relation and the sets. When moving from functional computation to concurrent computation, a natural thing to do is to take concurrent composition operator as basic and retain some form of function. It appears at first sight that the interactive model of the lazy λ-calculus is straightforward. Its syntax could be defined by the following grammar:

$$T := X \mid a(X).T \mid \overline{a}\langle T\rangle.T.$$

Unfortunately the model so defined is not a legal citizen in Theory of Interaction. The problem is that, unlike in the λ-calculus, a process has little control over what it receives through interaction. See Section 5.2.2 for more technical explanation. A paradigm shift from the functional scenario to the object oriented scenario was made by Milner et al. (1992). Instead of passing around processes, the π-processes send and receive names. The set $\mathcal{T}_\pi$ of the π-terms is generated by the following grammar:

$$T := \sum_{1\le i\le k} n(x).T_i \mid \sum_{1\le i\le k} \overline{n}m_i.T_i \mid [p{=}q]T \mid [p{\neq}q]T \mid !n(x).T \mid !\overline{n}m.T.$$

The match operator $[\_ = \_]$ and the mismatch operator $[\_ \neq \_]$ are extremely important in both theory and practice. They are independent and conservative over the π-calculus without the conditionals (Fu and Lu, 2010). The name variable x in $n(x).T$ is bound. The set $\mathcal{P}_\pi$ of the π-processes consists of those π-terms that contain no free name variables. The label set $\mathcal{L}_\pi$ is $\{ac, \overline{a}c, \overline{a}(c) \mid a, c \in \mathcal{N}\}$. The semantics of the π-calculus is defined by the following rules, where $[k] = \{1, \ldots, k\}$.

Action

$$\frac{}{\sum_{1\le i\le k} a(x).T_i \xrightarrow{ac} T_j\{c/x\}}\ j\in[k] \qquad \frac{}{\sum_{1\le i\le k} \overline{a}m_i.T_i \xrightarrow{\overline{a}c} T_j}\ m_j = c,\ j\in[k]$$

Interaction

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

Condition

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

Recursion

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

Some variants of the π-calculus can be obtained by imposing additional syntactical restrictions. Let π^- be obtained from the π-calculus by replacing the guarded choice operators by the plain prefix operator. If the match/mismatch operator is further removed from π^-, we get π^M, the minimal π-calculus. For a comprehensive theory of the π-calculus and its variants, the reader is referred to the satellite paper by Fu and Zhu (2015).

3. Theory of Equality

Theory of equality studies observational equalities of processes. In line with the spirit of Theory of Interaction, we shall be focusing exclusively on model independent equalities. Conceivably there are a number of choices. What we are looking for is *the* equality "=" on processes. Our strategy to uncover the definition of this equality is to derive several corollaries from the four foundational principles. The properties stated in these corollaries are minimal in the sense that none of them can be properly weakened. We then turn these minimal properties into defining properties. What we get is the absolute equality. Here the word "absolute" refers to minimality, model independence, and uniqueness. The success of Theory of Interaction depends on the fact that if we strengthen or weaken the definition of the absolute equality, we would get an equivalence subject to criticism.

Before proceeding ahead, a review of some standard terminologies is in order. A (binary) relation $\mathcal{R}$ on $\mathbb{M}$ is a subset of $\mathcal{P}_{\mathbb{M}} \times \mathcal{P}_{\mathbb{M}}$. It is reflexive if $\forall P \in \mathcal{P}_{\mathbb{M}}.(P,P) \in \mathcal{R}$, symmetric if $(P,Q) \in \mathcal{R}$ implies $(Q,P) \in \mathcal{R}$, and transitive if $(N,P) \in \mathcal{R}$ and $(P,Q) \in \mathcal{R}$ imply $(N,Q) \in \mathcal{R}$. We will often use the infix notation $P\mathcal{R}Q$ for $(P,Q) \in \mathcal{R}$. Let $\mathcal{R}^i$ be the composition of i copies of $\mathcal{R}$ with $\mathcal{R}^0$ being the identity relation. The reflexive and transitive closure $\bigcup_{i\in\omega} \mathcal{R}^i$ of $\mathcal{R}$ is denoted by $\mathcal{R}^*$. A relation $\mathcal{S}$ from $\mathbb{M}$ to $\mathbb{M}'$ is a subset of $\mathcal{P}_{\mathbb{M}} \times \mathcal{P}_{\mathbb{M}'}$. It is *total* if $\forall P \in \mathcal{P}_{\mathbb{M}}.\exists Q \in \mathcal{P}_{\mathbb{M}'}.(P,Q) \in \mathcal{S}$. Given a relation $\mathcal{S}$, the reverse relation is denoted by $\mathcal{S}^{-1}$. The composition $\{(x,z) \mid \exists y.x\mathcal{S}_0 y \wedge y\mathcal{S}_1 z\}$ is denoted by $\mathcal{S}_0;\mathcal{S}_1$, or even by $\mathcal{S}_0\mathcal{S}_1$.

Definition 2. A relation $\mathcal{R}$ on $\mathbb{M}$ is *closed* if the following are valid:

1. For each $a \in \mathcal{N}$, $(a)P \; \mathcal{R} \; (a)Q$ whenever $P\mathcal{R}Q$.
2. For each $O \in \mathcal{P}_{\mathbb{M}}$, $O \,|\, P \; \mathcal{R} \; O \,|\, Q$ and $P \,|\, O \; \mathcal{R} \; Q \,|\, O$ whenever $P\mathcal{R}Q$.

3.1. Equality for Evolving Object

Computations are carried out over time. Interactions are conducted in space. The equalities for *self evolving* and *interacting* objects must span in both time and space. They should never be refuted by any computation or interaction. Time, space, computation, and interaction are all we have in mind when looking for *the* equality on processes.

3.1.1. Time Invariance

The equality $=$ on the processes must take into account of the dynamic self evolutions of processes. What if

$$P = Q \Longrightarrow Q'$$

for possibly an infinite number of distinct Q'? If for some Q'' there does not exist any P' such that

$$P \Longrightarrow P' = Q''$$

then the process Q might silently evolve into some state to which P has no matching state. If such situations may occur, how can $=$ even be considered an equality in the first place? The point is that, when left alone, processes evolve by themselves over time. The self evolutions can be neither controlled nor detected. If the equality $P = Q$ holds right now, it should be possible that the equality is maintained at any point in future. Moreover the maintenance is done in a way that the history of the equality can be traced when going backwards in time. In Theory of Interaction the slogan is this:

> Equal objects have been equal in history and will be equal in future.

It is not trivial to formalize this proposition for self evolving objects. In our opinion the next definition, introduced by van Glabbeek and Weijland (1989), captures precisely the future aspect of the above slogan. It is a refinement of the weak bisimulation, confer Definition 12.

Definition 3. 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:

   (i) $Q \Longrightarrow Q'$ for some Q' such that $Q'\mathcal{R}^{-1}P$ and $Q'\mathcal{R}^{-1}P'$.

(ii) $Q \Longrightarrow Q''\mathcal{R}^{-1}P$ for some Q'' such that $\exists Q'.Q'' \xrightarrow{\tau} Q'\mathcal{R}^{-1}P'$.

2. If $P\mathcal{R}Q \xrightarrow{\tau} Q'$ then one of the following statements is valid:

   (i) $P \Longrightarrow P'$ for some P' such that $P'\mathcal{R}Q$ and $P'\mathcal{R}Q'$.

   (ii) $P \Longrightarrow P''\mathcal{R}Q$ for some P'' such that $\exists P'.P'' \xrightarrow{\tau} P'\mathcal{R}Q'$.

It is important that the bisimulation property is stated in terms of internal actions. External actions are model dependent and consequently cannot be explicitly referred to in any model independent definition.

3.1.2. Space Invariance

According to the Principle of Observation, if M is equal to N and P is equal to Q, then the result of M observing P should not be different from the result of N observing Q. But what does it mean that two observations are not different? The only possible interpretation is that the result of L observing $M \,|\, P$ should not be different from the result of O observing $N \,|\, Q$ whenever L is equal to O. In other words, $M \,|\, P$ and $N \,|\, Q$ must be equal. Now if no process may ever observe any difference between P and Q, then no process that does not make use of the name a can ever observe any difference between P and Q. This is equivalent to saying that $(a)P$ and $(a)Q$ are equal. Contrapositively if a process can observe a difference between $(a)P$ and $(a)Q$, then it can also detect the difference between P and Q since the difference cannot involve a. To conclude, the equality for the processes must be closed under both composition and localization. It is in this sense that the equality for processes spans in space.

Definition 4. A binary relation $\mathcal{R}$ is *extensional* if the following extensionality property holds:

1. If $M\mathcal{R}N$ and $P\mathcal{R}Q$ then $(M \,|\, P)\ \mathcal{R}\ (N \,|\, Q)$;
2. If $P\mathcal{R}Q$ then $(a)P\ \mathcal{R}\ (a)Q$ for every $a \in \mathcal{N}$.

The Principle of Observation guarantees that extensionality is a model independent property. It will become clear that condition (2) of Definition 4 is indispensable. Without it we would not be able to distinguish between $!\tau \,|\, !a \,|\, b$ and $!\tau \,|\, !a \,|\, c$ in a model independent way, bearing in mind that the external actions are model dependent. The relationship between the extensional relations and the closed relations is pointed out in the following lemma.

Lemma 3.1. *The following statements are valid:*

1. *If $\mathcal{R}$ is reflexive and extensional, then $\mathcal{R}$ is closed.*
2. *If $\mathcal{R}$ is closed, then $\mathcal{R}^*$ is extensional.*

When reasoning about process equality, it is often necessary to construct the extensional closure operation on a relation. It is therefore convenient to make available the following definition.

Definition 5. The *extensional closure* $\mathcal{R}^\circ$ of a binary relation $\mathcal{R}$ is inductively defined as follows:

$$\begin{aligned}
\mathcal{R}_0 &\stackrel{\text{def}}{=} \mathcal{R} \\
&\vdots \\
\mathcal{R}_{i+1} &\stackrel{\text{def}}{=} \mathcal{R}_i \cup \left\{ \begin{array}{l|l} ((a)P,(a)Q) & a \in \mathcal{N} \text{ and } P\mathcal{R}_i Q \\ (L \mid M, N \mid O) & L\mathcal{R}_i N \text{ and } M\mathcal{R}_i O \end{array} \right\} \\
&\vdots \\
\mathcal{R}^\circ &\stackrel{\text{def}}{=} \bigcup_{i\in\omega} \mathcal{R}_i
\end{aligned}$$

Clearly a binary relation $\mathcal{R}$ is extensional if and only if $\mathcal{R} = \mathcal{R}^\circ$.

3.1.3. Computation Invariance

The equality $=$ on the processes must also answer to the question on divergence. What if

$$P = Q \xrightarrow{\tau} Q_1 \xrightarrow{\tau} Q_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_i \xrightarrow{\tau} Q_{i+1} \xrightarrow{\tau} \ldots$$

where Q can continue to do internal actions ad infinitum? If P always eventually interacts with some environment, then intuitively it is not completely equivalent to Q since an infinite internal action sequence can preempt any interactions with any environments. A standard solution to the divergence problem is to impose the divergence preservation condition which requires that P is divergent if and only if Q is divergent. But the condition does not seem to fit well with the idea of bisimulation. Let's see an example. Suppose A is the CCS process $(c)((\tau.c.good+\tau) \mid !\overline{c}.(\tau.c.good+\tau))$. The internal choice $\tau.c.good + \tau$ can be defined by $(a)((a.c.good + a) \mid \overline{a})$. More often than not, the process $A \mid \mu X.\tau.X$ is thought to be equivalent to A. Both are divergent.

There is however good reasons why they should not be equated. A divergent computation of $A \,|\, \mu X.\tau.X$ may never produce anything good. On the other hand, A keeps on producing something good after every two consecutive internal actions. An improvement on the divergence preservation condition requires that an infinite internal action sequence of Q is *bisimulated* by an infinite internal action sequence of P; and vice versa. This leads to the next definition that captures part of the Principle of Consistency.

Definition 6. A binary relation $\mathcal{R}$ is *codivergent* if the following codivergence property holds whenever $P\mathcal{R}Q$:

1. If $P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_{i+1} \xrightarrow{\tau} \ldots$ is an infinite internal action sequence then $Q \overset{\tau}{\Longrightarrow} Q'\mathcal{R}^{-1}P_i$ for some Q' and some $i \geq 1$;
2. If $Q \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} Q_{i+1} \xrightarrow{\tau} \ldots$ is an infinite internal action sequence then $P \overset{\tau}{\Longrightarrow} P'\mathcal{R}Q_i$ for some P' and some $i \geq 1$.

It is obvious that a codivergent relation is divergence preserving.

3.1.4. Interaction Invariance

The equality $=$ on processes must also take into account of the dynamic interactions between processes and environments. If $P = Q$ then P and Q should exert similar influence on, or inflict comparable damage to an environment. Since we are interested in the minimal properties the equality has to satisfy, we may choose to settle on the following property: If $P = Q$ and one of P, Q can interact with an environment then the other can interact with an environment as well. We say that P and Q are *equipollent* if $P\Downarrow \Leftrightarrow Q\Downarrow$.

Definition 7. A binary relation $\mathcal{R}$ is *equipollent* if P and Q are equipollent whenever $P\mathcal{R}Q$.

From the viewpoint of model independence, there is no way to strengthen the equipollence condition. From the point of view of observation, there does not seem to be any room to weaken the condition. It ought to be just right.

3.2. Absolute Equality

The definitions of bisimulation, extensionality, codivergence and equipollence are given without any reference to any model. We are now turning these conditions into the defining properties of the equality. Before doing that, the following technical lemma is necessary.

Lemma 3.2. *If $\{\mathcal{R}_i\}_{i\in I}$ is a family of reflexive, equipollent, extensional, codivergent bisimulations on $\mathbb{M}$, then $(\bigcup_{i\in I}\mathcal{R}_i)^*$ is a reflexive, equipollent, extensional, codivergent bisimulation on $\mathbb{M}$.*

PROOF. The bisimulation property is closed under the relational composition. See (Baeten, 1996) for the subtlety of this point. □

It follows that the largest reflexive, equipollent, codivergent, extensional bisimulation of every model of interaction exists, hence the next definition.

Definition 8. The *absolute equality* $=_{\mathbb{M}}$ of $\mathbb{M}$ is the largest relation on $\mathcal{P}_{\mathbb{M}}$ that validates the following statements:

1. The relation is reflexive.
2. The relation is equipollent, extensional, codivergent and bisimilar.

An alternative to Definition 8 is given in the next lemma.

Lemma 3.3. *The absolute equality $=_{\mathbb{M}}$ coincides with the largest equipollent, codivergent, closed bisimulation on $\mathcal{P}_{\mathbb{M}}$.*

Since the definition of $=_{\mathbb{M}}$ is model independent, we often omit the subscript.

We can now make precise the definition of the deterministic computations and that of the nondeterministic computations.

- We say that $P \xrightarrow{\tau} P'$ is a one step deterministic computation, notation $P \rightarrow P'$, if $P' = P$.

- We say that $P \xrightarrow{\tau} P'$ is a one step nondeterministic computation, notation $P \xrightarrow{\iota} P'$, if $P' \neq P$.

If $P \xrightarrow{\iota} P'$ then the internal action changes the communication capacity of P or the (in)ability of P to diverge. By definition if $P = Q$ and P changes the state to P' in a single step, meaning $P \xrightarrow{\iota} P'$, then Q may evolve to Q'' via a finite sequence of internal actions that do not change the state, i.e. $Q \rightarrow^* Q''$, and then changes the state from Q'' to Q' in a single step, that is $Q'' \xrightarrow{\iota} Q'$, in order to match $P \xrightarrow{\iota} P'$.

Using Lemma 3.3 it is easy to derive the following fundamental property of the absolute equality.

Lemma 3.4. *If $P \Longrightarrow = Q$ and $Q \Longrightarrow = P$ then $P = Q$.*

Proof. Suppose $P \equiv P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_k = Q$ and $Q \equiv Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} Q_{k'} = P$. Let $\mathcal{R}$ be the relation

$$\{(P_i, Q_j) \mid 0 \le i \le k \wedge 0 \le j \le k'\} \cup = .$$

It is routine to check that $\mathcal{R}^\circ$ is a reflexive, equipollent, extensional, codivergent bisimulation. □

We shall refer to Lemma 3.4 as *Bisimulation Lemma*. As far as we know, the property described in Lemma 3.4 was discovered by De Nicola et al. (1990), who called it X-property. It is an extremely general result, valid for all the observational equivalences one may think of.

A simple yet fundamental property about computation is stated next.

Lemma 3.5. *If $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_k = P_0$, then $P_0 \to P_1 \to \dots \to P_k$.*

Proof. Suppose $1 \le i \le j \le k$. Clearly $P_i \Longrightarrow P_j = P_j$ and $P_j \Longrightarrow P_k \Longrightarrow P_i' = P_i$ for some P_i'. It follows from Lemma 3.4 that $P_i = P_j$. □

Lemma 3.5 will be referred to as *Computation Lemma*. It was discovered by van Glabbeek and Weijland (1989), who termed it Stuttering Lemma. The significance of this lemma lies in that it reveals how systems evolve. All internal action sequences are of the form

$$P_0 \to^* P_0' \xrightarrow{\iota} P_1 \to^* P_1' \xrightarrow{\iota} \dots \xrightarrow{\iota} P_i \to^* P_i' \xrightarrow{\iota} P_{i+1} \to^* \dots . \quad (5)$$

After P_0' has made a change-of-state move to P_1, there is no turn-back. No later state can ever be equal to P_0'. In other words, systems evolve in stages. Once a system has reached to a new stage, it will never go back to any of its previous stages unless there is an intervention from environment. Now if $Q_0 = P_0$ then Q_0 has to simulate (5) by an internal action sequence of the following shape

$$Q_0 \to^* Q_0' \xrightarrow{\iota} Q_1 \to^* Q_1' \xrightarrow{\iota} \dots \xrightarrow{\iota} Q_i \to^* Q_i' \xrightarrow{\iota} Q_{i+1} \to^* \dots \quad (6)$$

such that $Q_1 = P_1, \dots, Q_i = P_i, \dots$. The two internal action sequences (5) and (6) enjoy another interesting property. The corresponding pair P_i, Q_i not only bisimulate into the future, they also bisimulate backwards to history.

This *back and forth* bisimulation property was pointed out by De Nicola et al. (1990).

We tend to think that Computation Lemma describes an intrinsic property about self evolving systems; it is more about computation than about system equivalence.

Computation Lemma also points out that as far as the absolute equality is concerned, the codivergence condition is equivalent to the following statement in terms of deterministic computation:

> If $P_0 = Q_0$ and $P_0 \to P_1 \to \ldots \to P_k \to \ldots$ is an infinite sequence of deterministic computation, then there exists an infinite sequence of deterministic computation $Q_0 \to Q_1 \to \ldots \to Q_k \to \ldots$.

So codivergence is about classical computations as it should be. The termination preservation property should be understood as saying that if $P = Q$ and P can do deterministic computation ad infinitum, then Q can do deterministic computation ad infinitum; and vice versa.

To demonstrate the power of the absolute equality we take a look at it in the Computability Model. We remark that since the $\mathbb{C}$-calculus disowns the localization operator, the condition (2) of Definition 4 is vacuously met. To begin with we introduce a congruence relation on the $\mathbb{C}$-processes.

Definition 9. The structural congruence $\equiv_{\mathbb{C}}$ is the least equivalent and congruent relation satisfying the following equalities:

1. $\mathbf{0} \,|\, P \equiv_{\mathbb{C}} P$; $P \,|\, Q \equiv_{\mathbb{C}} Q \,|\, P$; $(P \,|\, Q) \,|\, R \equiv_{\mathbb{C}} P \,|\, (Q \,|\, R)$;
2. $\Omega \,|\, \Omega \equiv_{\mathbb{C}} \Omega$.

The next result says that $=_{\mathbb{C}}$ is almost the syntactical equality.

Theorem 3.6. *$P =_{\mathbb{C}} Q$ if and only if $P \equiv_{\mathbb{C}} Q$.*

Proof. The main steps of the proof can be structured as follows:

(a) Every $\mathbb{C}$-process must be of the form

$$\prod_{i\in I} F_{a_i}^{b_i}(\mathsf{h}_i(x)) \,|\, \prod_{j\in J} \overline{c_j}(\underline{m_j}) \,|\, \prod_{k\in K} \Omega \,|\, \prod_{l\in L} \mathbf{0} \qquad (7)$$

up to the associativity and commutativity of the composition operator. We say that $F_{a_i}^{b_i}(\mathsf{h}_i(x))$ is a *functional component* and $\overline{c_j}(\underline{m_j})$ is a *value component* at c_j.

(b) For each $\mathbb{C}$-process A there is some $\mathbb{C}$-process $\overline{A}$ such that $A \,|\, \overline{A} \Longrightarrow U$ for some unobservable $\mathbb{C}$-process U.

(c) If $P \xrightarrow{\tau} P'$ then the number of the value components of P' is no more than that of P.

(d) For numerals $\underline{n}, \underline{m}$, let $\mathsf{f}_{\underline{n}\to\underline{m}}(x)$ be defined as follows:

$$\mathsf{f}_{\underline{n}\to\underline{m}}(x) \overset{\text{def}}{=} \textit{if}\; x = \underline{n} \;\textit{then}\; \underline{m} \;\textit{else}\; \text{diverge}. \tag{8}$$

Suppose f does not appear in $A \,|\, B$. Now $A \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) \,|\, \overline{a}(\underline{n+1}) \xrightarrow{\tau} A \,|\, \Omega$. According to (b) there is some $\overline{A}$ such that f does not appear in $\overline{A}$ and

$$\overline{A} \,|\, A \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) \,|\, \overline{a}(\underline{n+1}) \overset{\tau}{\Longrightarrow} = \Omega. \tag{9}$$

Now $A \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) \neq B \,|\, \overline{f}(\underline{m})$ follows from (9) and the fact that C is observable whenever $\overline{A} \,|\, B \,|\, \overline{f}(\underline{m}) \,|\, \overline{a}(\underline{n+1}) \Longrightarrow C$.

(e) Suppose f does not appear in $A \,|\, B$ and $A \,|\, \overline{f}(\underline{n}) = B \,|\, \overline{f}(\underline{n})$. Let g be a name that does not appear in $A \,|\, B$. Then $A \,|\, \overline{f}(\underline{n}) \,|\, F_f^g(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) \xrightarrow{\tau} A \,|\, \overline{g}(\underline{m})$ must be bisimulated by $B \,|\, \overline{f}(\underline{n}) \,|\, F_f^g(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) \overset{\tau}{\Longrightarrow} B' \,|\, \overline{g}(\underline{m})$ due to the property proved in (d). So $B \,|\, \overline{g}(\underline{m}) \overset{\tau}{\Longrightarrow} B' \,|\, \overline{g}(\underline{m}) = A \,|\, \overline{g}(\underline{m})$. By symmetry and the Bisimulation Lemma we conclude that $A \,|\, \overline{g}(\underline{m}) = B \,|\, \overline{g}(\underline{m})$.

(f) Let $\mathcal{R}$ be the following relation

$$\left\{ (A, B) \;\middle|\; \begin{array}{l} \text{the name } f \text{ does not appear in } A \,|\, B, \\ \text{and } A \,|\, \overline{f}(\underline{n}) = B \,|\, \overline{f}(\underline{n}). \end{array} \right\}.$$

The property proved in (e) implies that $\mathcal{R}$ is closed under composition. It is easily seen that it is also equipollent, codivergent and bisimilar. We conclude that if f does not appear in $A \,|\, B$ and $A \,|\, \overline{f}(\underline{n}) = B \,|\, \overline{f}(\underline{n})$ then $A = B$.

(g) Suppose $A \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{m}}(x)) = B \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{m}}(x))$ and f does not appear in $A \,|\, B$. It is an easy consequence of (d) that $A \,|\, \overline{f}(\underline{m}) = B \,|\, \overline{f}(\underline{m})$, hence $A = B$ by (f).

(h) Suppose $P \to^* P_1$ and P_1 may not perform any computation. Let $\overline{a}(\underline{n})$ be a value component of P and f be a name that does not appear in P. The action $P \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{n}}(x)) \xrightarrow{\tau} P' \,|\, \overline{f}(\underline{n})$ must be bisimulated by

$$P_1 \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{n}}(x)) \to^* P_1'' \,|\, F_a^f(\mathsf{f}_{\underline{n}\to\underline{n}}(x)) \xrightarrow{\tau} P_1' \,|\, \overline{f}(\underline{n})$$

by (d). According to (g) one has that $P_1 \rightarrow^* P_1''$. And by assumption P_1'' must be P_1. So $\overline{a}(\underline{n})$ is also a value component of P_1. In the light of the property stated in (c) we conclude that P and P_1 have the same *multi-set* of the value components.

(i) Suppose $P =_{\mathbb{C}} Q$, $P \rightarrow^* P_1$, $Q \rightarrow^* Q_1$ and neither P_1 nor Q_1 may perform any computation. Using the idea of (h), we can show that P_1 and Q_1 have the same multi-set of the value components. Consequently P and Q have the same *multi-set* of the value components.

(j) A consequence of (i) is that $P \neq_{\mathbb{C}} P'$ whenever $P \xrightarrow{\tau} P'$. This is because in every function process $F_a^b(\mathsf{f}(x))$ the names a, b are distinct.

(k) Now suppose $P =_{\mathbb{C}} Q$ and the functional components of P are

$$F_{a_1}^{b_1}(\mathsf{h}_1(x)), \ldots, F_{a_k}^{b_k}(\mathsf{h}_k(x)).$$

Let $\underline{m_1}$ be a numeral. Then the action

$$P \,|\, \overline{a_1}(\underline{m_1}) \xrightarrow{\tau} P_1 \tag{10}$$

must be bisimulated by

$$Q \,|\, \overline{a_1}(\underline{m_1}) \xrightarrow{\tau} Q_1. \tag{11}$$

According to (i) the value component $\overline{a_1}(\underline{m_1})$ must be consumed in the action of (11). It follows that the number of the functional components of Q is no less than k. By symmetry we conclude that P, Q must have the same *number* of the functional components.

(l) It also follows from (f), (i) and (j) that if $P =_{\mathbb{C}} Q$ and P', Q' are obtained from P, Q by removing the value components then $P' =_{\mathbb{C}} Q'$.

(m) Suppose $P =_{\mathbb{C}} Q$ and P, Q do not have any value components. Let $F_a^b(\mathsf{f}(x))$ be a functional component of P. If for every functional component of Q that is of the form $F_a^b(\mathsf{g}(x))$ the computable function $\mathsf{g}(x)$ is not the same as $\mathsf{f}(x)$, then we may force a contradiction by composing with P, Q a numeral of output $\mathbb{C}$-processes of the form $\overline{a}(\underline{n})$. So $F_a^b(\mathsf{f}(x))$ must be a functional component of Q. By symmetry and (k) we conclude that P, Q must have the same *multi-set* of the functional components.

The codivergence property takes care of the Ω components. $\square$

Having seen Theorem 3.6, one wonders what if the localization operator is admitted to $\mathbb{C}$. Let $\mathbb{A}$ be obtained from $\mathbb{C}$ by adding the localization operator whose semantics is defined by the rule (3). The following proposition tells us that the equality becomes very different in $\mathbb{A}$.

Proposition 3.7. *Suppose $I_a = (b)(F_a^b(\mathsf{f}) \mid F_b^a(\mathsf{f}^{-1}))$, where f is a total computable function. Then $I_a \mid I_a =_{\mathbb{A}} I_a$.*

Proof. It is clear that $(b)(\overline{b}(\mathsf{f}(\underline{i})) \mid F_b^a(\mathsf{f}^{-1})) =_{\mathbb{A}} (b)(F_b^a(\mathsf{f}) \mid \overline{b}(\mathsf{f}^{-1}(\underline{i}))) =_{\mathbb{A}} \overline{a}(\underline{i})$ and $(b)(\mathbf{0} \mid \overline{a}(\underline{i})) =_{\mathbb{A}} (b)(\overline{a}(\underline{i}) \mid \mathbf{0}) =_{\mathbb{A}} \overline{a}(\underline{i})$ for every numeral $\underline{i}$. Let $\mathcal{R}$ be the following relation

$$\{(C[I_a \mid I_a], C[I_a]) \mid C[\_] \text{ is an environment}\}.$$

It is easy to see that $\mathcal{R} \cup \mathcal{R}^{-1} \cup =_{\mathbb{A}}$ is reflexive, equipollent, extensional, codivergent and bisimilar. □

The equality stated in the above proposition is often associated with the asynchronous π-calculus (Honda and Tokoro, 1991a,b; Boudol, 1992), which is a variant of π where output primitive does not have any continuation. It is well-known that this variant of the π-calculus enjoys some very different algebraic laws (Amadio et al., 1998; Sangiorgi, 1996b; Boreale, 1996; Merro, 2000; Merro and Sangiorgi, 2004; Fu, 2010), among which the equality stated in Proposition 3.7 is typical.

We remark that $\mathbb{C}$ does not have the problem for the simple reason that b must be distinct from a for every functional process at a, b. Had we admitted functional process of the form $F_a^a(\mathsf{f})$, we would have $F_a^a(\mathsf{f}_{id}) \mid F_a^a(\mathsf{f}_{id}) =_{\mathbb{C}} F_a^a(\mathsf{f}_{id})$, where f_{id} denotes the identity function. This is an equality that does not translate to most models. As this example indicates, one has to be very careful about the definition of $\mathbb{C}$.

3.3. Below and Above the Absolute Equality

We will provide further evidence that the absolute equality is the only equality for *both* computation and interaction. We do that by taking a look at two modifications of the absolute equality. At the present level of abstraction, there is little room for any refinement on the equipollence, codivergence and extensionality conditions. Moreover since these conditions are proposed as minimal conditions, there is no way to weaken any of them either. Only the bisimulation condition is subject to modification.

Firstly let's think for a while how the bisimulation property can be strengthened. It is not difficult to see that there is really only one sensible way to do that.

Definition 10. A binary relation $\mathcal{R}$ is a *strong bisimulation* if the following strong bisimulation property holds.

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

Clearly the strong bisimulation property subsumes the codivergence property. To keep it consistent with Definition 8, we introduce the following.

Definition 11. The *strong equality* $\sim_{\mathbb{M}}$ is the largest reflexive, equipollent, extensional, strong bisimulation on $\mathcal{P}_{\mathbb{M}}$.

For the familiar process calculi, strong equality $\sim_{\mathbb{M}}$ is essentially the strong bisimilarity of Park (1981) and Milner (1989a). It follows from definition that $\sim_{\mathbb{M}} \subseteq =_{\mathbb{M}}$. The strong equality is useful in constructing proof systems (Hennessy and Milner, 1985) and in the use of bisimulation-up-to technique (Sangiorgi and Milner, 1992). However from the point of view of the observation theory the requirement that one computation step of a process must be simulated by one computation step of an equal process has serious negative consequences. Church-Turing Thesis would fail under this strong interpretation.

How about weakening the definition of bisimulation? In literature the delay bisimulation (Milner, 1981) and the η-bisimulation (Baeten and van Glabbeek, 1987) have been proposed as weaker forms of bisimulation. The definitions of these bisimulations must refer to external actions, which is against our model independent philosophy. The least modification of bisimulation is in our view the weak bisimulation of Milner (1989a).

Definition 12. A binary relation $\mathcal{R}$ is a *weak bisimulation* if the following *weak bisimulation property* holds.

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'.

Like the definition of bisimulation Definition 12 does not give rise to a useful relation if it is not combined with additional properties. The weak equality given in the next definition is stronger than the weak bisimilarity of Milner in that the former takes divergence into account.

Definition 13. The *weak equality* $=_w^{\mathbb{M}}$ is the largest reflexive, equipollent, extensional, codivergent weak bisimulation on $\mathcal{P}_{\mathbb{M}}$.

Clearly $=_{\mathbb{M}} \subseteq =_w^{\mathbb{M}}$. A well-known equality valid for $=_w^{\mathbb{M}}$ is the so called Milner's third τ-law, a consequence of which is the following equality

$$\tau.(a{+}\tau.b) + \tau.c \;=_w^{\mathbb{M}}\; \tau.(a{+}\tau.b) + \tau.b + \tau.c. \tag{12}$$

Equality (12) is invalid for the absolute equality. The right hand side of (12) has an internal action of the form $\tau.(a{+}\tau.b) + \tau.b + \tau.c \xrightarrow{\tau} b$. The only way to simulate such an internal action by the left hand is $\tau.(a{+}\tau.b) + \tau.c \xrightarrow{\tau} a{+}\tau.b \xrightarrow{\tau} b$. However according to the picture given in (5) and (6), these two internal action sequences should never be identified since a single change-of-state action $\tau.(a{+}\tau.b) + \tau.b + \tau.c \xrightarrow{\tau} b$ is quite different from two consecutive change-of-state actions $\tau.(a{+}\tau.b) + \tau.c \xrightarrow{\tau} a{+}\tau.b \xrightarrow{\tau} b$. For another example, consider the π-processes M, A, B defined as follows:

$$\begin{aligned} M &\overset{\text{def}}{=} \mu X.a(x).[x{=}c](\tau.X + \tau.\overline{x}x), \\ A &\overset{\text{def}}{=} !M, \\ B &\overset{\text{def}}{=} !M \,|\, a(x).[x{=}c]\overline{x}x. \end{aligned}$$

It should be clear that $A =_w^{\pi} B$ but not $A =_{\pi} B$. The problem of the weak equality is that it wrongfully confuses the deterministic computations with the nondeterministic ones. In theory of computation the distinction between these two kinds of computation is never compromised.

We have looked at two nearest cousins of the absolute equality. Both are rejectable from the point of view of computation.

3.4. Respectful Operator

An expected criticism to the absolute equality $=_{\mathbb{M}}$ is that it is not necessarily closed under all the $\mathbb{M}$-operators. Congruence property is so important for an equality that it is tempting to introduce the following definition.

> The algebraic $\mathbb{M}$-equality is the largest reflexive, equipollent, codivergent bisimulation on $\mathcal{P}_{\mathbb{M}}$ closed under all the $\mathbb{M}$-operators.

One could argue that the above definition is just as model independent as Definition 8. Another popular way of enforcing the algebraic property is through the following definition.

P and Q are $\mathbb{M}$-congruent if the equality $P =_{\mathbb{M}} Q$ is closed under all the $\mathbb{M}$-operators.

From the point of view of observation theory there are good reasons to reject both definitions. Several arguments are given below.

1. If a unary operator *op* does not preserve the equality $P =_{\mathbb{M}} Q$, then both the algebraic $\mathbb{M}$-equality and the $\mathbb{M}$-congruence are strictly contained in the absolute equality $=_{\mathbb{M}}$. In other words both equalities introduce additional distinguishing power that cannot be achieved through observation/interaction; they are not really observational equivalence.
2. The requirement that the equality be closed under whatever operator one has in mind is not justifiable. In an interactive framework, one simply cannot grab two pieces of *running* programs by brutal force, put them in a local testing platform, and run the test. Testing for the self evolving processes means that the only thing the testers can do is to interact with the testees, which implies that the testers are the environments in the sense of Definition 1. In the π-calculus for instance a context like $a(x).\_$ should not be considered a legitimate environment.
3. Algebraic property is not something that can be observed. An observer cannot observe the choice operator of $a.b + b.a$ since the observee may well be $a \,|\, b$. The point is that algebraic property has nothing to do with the observation theory. It has everything to do with the semantic definition of the operators. As long as the semantics is right, algebraicity comes for free. Instead of taking algebraicity as a property, we should instead take it as a criterion.

A famous misbehaved operator is the unguarded choice. In order to make a congruence out of the bisimulation equivalence in the presence of this operator, one has to reject $P = \tau.P$. This is ridiculous since the equality between P and $\tau.P$ should never be rejected by any *observational* equality. What should really be rejected is the unguarded choice. The guarded version should always be preferred (Nestmann and Pierce, 1996; Fu and Lu, 2010). Another operator that ought to be rejected is the unrestrained replication. It is clear that $\mathbf{0} = \tau.\mathbf{0}$ but not $!\mathbf{0} = !\tau.\mathbf{0}$. The latter violates the codivergence condition. In retrospect the replicator should have been introduced in its guarded form in the first place. There is really no need for the unrestrained replications since the guarded replications are just as expressive as the unguarded ones.

An operator of $\mathbb{M}$ is said to be *respectful* if it respects the absolute equality. Most of the operators introduced in the theory of process calculus are indeed respectful. In the current paper we outlaw any operators that are not respectful.

In Theory of Interaction the absolute equality is a congruence.

So we have imposed another constraint on the models we shall consider.

3.5. Observation Theory

We now inspect the absolute equality in the models defined in Section 2. It turns out that in each of these models, the absolute equality of two processes can be established using a more tractable method. By such an approach the external actions are explicitly bisimulated. In the cases of $\mathbb{VPC}$ and π, if the codivergence condition is dropped, the explicit characterizations are the branching bisimilarities of van Glabbeek and Weijland (1989). If moreover the bisimulation is replaced by the weak bisimulation, the explicit characterizations are precisely the weak bisimilarities.

In the following definition and the theorem the model $\mathbb{M}$ can be understood as any of the four models $\mathbb{C}$, $\mathbb{IM}$, $\mathbb{VPC}$, π.

Definition 14. An $\mathbb{M}$-*bisimulation* $\mathcal{R}$ is a codivergent bisimulation on $\mathcal{P}_{\mathbb{M}}$ that validates the following statements whenever $P\mathcal{R}Q$.

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

The $\mathbb{M}$-*bisimilarity* $\approx_{\mathbb{M}}$ is the largest $\mathbb{M}$-bisimulation.

We say that (1) and (2) of the above definition provide an external characterization of the absolute equality $=_{\mathbb{M}}$. We often refer to $\approx_{\mathbb{M}}$ as the *external bisimilarity* for $\mathbb{M}$. The next theorem provides a strong justification for introducing $\mathbb{M}$-bisimilarities.

Theorem 3.8. *The following statements are valid.*

1. *The external bisimilarity* $\approx_{\mathbb{VPC}}$ *coincides with* $=_{\mathbb{VPC}}$.
2. *The external bisimilarity* $\approx_{\pi}$ *coincides with* $=_{\pi}$.

Proof. First of all we remark that both external bisimilarities are congruence. So the inclusions in one direction are easy.

The value carried by an output action of a $\mathbb{VPC}$-process can always be recognized unambiguously by some environments. If say $P =_{\mathbb{VPC}} Q$ and $P \xrightarrow{\overline{a}(\underline{5})} P'$, then it is easy to derive $Q \overset{\overline{a}(\underline{5})}{\twoheadrightarrow} Q' =_{\mathbb{VPC}} P'$ for some Q' using the environment $a(x).if\ x = \underline{5}\ then\ c(z)$, where c appears neither in P nor in Q. For input actions one needs to use environments of the form $\overline{a}(\underline{k}).f + \overline{a}(\underline{k}).g$.

In the case of the π-calculus the crucial observation is that

$$\left\{ (P,Q) \;\middle|\; \begin{array}{l} (a_1,..,a_k)(\overline{c_1}a_1 \,|\, \ldots \,|\, \overline{c_k}a_k \,|\, P) =_\pi \\ (a_1,..,a_k)(\overline{c_1}a_1 \,|\, \ldots \,|\, \overline{c_k}a_k \,|\, Q), \\ \{c_1,\ldots,c_k\} \cap gn(P \,|\, Q) = \emptyset,\ k \geq 0 \end{array} \right\}$$

is a π-bisimulation up to $\sim_\pi$. For the complete proof, the reader is referred to (Fu and Zhu, 2015). □

For the Computability Model it would really be disturbing if the external bisimilarity gives rise to an equality different from the absolute equality. Fortunately the coincidence is an immediate consequence of Theorem 3.6.

Corollary 3.9. *The external bisimilarity* $\approx_{\mathbb{C}}$ *coincides with* $=_{\mathbb{C}}$.

Theorem 3.8 and Corollary 3.9 are among a few coincidence results that we know of. However in any event the inclusion $\approx_{\mathbb{M}} \subsetneq =_{\mathbb{M}}$ is valid, which is all that matters when using $\approx_{\mathbb{M}}$ to prove process equality.

The external characterization of the absolute equality $=_{\mathbb{M}}$ of $\mathbb{M}$ is the basis for the *observation theory* of $\mathbb{M}$.

4. Theory of Expressiveness

Theory of expressiveness ought to be the most important part of the concurrency theory. It remains however one of the least investigated area in the theory. The lack of a theory of expressiveness has impeded the studies into several important issues. One such study is about operator independence. Given a model $\mathbb{M}$, the operator independence problem asks if an operator of $\mathbb{M}$ can be expressed by the remaining operators of $\mathbb{M}$. Fu and Lu (2010) have shown that all the operators of the π-calculus are independent. This is achieved by studying the expressiveness of every sub-model of the π-calculus obtained by removing one of its operators. Results of this kind are very rare. At the moment we are not even able to say for example that the operators of FA (Fu, 2007) are independent to each other. Another branch of investigation heavily depends on the theory of expressiveness is the expressive completeness (Abramsky, 2006). There have been several studies on the problem of Turing completeness of process calculi. But by and large the results obtained so far are not very conclusive. The lack of a widely acceptable theory of expressiveness is mainly to blame.

The central issue of the theory of expressiveness is to uncover the definition of expressiveness. Expressiveness is a relative concept. When we are stating an expressiveness result about a model, we are comparing it against some other model(s). It goes without saying that the "being-more-expressive" relationship has to be transitive, and necessarily model independent. An expressive result is often obtained by providing a structural translation, also called an encoding. In this paper we shall maintain a distinction between interpretations and translations/encodings.

- An *interpretation* of $\mathbb{M}_0$ into $\mathbb{M}_1$ is a total relation from $\mathcal{P}_{\mathbb{M}_0}$ to $\mathcal{P}_{\mathbb{M}_1}$. This relation interprets a process of $\mathbb{M}_0$ by a set of processes of $\mathbb{M}_1$.
- A *translation/encoding* from $\mathbb{M}_0$ to $\mathbb{M}_1$ is an effective function from $\mathcal{P}_{\mathbb{M}_0}$ to $\mathcal{P}_{\mathbb{M}_1}$. A translation/encoding gives rise to an interpretation by composing it with the absolute equality $=_{\mathbb{M}_1}$ of the target model.

An encoding is often defined by structural induction.

In this section we provide a basic approach to the theory of expressiveness. The philosophy of the approach is that the relative expressiveness relationship must be a generalization of the absolute equality. It is simply inconsistent to apply an expressiveness criterion that is weaker or stronger than the equality to which the expressiveness criterion must refer to.

4.1. Subbisimilarity

An interpretation from $\mathbb{M}_0$ to $\mathbb{M}_1$ that formalizes the idea of $\mathbb{M}_1$ being at least as expressive as $\mathbb{M}_0$ associates to a process P of $\mathbb{M}_0$ a set of $\mathbb{M}_1$-processes that are equal to P as it were. Such a relation is equipollent, extensional, codivergent and bisimilar. The reflexivity turns into totality plus soundness. We now motivate the soundness requirement.

There are two dual aspects of the expressive power, the distinguishing power and the control power. They are like the two sides of a coin. If $\mathbb{M}_1$ is more expressive than $\mathbb{M}_0$, then $\mathbb{M}_1$ should have more distinguishing power than $\mathbb{M}_0$. Distinguishing power is about the capacity to interact. It speaks about the ability of observers. At the same time, if $\mathbb{M}_1$ is more expressive than $\mathbb{M}_0$, then $\mathbb{M}_1$ should also have more control power than $\mathbb{M}_0$. Control power is about the capacity not to interact. It pronounces the ability of observees.

Now suppose $\mathcal{T}$ is an interpretation from $\mathbb{M}_0$ to $\mathbb{M}_1$ indicating that the latter is at least as expressive as the former. If $\mathbb{M}_1$ cannot distinguish two processes of $\mathbb{M}_0$ under the interpretation $\mathcal{T}$, then according to the above discussion $\mathbb{M}_0$ cannot distinguish the two processes.

Definition 15. The interpretation $\mathcal{T}$ is *complete* if $\mathcal{T}; =_{\mathbb{M}_1}; \mathcal{T}^{-1} \subseteq =_{\mathbb{M}_0}$.

Complementarily if two processes of $\mathbb{M}_0$ have equal control power to defeat all attempts to distinguish them, then their interpretations under $\mathcal{T}$ should have equal control power to remain indistinguishable.

Definition 16. The interpretation is *sound* if $\mathcal{T}^{-1}; =_{\mathbb{M}_0}; \mathcal{T} \subseteq =_{\mathbb{M}_1}$.

We remark that the existence of a universal equality, the absolute equality, is essential to both completeness and soundness. An interpretation relating two different equivalences should not be perceived as a relative expressiveness result. Another point is that, since the absolute equality is a special interpretation and an interpretation is a generalization of the absolute equality, the conditions $\mathcal{T}; =_{\mathbb{M}_1}; \mathcal{T}^{-1} \subseteq =_{\mathbb{M}_0}$ and $\mathcal{T}^{-1}; =_{\mathbb{M}_0}; \mathcal{T} \subseteq =_{\mathbb{M}_1}$ are nothing but reflexivity from the point of view of the source model and the target model respectively. This seems to be a good reason for the next definition.

Definition 17. An interpretation $\mathcal{T}$ from $\mathbb{M}_0$ to $\mathbb{M}_1$ is *fully abstract* if it is both sound and complete.

Following the previous remark, we would like to emphasize the role of the full abstraction by pointing out the following relationship:

Full abstraction is a generalization of reflexivity.

It is easy to see that $\mathcal{T}; =_{\mathbb{M}_1}; \mathcal{T}^{-1} \subseteq =_{\mathbb{M}_0}$ if $\mathcal{T}$ is total, equipollent, extensional, codivergent and bisimilar. This is why in the following model independent definition, only totality and soundness are required.

Definition 18. A relation $\mathfrak{R}$ from $\mathbb{M}_0$ to $\mathbb{M}_1$ is a *subbisimilarity*, notation $\mathfrak{R} : \mathbb{M}_0 \rightarrow \mathbb{M}_1$, if it validates the following statements.

1. The relation is total and sound.
2. The relation is equipollent, extensional, codivergent and bisimilar.

The condition (1) of Definition 18 corresponds to the condition (1) of Definition 8. It makes sure that the following proposition is valid.

Proposition 4.1. *Subbisimilarities are fully abstract.*

We say that $\mathbb{M}_0$ is *subbisimilar* to $\mathbb{M}_1$, notation $\mathbb{M}_0 \sqsubseteq \mathbb{M}_1$, if there is a subbisimilarity from $\mathbb{M}_0$ to $\mathbb{M}_1$. We write $\mathbb{M}_0 \sqsubset \mathbb{M}_1$ if $\mathbb{M}_0 \sqsubseteq \mathbb{M}_1$ but $\mathbb{M}_1 \not\sqsubseteq \mathbb{M}_0$. The proposition $\mathbb{M}_0 \sqsubseteq \mathbb{M}_1$ can now be reiterated as follows: $\mathbb{M}_1$ is at least as expressive as $\mathbb{M}_0$ if for each $\mathbb{M}_0$-process P there exists an $\mathbb{M}_1$-process Q that is equal to P.

Proposition 4.2. *Both $\sqsubseteq$ and $\sqsubset$ are transitive.*

The next simple fact will be used without further reference.

Lemma 4.3. *Suppose $\mathfrak{S}$ is a subbisimilarity. If $P\mathfrak{S}Q \not\rightarrow$ then $P \rightarrow^* P'\mathfrak{S}Q$ for some P' such that $P' \not\rightarrow$. Conversely if $Q\mathfrak{S}^{-1}P \not\rightarrow$ then $Q \rightarrow^* Q'\mathfrak{S}^{-1}P$ for some Q' such that $Q' \not\rightarrow$.*

We can now apply the subbisimilarity tool to the prime models defined in Section 2. The first result assures that the machine model is more elementary than the value passing calculi.

Theorem 4.4. $\mathbb{IM} \sqsubset \mathbb{VPC}$.

The negative result $\mathbb{VPC} \not\sqsubseteq \mathbb{IM}$ is due to the fact that every Interactive Machine M is essentially of the form $(\widetilde{c})(\mathsf{M}_1 \,|\, \ldots \,|\, \mathsf{M}_k)$ where $\mathsf{M}_1, \ldots, \mathsf{M}_k$ are Atomic Interactive Machines. This machine may perform at most a fixed number of immediate actions at a time. It follows that no Interactive Machines can simulate a $\mathbb{VPC}$ process, say $!a(x).\overline{b}(x)$, which may produce more and more immediate interactive capabilities when it evolves. The negative result does not depend on the output choice operator of $\mathbb{VPC}$.

Corollary 4.5. $\mathbb{VPC}^- \not\sqsubseteq \mathbb{IM}$.

More examples will be given in the following subsections.

The existence of a unique expressiveness relationship helps to formalize many semantic concepts based on expressiveness. Let's see two examples. Suppose *op* is an operator of $\mathbb{M}$. Let $\mathbb{M}^{-op}$ be the model obtained from $\mathbb{M}$ by removing *op*. The independence of *op* asks if the addition of *op* to $\mathbb{M}^{-op}$ changes the expressive power of $\mathbb{M}^{-op}$.

Definition 19. An operator *op* of $\mathbb{M}$ is *independent* if $\mathbb{M} \not\sqsubseteq \mathbb{M}^{-op}$.

Normally we are only concerned with the independence for the operators other than the universal operators. The following fact is established in (Fu and Lu, 2010).

Fact 4.6. *The operators of the π-calculus are all independent.*

A related issue asks if an extension is faithful.

Definition 20. An operator *op* of $\mathbb{M}$ is *conservative* if the identity map from $\mathbb{M}^{-op}$ to $\mathbb{M}$ is a subbisimilarity.

Clearly an operator *op* of $\mathbb{M}$ is conservative if the identity map from $\mathbb{M}^{-op}$ to $\mathbb{M}$ is sound. An explicit characterization of $=_{\mathbb{M}}$ in terms of the external bisimilarity is helpful in deciding if an extension is conservative. The following is an example (Fu and Lu, 2010).

Fact 4.7. *Both the match operator and the mismatch operator of the π-calculus are conservative.*

4.2. Soundness and Relative Expressiveness

A relative expressiveness result, say $\mathbb{M} \sqsubseteq \mathbb{N}$, asserts that $\mathbb{M}$ is a submodel of $\mathbb{N}$; in other words, the former can be faithfully represented in the latter. Here soundness plays an indispensable role. To get a feeling of the subtlety of the soundness, we take a look at two translations that violate the soundness condition. These exercises will demonstrate that a lot of liberal encodings would be admitted if the soundness condition is dropped.

The first counter example is about a self-translation of the π-calculus. Let $[\![\_]\!]^{\bowtie}$ be a function from $\mathcal{P}_\pi$ to $\mathcal{P}_\pi$ that is structural on $\mathbf{0}$, composition, localization, replication, match and mismatch. Its definition on the prefix terms is given by the following clauses.

$$\begin{array}{rcl}[\![n(x).T]\!]^{\bowtie} & \stackrel{\text{def}}{=} & \overline{n}(c).c(x).[\![T]\!]^{\bowtie}, \\ [\![\overline{n}m.T]\!]^{\bowtie} & \stackrel{\text{def}}{=} & n(z).\overline{z}m.[\![T]\!]^{\bowtie}, \quad \text{where } z \text{ is fresh.}\end{array}$$

It is shown in Appendix A that the translation satisfies all but the soundness condition of a subbisimilarity. As it turns out, there is no nontrivial interpretation of the π-calculus into itself, see Theorem 4.12.

The second example is more subtle. Suppose we intend to substantiate our vague intuition that the π-calculus is more powerful than $\mathbb{VPC}$. After some careful research, we come up with an encoding of $\mathbb{VPC}$ into the π-calculus. It consists of four parts:

1. an encoding of the numerals by the π-processes;
2. an encoding of the boolean expressions by the π-processes;
3. an encoding of the term expressions by the π-processes; and
4. an encoding of the $\mathbb{VPC}$-processes by the π-processes.

The encoding is defined in detail in Appendix B. It looks very good at first sight. But again the soundness fails. Theorem 4.9 will confirm that there is no subbisimilarity from $\mathbb{VPC}$ to π.

4.3. Incompatibility of VPC and Pi

We report in this section the expressiveness results obtained by applying the subbisimilarity tool to $\mathbb{VPC}$ and π. The first result confirms that the object oriented programming style admitted by the π-calculus is very different from the functional programming style.

Proposition 4.8. *$\pi \not\sqsubseteq \mathbb{VPC}$.*

Proof. It suffices to show that the π-process $a(x).\overline{x}$ cannot be simulated by any $\mathbb{VPC}$-process. Assume that there were a subbisimilarity $\mathfrak{F}$ from π to $\mathbb{VPC}$. Let A and A_b be such that $a(x).\overline{x}$ $\mathfrak{F}$ A and $\overline{a}b$ $\mathfrak{F}$ A_b. It is easy to see that after A_b has done an external action at the name a it becomes some process equal to $\mathbf{0}$. Now A may contain only a finite number of global names. So its interactions with the interpretations of $\overline{a}b_1, \ldots, \overline{a}b_k, \ldots$ cannot be all correct. $\square$

The second result reveals that the value-passing mechanism of the functional programming is beyond the communication capacity of the object oriented programming.

Theorem 4.9. *$\mathbb{VPC} \not\sqsubseteq \pi$.*

Proof. Assume that there were a subbisimilarity $\mathfrak{G}$ from $\mathbb{VPC}$ to π. For each name a and each numeral $\underline{i}$, let $\lceil \underline{i} \rfloor_a \not\twoheadrightarrow$ be a chosen interpretation of $\overline{a}(\underline{i})$ under $\mathfrak{G}$ and $A \not\twoheadrightarrow$ be a chosen interpretation of $a(x).\overline{b}(x)$ by $\mathfrak{G}$. Before proving the theorem, we need to derive a number of necessary properties about $\lceil \underline{i} \rfloor_a$ and A.

1. By the codivergence condition there is no infinite internal action sequence from $\lceil \underline{i} \rfloor_a \,|\, A$. According to König Lemma there are only a finite number of internal action sequences from $\lceil \underline{i} \rfloor_a \,|\, A$. Every internal action sequence of $\lceil \underline{i} \rfloor_a \,|\, A$ terminates in a process equal to $\lceil \underline{i} \rfloor_b$.
2. If $\lceil \underline{i} \rfloor_a \,|\, A \xrightarrow{\tau} B$ then $\lceil \underline{i} \rfloor_a \,|\, A \neq B$. Observe that $\lceil \underline{i} \rfloor_a \,|\, A \,|\, \lceil \underline{i} \rfloor_a \,|\, A$ would have an infinite computation sequence if $\lceil \underline{i} \rfloor_a \,|\, A \rightarrow B$ for some B.
3. It follows from the previous fact that the action $\overline{a}(\underline{i}) \,|\, a(x).\overline{b}(x) \xrightarrow{\iota}= \overline{b}(x)$ must be matched up by every immediate internal action of $\lceil \underline{i} \rfloor_a \,|\, A$ and that $\lceil \underline{i} \rfloor_a \,|\, A \xrightarrow{\iota} B =_\pi \lceil \underline{i} \rfloor_b$ whenever $\lceil \underline{i} \rfloor_a \,|\, A \xrightarrow{\tau} B$.
4. The process $\lceil \underline{i} \rfloor_a$ may perform either an input action at a or an output action at a. It cannot perform both an input action and an output action. Otherwise one would have $\lceil \underline{i} \rfloor_a \,|\, \lceil \underline{i} \rfloor_a \xrightarrow{\tau} M$ for some M, which is impossible for the following reasons: (i) $\lceil \underline{i} \rfloor_a \,|\, \lceil \underline{i} \rfloor_a \xrightarrow{\iota} M$ is impossible since $\overline{a}(\underline{i}) \,|\, \overline{a}(\underline{i})$ cannot do any computation at all; and (ii) $\lceil \underline{i} \rfloor_a \,|\, \lceil \underline{i} \rfloor_a \rightarrow M$ is also impossible because that would induce an infinite computation sequence from $\lceil \underline{i} \rfloor_a \,|\, \lceil \underline{i} \rfloor_a$.
5. If $\lceil \underline{i} \rfloor_a$ can perform an input (output) action at a, then for every j the process $\lceil \underline{j} \rfloor_a$ can perform an input (output) action because the latter needs to interact with A.
6. If $\lceil \underline{i} \rfloor_a \xrightarrow{\overline{a}b} L$, then $L =_\pi \mathbf{0}$. Notice that $a(x) \;\mathfrak{G}\; (b)A$ by extensionality and $(b)A$ cannot do any internal action. Consequently $\lceil \underline{i} \rfloor_a \,|\, (b)A \xrightarrow{\iota} L \,|\, A'$ for some A'. It follows that $L \,|\, A' =_\pi \mathbf{0}$, hence $L =_\pi \mathbf{0}$.
7. We conclude from the previous observation that b and b' must be distinct names whenever $i \neq j$ and $\lceil \underline{i} \rfloor_a \xrightarrow{\overline{a}b}$ and $\lceil \underline{j} \rfloor_a \xrightarrow{\overline{a}b'}$.

There are three cases according to the type of the external actions of $\lceil \underline{i} \rfloor_a$.

1. There are two distinct numerals $\underline{j}, \underline{k}$ such that both $\lceil \underline{j} \rfloor_a$ and $\lceil \underline{k} \rfloor_a$ can perform bound output actions. Suppose

$$\begin{aligned} \lceil \underline{j} \rfloor_a &\xrightarrow{\overline{a}(b)} N, \\ \lceil \underline{k} \rfloor_a &\xrightarrow{\overline{a}(b)} O. \end{aligned}$$

   Consider the $\mathbb{VPC}$-processes defined as follows:

$$\begin{aligned} C_d &\stackrel{\text{def}}{=} !a(x).\overline{d}(x), \\ C_e &\stackrel{\text{def}}{=} !a(x).\overline{e}(x), \\ C_j &\stackrel{\text{def}}{=} a(x).\mathit{if}\, x = \underline{j}\ \mathit{then}\, \overline{d}(x)\ \mathit{else}\, \overline{e}(x). \end{aligned}$$

   It is easily seen that $C_d \,|\, C_e = C_d \,|\, C_e \,|\, C_j$. Let D_d, D_e, D_j be the π-processes such that $C_d \mathfrak{G} D_d \not\twoheadrightarrow$, $C_e \mathfrak{G} D_e \not\twoheadrightarrow$ and $C_j \mathfrak{G} D_j \not\twoheadrightarrow$. By extensionality and soundness one has that $D_d \,|\, D_e = D_d \,|\, D_e \,|\, D_j$ and that D_d, D_e, D_j can only do input actions at a. Further properties about D_d, D_e, D_j are stated as follows:

   - $D_e \,|\, \lceil \underline{j} \rfloor_a \xrightarrow{\iota} D'_e$ for some D'_e such that D'_e may only perform external actions at a and e. Notice that D'_e may not perform any output action at a due to the fact that D'_e cannot do a nondeterministic computation step.
   - $D_j \,|\, \lceil \underline{j} \rfloor_a \xrightarrow{\iota} D'_j$ for some D'_j such that D'_j may only perform actions at d.

   Now suppose $D_d \xrightarrow{ab} D'_d$, $D_e \xrightarrow{ab} D'_e$ and $D_j \xrightarrow{ab} D'_j$. Without loss of generality we may assume that $D_d \,|\, D_e \,|\, D_j \xrightarrow{ab} D_d \,|\, D_e \,|\, D'_j$ is simulated by $D_d \,|\, D_e \xrightarrow{ab} D_d \,|\, D'_e$. We would then have the following equality

$$(a)(e)(b)(D_d \,|\, D'_e \,|\, N) = (a)(e)(b)(D_d \,|\, D_e \,|\, D'_j \,|\, N). \qquad (13)$$

   This is a contradiction since the right hand side of (13) is observable whereas the left hand side is unobservable.

2. Suppose that almost all members of $\{\lceil \underline{0} \rfloor_a, \lceil \underline{1} \rfloor_a, \lceil \underline{2} \rfloor_a, \ldots\}$ can perform free output actions. Let Q be an interpretation of the following $\mathbb{VPC}$-process

$$a(z).\mathit{if}\ z \text{ is even } \mathit{then}\ \overline{a}(z+1)\ \mathit{else}\ \text{diverge}.$$

For each natural number i, we have the following interaction

$$\lceil \underline{2i} \rfloor_a \,|\, Q \xrightarrow{\iota}= \lceil \underline{2i+1} \rfloor_a. \qquad (14)$$

What is described in (14) renders a contradiction since Q may contain only a finite number of global names and consequently the set of the global names released at a by the processes in $\{\lceil \underline{2} \rfloor_a, \lceil \underline{4} \rfloor_a, \ldots\}$ cannot be disjoint from those released by the processes in $\{\lceil \underline{1} \rfloor_a, \lceil \underline{3} \rfloor_a, \ldots\}$.

3. Now suppose the immediate actions of $\lceil \underline{i} \rfloor_a$ are input actions. Then the processes D_d, D_e, D_j defined in the previous case would have to do output actions. A contradiction can be similarly derived as in the first case. So this case is also impossible.

We conclude that $\mathbb{VPC} \not\sqsubseteq \pi$. $\square$

The intuition behind the above proof is that a numeral has to be coded up in π-calculus by a nontrivial process that first releases a local name and then tells the caller the numeral it represents, but then the separation in time between releasing a local name and doing the real transition makes it vulnerable to attacks from environments. Suppose a π process P invokes D_j by sending a local name to D_j. This is an interaction step between P and $D_d \,|\, D_e \,|\, D_j$, which must be simulated by at least one step interaction between P and $D_d \,|\, D_e$, otherwise some process in the environment might interact with P and possesses the resource for good. Now $D_d \,|\, D_e$ can carry out the interaction with the help of either D_d or D_e. Either way it may fail since it does not know at this stage which numeral it is getting from P. The problem caused by simulating an atomic action by a sequence of atomic actions is one of the reasons to introduce bisimulation equivalence in the first place. See the introductory discussions given in (Milner, 1989a).

4.4. Self Interpretation

An obvious consequence of Definition 18 is that there could be many subbisimilarities from one model to another. Why don't we focus on the "largest subbisimilarity" between two models? The results in this section will tell us that we really should not do that. Before proving these results we need to introduce some terminologies.

Definition 21. Two subbisimilarities $\mathfrak{R}, \mathfrak{R}' : \mathbb{M} \to \mathbb{N}$ are *incompatible* if there is some $P \in \mathcal{P}_{\mathbb{M}}$ such that $Q \neq O$ whenever $P\mathfrak{R}Q$ and $P\mathfrak{R}'O$. They are compatible if they are not incompatible. A subbisimilarity $\mathfrak{R} : \mathbb{M} \to \mathbb{N}$ is *maximal* if $\mathfrak{R}' \subseteq \mathfrak{R}$ whenever $\mathfrak{R}' : \mathbb{M} \to \mathbb{N}$ is compatible with $\mathfrak{R}$.

Every subbisimilarity $\mathfrak{R}$ is contained in the maximal subbisimilarity $\mathfrak{R};=$. Two compatible subbisimilarities are essentially the same interpretation. So we may as well identify a subbisimilarity $\mathfrak{R}$ with the maximal subbisimilarity $\mathfrak{R};=$. We say that there is a unique interpretation from $\mathbb{M}$ to $\mathbb{N}$ if all the subbisimilarities from $\mathbb{M}$ to $\mathbb{N}$ are compatible; in other words, the largest subbisimilarity from $\mathbb{M}$ to $\mathbb{N}$ exists. The largest subbisimilarity from $\mathbb{M}$ to itself is essentially the absolute equality on the model $\mathbb{M}$. In the rest of this section we inspect the self interpretations on the three prime models.

Proposition 4.10. *There are an infinite number of pairwise incompatible subbisimilarities from* $\mathbb{VPC}$ *to* $\mathbb{VPC}$.

PROOF. Suppose f is a bijective computable function. Clearly its inverse function f is also a bijective computable function. We may think of f as an encoding function and f^{-1} the corresponding decoding function. A relation $[\![\_]\!]^{\mathsf{f}}$ from $\mathbb{VPC}$ to $\mathbb{VPC}$ can be defined that makes use of the encoding and decoding functions. The encoding of the choice terms is defined as follows:

$$[\![\sum_{1\le i\le k} a(x).T_i]\!]^{\mathsf{f}} \stackrel{\text{def}}{=} \sum_{1\le i\le k} a(u).(c)((b)(\overline{b}(u)\,|\,F_b^c(\mathsf{f}^{-1}))\,|\,c(x).[\![T_i]\!]^{\mathsf{f}}),$$

$$[\![\sum_{1\le i\le k} \overline{a}(t_i).T_i]\!]^{\mathsf{f}} \stackrel{\text{def}}{=} (\widetilde{c})(\prod_{1\le i\le k}(b)(\overline{b}(t_i)\,|\,F_b^{c_i}(\mathsf{f}))\,|\,c_1(z_1)\ldots c_k(z_k).\sum_{1\le i\le k}\overline{a}(z_i).[\![T_i]\!]^{\mathsf{f}}),$$

where c is a fresh name. In the above encoding, $F_b^c(\mathsf{f}^{-1})$ is the $\mathbb{VPC}$-process that after inputting a numeral at b, say $\underline{n}$, calculates $\mathsf{f}^{-1}(\underline{n})$ before delivering the result at c. The process $F_b^{c_i}(\mathsf{f})$ is similar. The precise definition of $F_b^c(\_)$ is given in (Fu, 2013b). The replication terms can be interpreted in the same fashion. It is not difficult to see that $[\![\_]\!]^{\mathsf{f}};=_{\mathbb{VPC}}$ is a subbisimilarity from $\mathbb{VPC}$ to $\mathbb{VPC}$. □

$\mathbb{VPC}$ is an interactive version of the recursion theory. The above proposition implies that if the recursion theory can be embedded into a model of interaction, it can be embedded in an infinite number of ways.

Using the same idea it is routine to establish the next proposition.

Proposition 4.11. *There are an infinite number of pairwise incompatible subbisimilarities from* $\mathbb{IM}$ *to* $\mathbb{IM}$.

We have seen that generally models of interaction admit multiple self interpretations. It would be nice to see a model on which the only self interpretation is the trivial one.

Theorem 4.12. *The absolute equality $=_\pi$ is the largest subbisimilarity from the π-calculus to itself.*

PROOF. Suppose $\sqsubseteq_\pi$ is a maximal subbisimilarity from π to π. We shall show that $\sqsubseteq_\pi$ is a π-bisimulation. To start with we need to derive a number of properties about the interpretation $\sqsubseteq_\pi$.

1. First of all we prove that the process $\overline{a}c$ is interpreted by itself. The following arguments resemble those in the proof of Theorem 4.9.
   (a) Let A be such that $a(x) \sqsubseteq_\pi A \not\rightarrow$. The process A may not do both an input action and an output action at the name a. Suppose otherwise. Then $A \,|\, A$ would be able to do an internal action, say $A \,|\, A \xrightarrow{\tau} A_1$. By extensionality this internal action may be caused either by $A \xrightarrow{\overline{a}(c)}$ and $A \xrightarrow{ac}$, or by $A \xrightarrow{\overline{a}a}$ and $A \xrightarrow{aa}$. Assume $A \,|\, A = A_1$. In the first case $A \,|\, A \xrightarrow{\overline{a}(c_1)}\xrightarrow{ac_1} A_1'$ must be matched up by $A_1 \rightarrow^* \xrightarrow{\overline{a}(c_1)}\xrightarrow{ac_1} A_2 = A_1'$ for some A_2, where the two external actions at a may induce an interaction. Now $A_1' \rightarrow^* \xrightarrow{\overline{a}(c_1)}\xrightarrow{ac_1} A_2'$ must also be simulated by A_2. Continuing in this way we get an infinite action sequence
$$A \,|\, A \xrightarrow{\overline{a}(c_1)}\xrightarrow{ac_1}\rightarrow^* \xrightarrow{\overline{a}(c_2)}\xrightarrow{ac_2}\rightarrow^* \xrightarrow{\overline{a}(c_3)}\xrightarrow{ac_3} \ldots . \qquad (15)$$
   It follows from (15) that $A \,|\, A$ would be able to do an infinite sequence of internal actions, which would violate the codivergence property. The same contradiction can be derived if $A \,|\, A \xrightarrow{\tau} A_1$ was caused by $A \xrightarrow{aa}$ and $A \xrightarrow{\overline{a}a}$. So the assumption $A \,|\, A = A_1$ was wrong. However the change-of-state internal action $A \,|\, A \xrightarrow{\iota} A_1$ also renders a contradiction since $a(x) \,|\, a(x)$ cannot do any change of state internal action. So A may perform either input actions at a or output actions at a; it cannot do both input and output actions at a.
   (b) For each c, let $\overline{A}_c$ be such that $\overline{a}c \sqsubseteq_\pi \overline{A}_c \not\rightarrow$. By codivergence the set of the internal action sequences of $A \,|\, \overline{A}_c$ form a finite

tree. It follows that the longest external action sequence cannot be bisimulated by any A' satisfying $A \,|\, \overline{A}_c \xrightarrow{\tau} A'$. So $A \,|\, \overline{A}_c \xrightarrow{\iota} A' = \mathbf{0}$ whenever $A \,|\, \overline{A}_c \xrightarrow{\tau} A'$.

(c) If $\overline{A}_c$ can do an input (output) action then $\overline{A}_f$ can do an input (output) action for all f, where $\overline{A}_f$ is such that $\overline{a}f \sqsubseteq_\pi \overline{A}_f \not\twoheadrightarrow$.

(d) Let $C_d, C_e, C_\vee$ be defined as follows:

$$\begin{aligned} C_d &\stackrel{\text{def}}{=} !a(x).\overline{d}x, \\ C_e &\stackrel{\text{def}}{=} !a(x).\overline{e}x, \\ C_\vee &\stackrel{\text{def}}{=} a(x).([x{=}c]\overline{d}x \,|\, [x{\neq}c]\overline{e}x). \end{aligned}$$

Let $D_d, D_e, D_\vee$ be the π-processes such that $C_d \sqsubseteq_\pi D_d \not\twoheadrightarrow$, $C_e \sqsubseteq_\pi D_e \not\twoheadrightarrow$ and $C_\vee \sqsubseteq_\pi D_\vee \not\twoheadrightarrow$. By extensionality and soundness the equality

$$D_d \,|\, D_e = D_d \,|\, D_e \,|\, D_\vee \tag{16}$$

follows from the equality $C_d \,|\, C_e = C_d \,|\, C_e \,|\, C_\vee$.

(e) Suppose $D_\vee$ could do an output action, say $D_\vee \xrightarrow{\lambda} D'_\vee$. Then $\overline{A}_f$ may only do input actions at a, and consequently D_d, D_e may only do output actions at a. Using the fact that the τ-tree of $D_d \,|\, \overline{A}_f$ is finite, it is easy to see that every internal action $D_d \,|\, \overline{A}_f \xrightarrow{\tau} L$ is a change-of-state action that necessarily bisimulates $!a(x).\overline{d}x \,|\, \overline{a}f \xrightarrow{\tau} !a(x).\overline{d}x \,|\, \overline{d}f$. Moreover L may not perform any input action at the name a. These remarks also apply to D_e.

(f) It follows from (16) that the action must be bisimulated by either D_d or D_e. Without loss of generality assume that $D_d \xrightarrow{\lambda} D'_d$ such that $D'_d \,|\, D_e = D_d \,|\, D_e \,|\, D'_\vee$. Suppose $\overline{A}_f \xrightarrow{\lambda'} \overline{A}'_f$ for some input action λ' that is complementary to λ. Let $\overline{A}_f \,|\, D_d \xrightarrow{\tau} A'$ be the interaction induced by $\overline{A}_f \xrightarrow{\lambda'} \overline{A}'_f$ and $D_d \xrightarrow{\lambda} D'_d$, and $\overline{A}_f \,|\, D_\vee \xrightarrow{\tau} A''$ be induced by $\overline{A}_f \xrightarrow{\lambda'} \overline{A}'_f$ and $D_\vee \xrightarrow{\lambda} D'_\vee$. Clearly we must have

$$A' \,|\, D_e = D_d \,|\, D_e \,|\, A''. \tag{17}$$

According to property (e), the processes A' and D_e may not interact. Similarly A'' and $D_d \,|\, D_e$ may not interact. It follows that the equality (17) renders a contradiction since $(a)(e)(A' \,|\, D_e)$ is observable whereas $(a)(e)(D_d \,|\, D_e \,|\, A'')$ is not observable. We conclude that $D_d, D_e, D_\vee$ can only perform input action(s) at a. Accordingly for each c, the process $\overline{A}_c$ can only do output action(s) at a.

(g) Suppose that there existed some $\overline{A}_g$ that could do a bound output action, say $\overline{A}_g \xrightarrow{\overline{a}(d')}$. Now if $\overline{A}_h$ could also do a bound output action, then by α-convention we may assume that $\overline{A}_g \xrightarrow{\overline{a}(d'')}$ and $\overline{A}_h \xrightarrow{\overline{a}(d'')}$ for some fresh d''. If $\overline{A}_h$ performs a free output action, the output action must be $\overline{a}h$ according to (h). By extensionality we may assume that h does not appear as a global name in $\overline{A}_g$. Thus $\overline{A}_g \xrightarrow{\overline{a}(h)}$ by α-conversion. In either case we may repeat the argument in (f) to derive a contradiction.

(h) We conclude that, for all c, the process $\overline{A}_c$ can only do free output action(s). Since $A \,|\, \overline{A}_c \xrightarrow{\tau}= \mathbf{0}$, it must be the case that $\overline{A}_c \xrightarrow{\overline{a}c'}= \mathbf{0}$ for some c'. Now let O_a be a process such that $a(x).\overline{x} \sqsubseteq_\pi O_a \not\rightarrow$. By extensionality we may assume that a is the only global name that appears in O_a. The action $a(x).\overline{x} \,|\, \overline{a}c \xrightarrow{\iota} \overline{c}$ must be matched up by $O_a \,|\, \overline{A}_c \xrightarrow{\iota} \overline{C}_c$ such that $\overline{C}_c$ is observable at c. So c' must be c.

What we have proved is that $\overline{a}c \sqsubseteq_\pi \overline{a}c$ for all a, c.

2. The above argument can be repeated to show that a process of the form $\overline{a}c + \overline{a}d$ is interpreted by itself.
3. Let N be such that $a(x).[x{\neq}c]e + a(x).d \;\; \sqsubseteq_\pi \;\; N \not\rightarrow$. The following properties hold by the result stated in (2) and the extensionality:

   (a) $N \xrightarrow{ag}\rightarrow^* \xrightarrow{dh} \mathbf{0}$ for all g and all h;

   (b) $N \xrightarrow{ag}\rightarrow^* \xrightarrow{eh} \mathbf{0}$ for every name $g \neq c$ and every h.

   We conclude by Theorem 3.8 that $N = a(x).[x{\neq}c]e + a(x).d$. In other words $a(x).[x{\neq}c]e + a(x).d \sqsubseteq_\pi a(x).[x{\neq}c]e + a(x).d$.
4. Using the fact that processes of the form $\overline{a}c{+}\overline{a}d$ and $a(x).[x{\neq}c]e{+}a(x).d$ are interpreted by themselves, it is standard to prove the following properties for all P, Q such that $P \sqsubseteq_\pi Q$:
   - If $P \xrightarrow{ac} P'$ then $Q \Longrightarrow Q'' \xrightarrow{ac} Q'$ and $P \sqsubseteq_\pi Q''$ and $P' \sqsubseteq_\pi Q'$.
   - If $Q \xrightarrow{ac} Q'$ then $P \Longrightarrow P'' \xrightarrow{ac} P'$ and $P'' \sqsubseteq_\pi Q$ and $P' \sqsubseteq_\pi Q'$.
   - If $P \xrightarrow{\overline{a}c} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\overline{a}c} Q'$ and $P \sqsubseteq_\pi Q''$ and $P' \sqsubseteq_\pi Q'$.
   - If $Q \xrightarrow{\overline{a}c} Q'$ then $P \Longrightarrow P'' \xrightarrow{\overline{a}c} P'$ and $P'' \sqsubseteq_\pi Q$ and $P' \sqsubseteq_\pi Q'$.

5. Suppose L contains neither the replication operator nor the localization operator. Let M be such that $L \sqsubseteq_\pi M$. Assume that $M \overset{\lambda_1}{\twoheadrightarrow} \ldots \overset{\lambda_k}{\twoheadrightarrow} M' \overset{\overline{a}(c)}{\twoheadrightarrow} M''$ for some M', M'', where $\lambda_1, \ldots, \lambda_k$ are either input actions or free output actions. Using observers discussed in (2) and (3), one may force L to perform the action sequence $L \overset{\lambda_1}{\twoheadrightarrow} \ldots \overset{\lambda_k}{\twoheadrightarrow} L'$ such that $L' \sqsubseteq_\pi M'$. As in the proof of (3) one can argue that, for fresh d, e, the process $a(x).[x \notin gn(L' \mid M')]e + a(x).d$ is interpreted by itself. But then

   $$L' \mid (a(x).[x \notin gn(L' \mid M')]e + a(x).d) \sqsubseteq_\pi M' \mid (a(x).[x \notin gn(L' \mid M')]e + a(x).d)$$

   renders a contradiction. We conclude that M may never perform any bound output action.
6. It follows from (4) and (5) that the following relation

   $$\{(L, M) \mid L \sqsubseteq_\pi M, \text{ and } L \text{ contains neither replication nor localiztion}\}$$

   is an external bisimulation. So every π-process that contains neither the replication operator nor the localization operator is interpreted by itself under $\sqsubseteq_\pi$.

The importance of property (6) is that it allows one to repeat the proof of Theorem 3.8 to show that $\sqsubseteq_\pi$ is an external bisimulation. Therefore $\sqsubseteq_\pi \subseteq =_\pi$. The coincidence between $\sqsubseteq_\pi$ and $=_\pi$ follows by the maximality of $\sqsubseteq_\pi$. So the absolute equality on π is the only interpretation from π to π. □

4.5. Subbisimilarity for Pi Variant

The choice operator we have used in this paper is weaker than the choice operators one encounters in literature. Apart from the general unguarded choice operator (Milner, 1989a), the mixed choice (or guarded choice) and the separated choice are sufficient in many applications (Palamidessi, 2003; Fu and Lu, 2010). Further variants of the π-calculus can be obtained by replacing the replication operator by the fixpoint operator or parametric definition. Altogether we have nine variants of the π-calculus. See (Fu and Lu, 2010) for detailed description.

We are interested in the relative expressive power of the nine variants. Since the action sets of the variants are all the same, we could try an external characterization of the subbisimilarity.

Definition 22. Suppose π_0, π_1 are two of the nine π-variants. A total relation $\mathcal{R}$ from π_0 to π_1 is a π-*subbisimulation* if it is a codivergent bisimulation and the following statements are valid whenever $P\mathcal{R}Q$:

1. If $P \xrightarrow{\ell} P'$ then $Q \Longrightarrow Q'' \xrightarrow{\ell} Q'$ and $P\mathcal{R}Q''$ and $P'\mathcal{R}Q'$.
2. If $Q \xrightarrow{\ell} Q'$ then $P \Longrightarrow P'' \xrightarrow{\ell} P'$ and $P''\mathcal{R}Q$ and $P'\mathcal{R}Q'$.

We write $\pi_0 \sqsubseteq_\pi \pi_1$ if there is a π-subbisimulation from π_0 to π_1.

If $\pi_0 \sqsubseteq_\pi \pi_1$ then the largest π-subbisimulation from π_0 to π_1 exists, which is necessarily extensional and equipollent, hence the following lemma.

Lemma 4.13. *If $\pi_0 \sqsubseteq_\pi \pi_1$ then $\pi_0 \sqsubseteq \pi_1$.*

The converse of the lemma also holds.

Theorem 4.14. *Suppose π_0, π_1 are two of the nine π-variants. Then $\pi_0 \sqsubseteq_\pi \pi_1$ if and only if $\pi_0 \sqsubseteq \pi_1$.*

Proof. We only have to prove that $\pi_0 \sqsubseteq \pi_1$ implies $\pi_0 \sqsubseteq_\pi \pi_1$, which amounts to showing that $\sqsubseteq$ is a π-subbisimulation from π_0 to π_1. But this is just the proof of Theorem 4.12. $\square$

Corollary 4.15. *Suppose π_0, π_1 are two of the nine π-variants. If $\pi_0 \sqsubseteq \pi_1$ then there is a unique interpretation of π_0 into π_1.*

Fu and Lu (2010) have studied the relative expressive power of the nine π-variants in terms of a relationship, called codivergent subbisimilarity in (Fu and Lu, 2010), that is weaker than the π-subbisimulation of Definition 22. The difference is that in (Fu and Lu, 2010) only the weak bisimulation property is required whereas in Definition 22 the branching style bisimulation is expected. However the expressiveness results about the π-variants obtained in (Fu and Lu, 2010) should remain the same had we apply the subbisimilarity of this paper as the criterion. This is because a negative result by a certain criterion remains valid if a stronger criterion is adopted and all the encodings supporting the positive results in (Fu and Lu, 2010) satisfy the stronger requirement of the branching style bisimulation.

We have therefore placed the results obtained in (Fu and Lu, 2010) on a firmer foundation.

4.6. Expressiveness of Polyadic Pi

It has been a folklore that the polyadic π-calculus cannot be coded up in the monadic π-calculus. In this section we formally prove this popular belief using the subbisimilarity criterion. Let π^k, for $k \geq 1$ be the k-ary polyadic π-calculus whose input terms are of the form $n(x_1, \ldots, x_k).T$ and whose output terms are of the shape $\overline{n}\langle n_1, \ldots, n_k\rangle.T$. The 1-ary polyadic π-calculus is just the monadic π-calculus. The external π^k-bisimilarity $\approx_{\pi^k}$ is defined in the standard manner. The coincidence between $\approx_{\pi^k}$ and $=_{\pi^k}$ can be established by recycling the proof of Theorem 3.8.

Proposition 4.16. *For each $k \geq 2$ the relations $\approx_{\pi^k}, =_{\pi^k}$ coincide.*

Proposition 4.16 implies that the identity map from π^k to π^{k+1} is a subbisimilarity. The latter is also strictly more expressive than the former.

Theorem 4.17. $\pi \sqsubset \pi^2 \sqsubset \pi^3 \sqsubset \pi^4 \sqsubset \ldots$.

Proof. We only prove $\pi^2 \not\sqsubseteq \pi$. Consider the π^2-processes defined as follows:

$$\begin{aligned} C^2_d &\stackrel{\text{def}}{=} !a(x,y).\overline{d}, \\ C^2_e &\stackrel{\text{def}}{=} !a(x,y), \\ C^2_\vee &\stackrel{\text{def}}{=} a(x,y).(\overline{x}.\overline{d} \,|\, \overline{y}), \end{aligned}$$

where $\overline{d}$ for example stands for $(c)\overline{d}\langle c, c\rangle$. Let $D^2_d, D^2_e, D^2_\vee$ be the interpretations in π of $C^2_d, C^2_e, C^2_\vee$ respectively. Obviously $C^2_d \,|\, C^2_e \,|\, C^2_\vee = C^2_d \,|\, C^2_e$ and $D^2_d \,|\, D^2_e \,|\, D^2_\vee = D^2_d \,|\, D^2_e$. If $D^2_d, D^2_e, D^2_\vee$ can do immediate output actions then a contradiction can be easily derived. Now suppose $D^2_d, D^2_e, D^2_\vee$ can do immediate input actions. Let $\overline{A}_{f,g}, \overline{A}_f, \overline{A}_g, \overline{A}_a$ be the interpretations of $\overline{a}\langle f, g\rangle, \overline{a}\langle f, f\rangle, \overline{a}\langle g, g\rangle, \overline{a}\langle a, a\rangle$ respectively. If any of $\overline{A}_{f,g}, \overline{A}_f, \overline{A}_g, \overline{A}_a$ could do a bound output action at a, a contradiction would be derived. Otherwise $\overline{A}_{f,g}$ would be able to perform an output action that is also admitted by one of $\overline{A}_f, \overline{A}_g, \overline{A}_a$, which leads to a contradiction again. We are done. $\square$

It is worth remarking that the above proof does not make use of the match/mismatch operator. The polyadic π-calculi referred to in Theorem 4.17 can be understood as the minimal ones.

5. Theory of Completeness

The four principles introduced in Section 1 do not rule out uninteresting models. How interesting should our models be? One interpretation, and probably the only interpretation, is that a model of interaction should allow one to solve interesting problems. So the question is split into two:

1. what are the interesting problems? and
2. how do we place a minimal requirement on a model so that it does provide solutions to the interesting problems?

A preliminary answer to the second question is in terms of expressive completeness: There exists a *least* model that admits the solutions to all the interesting problems, and every model of interaction should subsume the least model. The existence of the universal relationship $\sqsubseteq$ makes it possible to formalize this idea. How powerful should the least model be? Since Theory of Interaction aims at a uniform treatment of both computation models and interaction models, the only reasonable assumption is that the interesting problems are those that are solvable by the Turing machines. Thus the answer to the first question ought to be that all computable functions are definable in the least model.

There are a number of well known computation models, all of them can be extended to interaction models. These interaction models may vary in the power of atomic actions, programming styles and so on. The differences are suppressed in the computation models due to the absence of external interactions, and are largely ignored since the focus has been on the definable functions. But the differences do imply that the interactive versions of these computation models are generally incompatible as we have seen in Section 4. What we seek is a model that provides the bare primitives for *computability* and *interactability*. The $\mathbb{C}$ model is an ideal candidate for several reasons:

1. A functional $\mathbb{C}$-process specifies what a function is, it says absolutely nothing about how to implement it. The requirement is minimal from the computational viewpoint.
2. The input-output mechanism of $\mathbb{C}$ has no more power than sending and receiving values, which is minimal from the viewpoint of interaction.
3. There is not any nondeterminism introduced by any atomic $\mathbb{C}$-process. All nondeterminism in $\mathbb{C}$ is caused by the composition operator.
4. The equality theory of $\mathbb{C}$ is extremely simple. The proof of Theorem 3.6 provides enough evidence that there is little redundancy in $\mathbb{C}$.

Let $\mathfrak{Mod}$ be the class of all models of interaction. The first fundamental postulate of Theory of Interaction states that all the models in $\mathfrak{Mod}$ are conservative extensions over $\mathbb{C}$.

Axiom of Completeness. $\forall\, \mathbb{M} \in \mathfrak{Mod}.\ \mathbb{C} \sqsubseteq \mathbb{M}$.

Since Theory of Interaction is a formal theory for both the computation models and the interaction models, the Axiom of Completeness is best seen as a formalization of the Church-Turing Thesis.

Axiom of Completeness has significant implication to the process theory. A calculus $\mathbf{L}$ that fails $\mathbb{C} \sqsubseteq \mathbf{L}$ is either too weak to define all the recursive functions, or is lack of an interaction mechanism that admits proper communications. As mentioned before, the purpose of introducing an interaction model is precisely to internalize the meta operations like the instantiations of function parameters, the announcements of computing results. If the interactions are useful to the participants at all, it should be possible that the correct inputs are picked up in an orderly manner without corruption, and the results are released to the targeted receivers eventually. The failure of $\mathbf{L}$ to satisfy $\mathbb{C} \sqsubseteq \mathbf{L}$ is an obvious indication that $\mathbf{L}$ cannot be considered as both a computation model and an interaction model. We shall use the phrase "$\mathbb{M}$ is complete" to refer to the fact that $\mathbb{C} \sqsubseteq \mathbb{M}$.

Axiom of Completeness points out the inadequacy of some of the Turing completeness claims stated in the process calculus literature. The criticism is that the proofs of these claims boil down to showing that the target models appear Turing complete to someone *outside* the models. This level of completeness can be exploited to establish undecidability results. It is however almost useless to programming since in a closed model a process never communicates to anyone outside the model. Our view has been that the processes of a complete model should appear Turing complete to the processes *inside* the model. In reality it is this internal completeness that truly matters.

Axiom of Completeness has far-reaching implications to practice. Let's see an example. We will show below that the π-calculus is complete. By the Axiom of Completeness there must be a π-process that after receiving a numeral $\underline{i}$ at channel a outputs the $\underline{i}$-th Fibonacci numeral at channel b, and then it is ready to input another numeral at a. The Axiom of Completeness allows us to talk about a process with certain functionality without worrying about its definition. Getting confident in using the Axiom of Completeness this way is the first step to more sophisticated Theory of Interaction.

5.1. Complete Model

We point out in this section that the three models defined in Section 2 are indeed complete. A proof of completeness can be structured in four steps:

1. Define a map $\mathcal{E}$ from the set $\mathcal{P}_{\mathbb{C}}$ of the functional $\mathbb{C}$-processes, the value $\mathbb{C}$-processes and Ω to $\mathcal{P}_{\mathbb{M}}$.
2. Extend $\mathcal{E}$ to a map from $\mathcal{P}_{\mathbb{C}}$ to $\mathcal{P}_{\mathbb{M}}$ by letting $\mathcal{E}(P_0 \,|\, P_1) \stackrel{\text{def}}{=} \mathcal{E}(P_0) \,|\, \mathcal{E}(P_1)$.
3. Prove that the composition $\mathcal{E}; =_{\mathbb{M}}$ is equipollent, codivergent and bisimilar. Notice that $\mathcal{E}; =_{\mathbb{M}}$ is extensional by definition.
4. Argue that $\mathcal{E}; =_{\mathbb{M}}$ is sound.

Suppose $P_0 =_{\mathbb{C}} P_1$, $\mathcal{E}(P_0) =_{\mathbb{M}} Q_0$ and $\mathcal{E}(P_1) =_{\mathbb{M}} Q_1$. If $\mathcal{E}; =_{\mathbb{M}}$ is equipollent and codivergent, then Ω must be correctly encoded. It follows immediately that $\mathcal{E}(P_0) =_{\mathbb{M}} \mathcal{E}(P_1)$, thanks to Theorem 3.6. Thus $Q_0 =_{\mathbb{M}} Q_1$. So the soundness comes for free once (3) has been done. We conclude that practically only step (1) and step (3) are compulsory.

By Church-Turing Thesis every computable function can be implemented by a Turing Machine. It is obvious that a process of the form $F_a^b(\mathsf{f}(x))$ can be coded up by an AITM. A value process of the form $\overline{a}(\underline{n})$ can also be easily translated to an ITM. By composing with the absolute equality one gets a subbisimilarity from $\mathbb{C}$ to $\mathbb{IM}$.

Theorem 5.1. $\mathbb{C} \sqsubseteq \mathbb{IM}$.

In (Fu, 2013b) an encoding of the recursive functions into $\mathbb{VPC}$ is given. By composing with $=_{\mathbb{VPC}}$ the encoding gives rise to a subbisimilarity from $\mathbb{C}$ to $\mathbb{VPC}$, which immediately implies the completeness of the latter. The encoding does not refer to any choice operator.

Theorem 5.2. $\mathbb{C} \sqsubseteq \mathbb{VPC}$ *and* $\mathbb{C} \sqsubseteq \mathbb{VPC}^-$.

$\mathbb{VPC}^-$ is the minimal model among the class of the value-passing calculi discussed in (Fu, 2013b). Using the terminology of this paper, $\mathbb{VPC}^-$ is subbisimilar to every value-passing calculus. It follows that all the value-passing calculi are complete. Using the result from (Fu, 2013b), we may summarize by saying that a value-passing calculus, in the general sense of the terminology defined in (Fu, 2013b), is complete if and only if it has a numeric system.

Next we prove that π^M is complete. We will show that the recursive functions are definable in π^M, which is more than necessary for our purpose.

The encoding of the recursive functions into the π-calculus makes use of the processes defined in Appendix B. The structural definition on the nonfunctional processes is as follows:

$$\begin{aligned}
[\![\mathbf{0}]\!]^\pi &\overset{\text{def}}{=} \mathbf{0}, \\
[\![\Omega]\!]^\pi &\overset{\text{def}}{=} (c)(\overline{c}c \mid !c(x).\overline{x}x), \\
[\![\overline{b}(\underline{n})]\!]^\pi &\overset{\text{def}}{=} [\![\underline{n}]\!]^\pi_b, \\
[\![P \mid Q]\!]^\pi &\overset{\text{def}}{=} [\![P]\!]^\pi \mid [\![Q]\!]^\pi,
\end{aligned}$$

The translations of the successor function, constant functions, projection functions and composition functions are given below. In the following structural definition we write $F^b_{a_1,..,a_k}(\mathsf{f}(x_1,..,x_k))$ to force the interpretation of the k-ary recursive function $\mathsf{f}(x_1,..,x_k)$ to be of the right type. The process $[\![F^b_{a_1,..,a_k}(\mathsf{f}(x_1,..,x_k))]\!]$ must pick up the inputs consecutively at $a_1,\ldots,a_k$ and deliver the result at b.

$$\begin{aligned}
[\![F^b_{a_1}(\mathsf{s}(x))]\!]^\pi &\overset{\text{def}}{=} (d_1)Rep(a_1,d_1).(c)Copy(d_1,c).\overline{b}(e,f).\overline{e}c, \\
[\![F^b_{a_1,..,a_k}(\underline{n}(x_1,..,x_k))]\!]^\pi &\overset{\text{def}}{=} (d_1)Rep(a_1,d_1).\cdots.(d_k)Rep(a_k,d_k).[\![\underline{n}]\!]^\pi_b, \\
[\![F^b_{a_1,..,a_k}(\mathsf{p}^i_k(x_1,..,x_k))]\!]^\pi &\overset{\text{def}}{=} (d_1)Rep(a_1,d_1).\cdots.(d_k)Rep(a_k,d_k).Copy(d_i,b), \\
[\![F^b_{a_1,..,a_k}(\mathsf{f}(\mathsf{f}_1(\widetilde{x}),..,\mathsf{f}_i(\widetilde{x})))]\!]^\pi &\overset{\text{def}}{=} (d_1)Rep(a_1,d_1).\cdots.(d_k)Rep(a_k,d_k).(c_1\ldots c_i) \\
&\qquad ([\![F^{c_1}_{d_1,..,d_k}(\mathsf{f}_1(\widetilde{x}))]\!]^\pi \mid \ldots \mid [\![F^{c_i}_{d_1,..,d_k}(\mathsf{f}_i(\widetilde{x}))]\!]^\pi \\
&\qquad \mid [\![F^b_{c_1,..,c_i}(\mathsf{f}(\widetilde{x}))]\!]^\pi).
\end{aligned}$$

The encodings of the recursion functions and the minimization functions are slightly more involved.

1. $[\![F^b_{a_1,..,a_{k+1}}(\mathsf{rec}\ z.[\mathsf{f}(\widetilde{x},x',z),\mathsf{g}(\widetilde{x})])]\!]^\pi$ is the following process

$$(d_1)Rep(a_1,d_1).\cdots.(d_{k+1})Rep(a_{k+1},d_{k+1}).(f)(\overline{f}(b,d_{k+1}) \mid Rec),$$

   where Rec stands for the π-process

$$\begin{aligned}
&!f(v,x).x(y,z).(z.[\![F^v_{d_1,..,d_k}(\mathsf{g}(x_1,..,x_k))]\!]^\pi \\
&\quad \mid y(x).(d)\overline{f}\langle d,x\rangle.[\![F^v_{d_1,..,d_k dx}(\mathsf{f}(x_1,..,x_{k+1},z))]\!]^\pi).
\end{aligned}$$

2. $[\![F^b_{a_1,..,a_k}(\mu z.\mathsf{f}(\widetilde{x},z))]\!]^\pi$ is the following process

$$(d_1)Rep(a_1,d_1).\cdots.(d_k)Rep(a_k,d_k).(e)([\![F^e_{d_1,..,d_k,d}(\mathsf{f}(\widetilde{x},z))]\!]^\pi \mid [\![!\underline{0}]\!]^\pi_d \mid Mu)$$

where Mu stands for

$$\begin{array}{l}\overline{f}\langle e,d\rangle \mid !f(x,w).x(y,z).(z.Copy(w,b) \\ \quad \mid y.(d)(!\overline{d}(c,g).\overline{c}w \mid (e)([\![F^e_{d_1,..,d_k d}(\mathsf{f}(\widetilde{x},z))]\!]^\pi \mid \overline{f}\langle e,d\rangle))).\end{array}$$

By associating a particular recursive function to each computable function, the above encoding generates the relation $\mathcal{T}_\pi \stackrel{\text{def}}{=} \{(P,[\![P]\!]^\pi) \mid P \in \mathcal{P}_\mathbb{C}\}$. It should be clear that $\mathcal{T}_\pi$ is equipollent and closed under composition. It is not a bisimulation, the reason being that a computational descendant of $[\![P]\!]^\pi$ is not in the relation. But of course the composition $\mathcal{T}_\pi;=_\pi$ does not have that problem.

Theorem 5.3. *$\mathbb{C} \sqsubseteq \pi^M$ and $\mathbb{C} \sqsubseteq \pi$.*

5.2. Incompleteness Result

It turns out that some well-known process calculi fail the Axiom of Completeness. We show in this section that CCS and the process-passing calculus are incomplete, the reason being that CCS has a very weak interaction mechanism and the process-passing calculus has insufficient control power.

5.2.1. CCS

Milner's *Calculus of Communicating System* (Milner, 1980, 1989a), CCS for short, provides a prototype for models of interaction. The interaction primitive of CCS is simplified to bare minimal. All interactions in CCS are synchronization. Nothing is communicated in any interaction. There are a number of variants of CCS, most of which have been mentioned in (Milner, 1989a). Discussions about the relative expressiveness of CCS are given in (Palamidessi, 2003; Fu and Lu, 2010). The variant we shall be concerned with has the guarded choice. The unguarded choice, apart from being a non-congruent operator, introduces infinite branching (Fu and Lu, 2010) which cannot be implemented in practice. The set $\mathcal{P}_{\text{CCS}}$ is constructed from the following grammar:

$$P := \sum_{i\in I} \ell_i.P_i \mid D(a_1,\ldots,a_k)$$

where I is a finite indexing set and ℓ_i ranges over the set $\mathcal{N} \cup \overline{\mathcal{N}}$ for every $i \in I$, $\sum_{i\in I} \ell_i.P_i$ is a guarded choice, and $D(a_1, \ldots, a_k)$ is the instantiation of a parametric definition at the names $a_1, \ldots, a_k$. Using the action set $\mathcal{A}_{\mathrm{CCS}} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$, the semantics of CCS is defined by the following rules.

Choice

$$\frac{}{\sum_{i\in I} \ell_i.P_i \xrightarrow{\ell_i} P_i}\ i \in I$$

Interaction

$$\frac{P \xrightarrow{\ell} P' \qquad Q \xrightarrow{\overline{\ell}} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'}$$

Recursion

$$\frac{T\{\widetilde{c}/\widetilde{x}\} \xrightarrow{\lambda} P'}{D(\widetilde{c}) \xrightarrow{\lambda} P'}\ D(\widetilde{x}) = T.$$

Despite of the result proved in (Busi et al., 2003) that CCS is Turing complete, the intuition is however that CCS apparently lacks of the ability to pick up a value or to deliver a result in a proper manner. This intuition strongly suggests the following.

Theorem 5.4. CCS *is not complete.*

Proof. We prove that the numerals cannot be properly defined in CCS. Assume that there were a subbisimilarity $\mathfrak{F}$ from $\mathbb{C}$ to CCS. Let $\mathsf{s}(x)$ denote the successor function and $\uparrow$ the nowhere defined function. Let $B_0, B, C, B^{\uparrow}$ be such that the following conditions are met.

- $\overline{b}(\underline{0})\ \mathfrak{F}\ B_0 \not\rightarrow$,
- $F_b^c(\mathsf{s}(x))\ \mathfrak{F}\ B \not\rightarrow$,
- $F_c^b(\mathsf{s}(x))\ \mathfrak{F}\ C \not\rightarrow$,
- $F_b^c(\uparrow)\ \mathfrak{F}\ B^{\uparrow} \not\rightarrow$.

We will derive a contradiction in a few steps. The argument can be organized in the following fashion.

1. To start with it is easy to show that the absolute equality on the CCS-processes coincides with its external bisimilarity.

2. Suppose that $B_0 \,|\, B \xrightarrow{\tau} B'_0 \,|\, B'$ is caused by an interaction at a global name say d. Then $B_0 \,|\, B \xrightarrow{d}\!\xrightarrow{\overline{d}} B'_0 \,|\, B'$ and $B_0 \,|\, B \xrightarrow{\overline{d}}\!\xrightarrow{d} B'_0 \,|\, B'$. If $B_0 \,|\, B = B'_0 \,|\, B'$ then using (1) we could derive an infinite action sequence of the form
$$B_0 \,|\, B \overset{d}{\twoheadrightarrow}\overset{\overline{d}}{\twoheadrightarrow}\overset{d}{\twoheadrightarrow}\overset{\overline{d}}{\twoheadrightarrow} \ldots$$
and another infinite sequence of the form
$$B_0 \,|\, B \overset{\overline{d}}{\twoheadrightarrow}\overset{d}{\twoheadrightarrow}\overset{\overline{d}}{\twoheadrightarrow}\overset{d}{\twoheadrightarrow} \ldots.$$
But then $B_0 \,|\, B \,|\, B_0 \,|\, B$ would be divergent, contradicting to the fact that $\overline{b}(\underline{0}) \,|\, F^c_b(\mathsf{s}(x)) \,|\, \overline{b}(\underline{0}) \,|\, F^c_b(\mathsf{s}(x))$ is terminating. So every immediate internal action of $B_0 \,|\, B$ is a change-of-state action.
3. Let's assume that $B_0 \,|\, B^{\uparrow} \xrightarrow{\tau} B^1_0 \,|\, B^1$ for some B^1_0, B^1. Suppose that $B_0 \,|\, B^{\uparrow} \xRightarrow{\ell_1} \ldots \xRightarrow{\ell_k} B^k_0 \,|\, B^k$ is the longest external action sequence such that $B^k_0 \,|\, B^k$ may not perform any external action. It is clear that the action sequence $B_0 \,|\, B^{\uparrow} \xRightarrow{\ell_1} \ldots \xRightarrow{\ell_k} B^k_0 \,|\, B^k$ cannot be matched up by any action sequence of $B^1_0 \,|\, B^1$. Therefore $B_0 \,|\, B^{\uparrow} \xrightarrow{\iota} B^1_0 \,|\, B^1$, which must match up the change-of-state internal action $\overline{b}(\underline{i}) \,|\, F^c_b(\uparrow) \xrightarrow{\iota} \Omega$. It should be obvious that both B^1_0 and B^1 are unobservable.
4. Let L be such that $B \,|\, C \rightarrow^* L \not\rightarrow$. Suppose $B_2, \ldots, B_{2i}, \ldots$ satisfy the following properties:
$$\begin{array}{lll} B_0 \,|\, L & \rightarrow^*\xrightarrow{\iota}\rightarrow^*\xrightarrow{\iota}\rightarrow^* & B_2 \,\mathfrak{F}^{-1}\, \overline{b}(\underline{2}) \text{ and } B_2 \not\rightarrow, \\ & \vdots & \\ B_{2i} \,|\, L & \rightarrow^*\xrightarrow{\iota}\rightarrow^*\xrightarrow{\iota}\rightarrow^* & B_{2i+2} \,\mathfrak{F}^{-1}\, \overline{b}(\underline{2i+2}) \text{ and } B_{2i+2} \not\rightarrow, \\ & \vdots & \end{array}$$
For each $i \geq 1$ the process B_{2i+2} must be of the form $B'_{2i} \,|\, L'_{2i}$. We may apply the argument in (3) to B_{2i+2} to conclude that either B'_{2i} or L'_{2i} is unobservable.
5. Suppose the number of the observable L'_{2i}'s is finite. Then there must be some $k > 1$ such that $B'_{2k}, B'_{2k+2}, B'_{2k+4}, \ldots$ are all observable. Each process in the sequence contains one and only one subterm in summation form that may induce external actions for the process. It follows that there is an infinite subsequence $B'_{2k_1}, B'_{2k_2}, B'_{2k_3}, \ldots$ such that all

the external actions are induced by the subterms that are syntactically identical, the reason being that all these subterms are derivatives of either B or L. Due to dynamic binding it is possible that there are B'_{2k_i}, B'_{2k_j} such that the set of the external actions admitted by B'_{2k_i} is different from that of B'_{2k_j}. But clearly there must be an infinite subsequence $B'_{2k'_1}, B'_{2k'_2}, B'_{2k'_3}, \ldots$ in which all processes may perform the same set of external actions. Now the crucial point is that there must be two processes $B'_{2k'_i}, B'_{2k'_j}$, where $1 \leq k'_i < k'_j$, that enjoy the following property:

(†) For each action ℓ, $B'_{2k'_i}$ becomes observable after performing ℓ if and only if $B'_{2k'_j}$ becomes observable after performing ℓ.

Let $\mathsf{f}(x)$ be the following computable function

$$\textit{if}\, x < 2k'_j \textit{ then}\, \underline{0} \textit{ else}\, \text{diverge},$$

and let D be a CCS process satisfying $F^c_b(\mathsf{f}(x)) \mathfrak{F} D \nrightarrow$. Clearly D' is observable whenever $B_{2k'_i} \,|\, D \xrightarrow{\iota} D'$. By property (†) we must have that $B_{2k'_j} \,|\, D \xrightarrow{\iota} D''$ for some observable D''. But this contradicts to the fact that all the change-of-state actions of $B_{2k'_j} \,|\, D$ should lead to unobservable states.

6. If there are infinite observable L'_{2i}'s, then we can apply the argument in (5) to derive a similar contradiction. So this case is also impossible.

We conclude that $\mathbb{C} \not\sqsubseteq \text{CCS}$. □

The above proof refers neither to the choice operator nor to the finite branching property. Therefore it is also valid for the CCS with binary unguarded choice operator and parametric definition. This is the most expressive CCS variant studied in (Fu and Lu, 2010). So we have justified the general statement of Theorem 5.4.

5.2.2. Process-Passing Calculus

The process-passing calculus, called Λ-calculus in this paper, extends the lazy λ-calculus with the ability to interact externally. The set of the Λ-terms $\mathcal{T}_\Lambda$ is generated by the following grammar:

$$T := X \mid a(X).T \mid \overline{a}(T).T.$$

The term variable X is bound in $a(X).T$. The set $\mathcal{P}_\Lambda$ of the Λ-processes consists of those Λ-terms in which all the term variables are bound. The label set $\mathcal{L}_\Lambda$ is

$$\{a(L), (\widetilde{c})\overline{a}(L) \mid L \in \mathcal{P}_\Lambda \wedge \widetilde{c}, a \in \mathcal{N} \wedge a \notin \{\widetilde{c}\} \wedge \{\widetilde{c}\} \subseteq gn(L)\}.$$

In both $a(L)$ and $(\widetilde{c})\overline{a}(L)$, the interface name is a. The rules defining the operational semantics are given below.

Action

$$\frac{}{a(X).T \xrightarrow{a(L)} T\{L/X\}} \qquad \frac{}{\overline{a}(L).T \xrightarrow{\overline{a}(L)} T}$$

Interaction

$$\frac{S \xrightarrow{a(L)} T' \qquad T \xrightarrow{(\widetilde{c})\overline{a}(L)} T'}{S \,|\, T \xrightarrow{\tau} (\widetilde{c})(S' \,|\, T')}$$

Localization

$$\frac{T \xrightarrow{(\widetilde{c})\overline{a}(L)} T'}{(d)T \xrightarrow{(d)(\widetilde{c})\overline{a}(L)} T'}\ d \in gn(L) \setminus \{\widetilde{c}\}.$$

In our semantics we require that a term released in an output action must be a process. The operational semantics of the Λ-calculus is complicated by the fact that the transportation of a process might extend the scope of a localization operator.

The relationship between higher order π-calculus and first order π-calculus has been studied by Sangiorgi (1992, 1993). It has been argued that the Λ-calculus is at most as expressive as the π-calculus. Prior to Sangiorgi's work, Thomsen (1989) has discussed the relationship between CHOCS, the calculus of higher order communicating systems, and the π-calculus. A related work is described in (Amadio, 1993). The Sangiorgi-Thomsen's translation of the Λ-calculus in the π-calculus is based on an injective function $st : \mathcal{V}_p \to \mathcal{V}_n$. We assume that $st(X) = x$ and so on. The extensional translation is defined as follows:

$$\begin{aligned}
[\![X]\!]_{st} &\stackrel{\text{def}}{=} x, \\
[\![a(X).T]\!]_{st} &\stackrel{\text{def}}{=} a(x).[\![T]\!]_{st}, \\
[\![\overline{a}(T).T']\!]_{st} &\stackrel{\text{def}}{=} (c)(\overline{a}c.[\![T']\!]_{st} \,|\, !\overline{c}.[\![T]\!]_{st}).
\end{aligned}$$

The encoding generates an equipollent, extensional, codivergent bisimulation by composing with $=_\pi$. Is the relation sound? We invite the reader to look for the answer.

Can the π-calculus be interpreted in Λ? Our intuition tells us that the answer is negative. In the π-calculus a received name can be used in a prefix, whereas in the Λ-calculus a received process cannot act as a prefix of anything. In other words, the Λ-calculus has far less control power than the π-calculus. This intuition is justified by the next theorem.

Theorem 5.5. *The Λ-calculus is not complete.*

Proof. The proof is similar to the proof of Theorem 5.4. Assume there were a subbisimilarity $\mathfrak{F}$ from the $\mathbb{C}$-calculus to the Λ-calculus. We derive a contradiction in a few steps.

1. Let B_i, B be such that $\overline{b}(\underline{i}) \mathfrak{F} B_i \nrightarrow$ and $F_b^c(\mathsf{s}(x)) \mathfrak{F} B \nrightarrow$. Assume that $B_i \,|\, B \xrightarrow{\tau} C_1$ is caused by an interaction between B_i and B at a global name d. If $B_i \,|\, B = C_1$ then $C_1 \rightarrow^* C_2$ for some C_2 such that C_2 does not admit any computation. It is easy to see that C_2 must be able to perform both an input action at d and an output action at d. These two complementary actions contribute to an internal action. Continuing in this way one could construct an infinite sequence of internal actions. This renders a contradiction since $\overline{b}(\underline{i}) \mid F_b^c(\mathsf{s}(x))$ does not admit any infinite internal action sequence. We conclude that every immediate internal action of $B_i \,|\, B$ is a change-of-state internal action.
2. Let $B^\uparrow$ be such that $F_b^c(\uparrow) \mathfrak{F} B^\uparrow \nrightarrow$. As is done in (1) one can prove that the immediate internal action of $B_i \,|\, B^\uparrow$ must be a change-of-state action. Consequently C_0 must be unobservable whenever $B_i \,|\, B^\uparrow \xrightarrow{\tau} C_0$.
3. Since there is no choice operator in the Λ-calculus, it follows from (2) that only one prefix term in B_i is fireable. Similarly only one prefix term in $B^\uparrow$ may act.
4. Now let $L_i, L^\uparrow$ be such that $\overline{g}(\underline{i}) \mathfrak{F} L_i$ and $F_g^h(\uparrow) \mathfrak{F} L^\uparrow$. The global name at which the processes $L_i, L^\uparrow$ interact must be different from the global name at which the processes $B_i, B^\uparrow$ interact. Otherwise $B_i \,|\, L^\uparrow$ would be able to perform a change-of-state action, rendering a contradiction. The consequence is that we can always choose a name say d such that whenever $\overline{d}(\underline{i}) \mathfrak{F} D$ then the only external action of D is carried out at a name that is not in any specific finite name set.

5. Suppose $B_i \xrightarrow{b(A'')} S_i\{A''/X\}$ and $B \xrightarrow{(\widetilde{d})\overline{b}(A'')} A'$. It should be clear that

$$B_i \,|\, B \xrightarrow{\iota} (\widetilde{d})(S_i\{A''/X\} \,|\, A') \tag{18}$$

is bisimulated by $\overline{b}(\underline{i}) \,|\, M^c_b(\mathsf{s}(x)) \xrightarrow{\iota} \overline{c}(\underline{i+1})$. Now let S'_i be such that $S_i \Longrightarrow S'_i$ and that S'_i cannot perform any internal action. Clearly $(\widetilde{d})(S_i\{A''/X\} \,|\, A') \to^* (\widetilde{d})(S'_i\{A''/X\} \,|\, A')$. By structural induction it is easy to prove that

$$S'_i\{A''/X\} = S''_i\{A''/X\} \,|\, \prod_{k_i} A''$$

for some $k_i \geq 0$ and some S''_i in which X is guarded. We can rewrite (18) to

$$B_i \,|\, B \overset{\iota}{\twoheadrightarrow}= (\widetilde{d})(S''_i\{A''/X\} \,|\, \prod_{k_i} A'' \,|\, A').$$

For a number j that differs from i, we also have

$$B_j \,|\, B \overset{\iota}{\twoheadrightarrow}= (\widetilde{d})(S''_j\{A''/X\} \,|\, \prod_{k_j} A'' \,|\, A')$$

for some k_j and some S''_j in which X is guarded. The external actions of

$$(\widetilde{d})(S''_i\{A''/X\} \,|\, \prod_{k_i} A'' \,|\, A') \text{ and } (\widetilde{d})(S''_j\{A''/X\} \,|\, \prod_{k_j} A'' \,|\, A')$$

must be caused by the components $\prod_{k_i} A'' \,|\, A'$ and $\prod_{k_j} A'' \,|\, A'$ respectively. Without loss of generality we assume $k_i < k_j$. Consider the process M satisfying $F^f_c(\mathsf{f}_{\underline{i+1}\to\underline{i+1}}(x)) \mathfrak{F} M \not\twoheadrightarrow$. We have the following

$$(\widetilde{d})(S''_i\{A''/X\} \,|\, \prod_{k_i} A'' \,|\, A') \,|\, M \to^* \xrightarrow{\iota} M' \mathfrak{F}^{-1} \overline{f}(\underline{i+1}).$$

According to (4) we can choose f in such a way that the next external action of M' is caused by a descendant of M. But then we must have

$$(\widetilde{d})(S''_j\{A''/X\} \,|\, \prod_{k_j} A'' \,|\, A') \,|\, M \to^* \xrightarrow{\iota} \Downarrow,$$

contradicting to the definition of $\mathsf{f}(x)$. So this case is impossible.

6. Suppose $B_i \overset{(\widetilde{d_i})\overline{b}(A_i)}{\longrightarrow} B'_i$ and $B \overset{b(A_i)}{\longrightarrow} T\{A_i/X\}$. It should be clear that

$$B_i \,|\, B \overset{\iota}{\longrightarrow} (\widetilde{d_i})(B'_i \,|\, T\{A_i/X\}) \tag{19}$$

is bisimulated by $\overline{b}(\underline{i}) \,|\, M^c_b(\mathsf{s}(x)) \overset{\iota}{\longrightarrow} \overline{c}(\underline{i+1})$. According to (4) we may choose the name c in such a way that the immediate external action of $(\widetilde{d_i})(B'_i \,|\, T\{A_i/X\})$ is carried out at a name different from any name appeared in B_i. Consequently the next external action of $(\widetilde{d_i})(B'_i \,|\, T\{A_i/X\})$ must be caused by T. It follows from (19) that

$$B_{i+1} \,|\, B \,|\, M \overset{\iota}{\longrightarrow} (\widetilde{d_{i+1}})(B'_{i+1} \,|\, T\{A_{i+1}/X\}) \,|\, M \rightarrow^* \overset{\iota}{\longrightarrow} M'' \Downarrow$$

for some M'', where M is the process introduced in (5). By (5) the change-of-state action of $(\widetilde{d_{i+1}})(B'_{i+1} \,|\, T\{A_{i+1}/X\}) \,|\, M$ is caused by M performing an input action. By assumption the external action of M'' is caused by a descendant of M. It is easy to see from these facts that

$$B_{i+2} \,|\, B \,|\, M \overset{\iota}{\longrightarrow} (\widetilde{d_{i+2}})(B'_{i+2} \,|\, T\{A_{i+2}/X\}) \,|\, M \rightarrow^* \overset{\iota}{\longrightarrow} \Downarrow,$$

contradicting to the definition of $\mathsf{f}_{\underline{i+1} \rightarrow \underline{i+1}}(x)$.

We conclude that $\mathbb{C} \not\sqsubseteq \Lambda$. $\square$

It is shown in (Lanese et al., 2010) that the abstraction-passing calculi (Sangiorgi, 1992) are strictly more expressive than the process-passing calculi. Similar result has been obtained by Xu (2012) for higher order calculi with relabeling. It remains to see if any of these stronger higher order process calculi is complete without incorporating some kind of name-passing communication mechanism.

5.3. Computability Model Justified

The Computability Model $\mathbb{C}$ plays a fundamental role in Theory of Interaction. Important issues about $\mathbb{C}$ must be addressed in a satisfactory way. Two inevitable questions are:

1. Is $\mathbb{C}$ strong enough?
2. Is $\mathbb{C}$ too strong?

The Axiom of Completeness places a lower bound on the models of interaction. It is stronger than the following form of the Church-Turing Thesis: All computable functions are definable in a (computation) model. The Axiom of Completeness is strictly stronger because it also takes into account of the external communications. Another aspect of Church-Turing Thesis, stating that the functions definable in all physically implementable computation models are computable functions, is not captured by the Axiom of Completeness. This is reasonable since concurrent models, like the non-deterministic Turing machine model, are not believed to be implementable physically. From the point of view of computation $\mathbb{C}$ is certainly strong enough. Is $\mathbb{C}$ strong enough from the viewpoint of interaction? This question, together with the second question raised in the above, are about the interaction mechanism of $\mathbb{C}$. A functional process in $\mathbb{C}$ is capable of inputting a value and then outputting another one. There is no doubt that it places only a minimal requirement on the interactability. What can be questioned is the capacity of the interactability admitted to $\mathbb{C}$. A process in $\mathbb{C}$ can send to another process a piece of message of any size, coded up by a number, in the twinkling of an eye. This is not what happens in most models! But thanks to Theorem 3.6 the Axiom of Completeness does not force this unbounded-message-size communication capacity on any complete model. As far as the functional processes and the value processes are concerned, the axiom only talks about what should be communicated, not how it should be accomplished. This is so because $\mathbb{C}$ does not have the localization operator. So now comes the big question: Why does $\mathbb{C}$ not have the localization operator? Wouldn't the Axiom of Completeness be a better axiom had $\mathbb{C}$ had the operator? A technical answer to these questions is that it wouldn't work. The new model would become so different that it is incompatible with any of the three prime models introduced in Section 2. Recall that $\mathbb{A}$ denotes the model $\mathbb{C}$ extended with the localization operator whose operational semantics is defined by the rule given in (3). We have the following negative results.

Proposition 5.6. $\mathbb{A} \not\sqsubseteq \pi$.

Proof. Assume that the π processes C, D are the interpretations of $F_a^b(\mathsf{f}_{id})$, $F_b^a(\mathsf{f}_{id})$ respectively. Then $(b)(F_a^b(\mathsf{f}_{id}) \,|\, F_b^{ab}(\mathsf{f}_{id}))$ would be the interpretation of $(b)(C \,|\, D)$. Using the analysis given in the proof of Theorem 4.9, one derives that $(b)(C\,|\,D)\,|\,(b)(C\,|\,D) \neq_\pi (b)(C\,|\,D)$. However

$$(b)(F_a^b(\mathsf{f}_{id})\,|\,F_b^{ab}(\mathsf{f}_{id}))\,|\,(b)(F_a^b(\mathsf{f}_{id})\,|\,F_b^{ab}(\mathsf{f}_{id})) =_{\mathbb{A}} (b)(F_a^b(\mathsf{f}_{id})\,|\,F_b^{ab}(\mathsf{f}_{id}))$$

according to Proposition 3.7. We conclude that the soundness condition cannot be satisfied. □

Proposition 5.7. $\mathbb{A} \not\sqsubseteq \mathbb{VPC}$.

Proof. First observe that a value process $\overline{a}(\underline{i})$ cannot be interpreted by a term that can do an input action. Otherwise the interpretation of a functional process of the form $F_a^b(\mathsf{f}_{id})$ can only do output actions at channel a. The number of the output actions it can do must be finite. A contradiction can then be easily derived. So $\overline{a}(\underline{i})$ must be interpreted by a process A that can only do output actions at a. Let C be an interpretation of $F_a^b(\mathsf{f}_{id})$. The interaction $(b)C\,|\,A \xrightarrow{\tau} B$, for any possible B, must be a nondeterministic computation step, otherwise $(b)C\,|\,A\,|\,(b)C\,|\,A$ would be capable of doing an infinite computation. We conclude that B is essentially $\mathbf{0}$ and consequently $\overline{a}(\underline{i})$ is essentially interpreted by $\overline{a}(\underline{i'})$ for some $\underline{i'}$. But then it is easy to see that $C' = \overline{b}(\underline{k})$ whenever $C \xrightarrow{a(\underline{k})} C'$. We conclude that $(b)(C\,|\,D)\,|\,(b)(C\,|\,D) \neq_{\mathbb{VPC}} (b)(C\,|\,D)$, where D is an interpretation of $F_b^a(\mathsf{f}_{id})$. Clearly the soundness fails because of Proposition 3.7. □

Corollary 5.8. $\mathbb{A} \not\sqsubseteq \mathbb{IM}$.

The above results rule out the possibility for $\mathbb{A}$ to play any role in the Axiom of Completeness. What is wrong with it? The localization operator brings out the asynchrony of the atomic $\mathbb{A}$-processes, which has been deliberately suppressed in $\mathbb{C}$. One could try to introduce some kind of sequential operation into $\mathbb{A}$ to remove the asynchrony. But that would solve one problem by introducing a bunch of others. If we introduce for example the output prefix operator in $\mathbb{A}$, we end up in a situation where the asynchronous calculi like the asynchronous π are ruled out since the output prefix operator cannot be properly interpreted by the asynchronous output. All these technical results suggest that $\mathbb{C}$ should not be made any stronger.

As a theoretical model $\mathbb{C}$ should keep neutral to any implementation method as well as to any programming style. Adding an additional operator would sacrifice the neutrality, and at the meantime would deny the models that do not pertain to a particular paradigm.

6. Related Work

The development of the process theory can be summarized in a number of ways. A handy approach is to classify the stages of the development by the influential models that have been studied in the process theory. Our understandings of the theories of equality, expressiveness, and completeness have advanced significantly with the investigations into each of these models.

The influence of CCSP model, a shorthand for Milner's CCS (Milner, 1980, 1989a) and Hoare's CSP (Hoare, 1978, 1985), reaches to almost every corner of the theory. Apart from the introduction of the fundamental operators of the process theory (**0**, concurrent operator, localization operator), the studies of CCSP have led to a theory of process equality (Milner, 1981, 1984, 1989b; Hennessy and Milner, 1985; Roscoe, 1997) that has been successfully applied/extended to other process models. The results established in the equality theory have been exploited in Bergstra and Klop's ACP (Baeten and Weijland, 1990) that features an axiomatic approach. There are a number of decidability results Hirshfeld and Jerrum (1999) for the variants of CCS/ACP (Hirshfeld and Jerrum, 1999; Mayr, 2000; Burkart et al., 2001; Srba, 2004; Kučera and Jančar, 2006; Jančar and Srba, 2008; Czerwiński et al., 2011; Fu, 2013a; Yin et al., 2014), some of which can be used to establish separation results. However a proper investigation into the relative expressiveness of the variants of CCS has been a recent endeavor (Busi et al., 2003; Giambiagi et al., 2004; Busi et al., 2004; Busi and Zavattaro, 2004; Palamidessi, 2003; Fu and Lu, 2010). The characterization by Fu and Lu (2010) of the expressiveness of CCS variants remains valid if we use the subbisimilarity defined in this paper as criterion.

In reality processes do not just synchronize, they pass messages from one to another. The value-passing calculi, in which the messages are coded up by numerals, were introduced in the CCSP framework (Milner, 1989a; Hoare, 1985). A systematic studies of the value-passing mechanism have been carried out by Hennessy and his collaborators (Hennessy, 1991; Hennessy and Ingólfsdóttir, 1993a,b; Hennessy and Lin, 1995, 1996; Rathke, 1997; Hennessy and Lin, 1997; Hennessy et al., 1997). It was discovered in these studies that the symbolic approach (Hennessy and Lin, 1995; Ingólfsdóttir and Lin, 2001) plays an important role in analyzing the branching structures of the value-passing calculi. In all these studies, the value domains are treated as oracles. This isolation of concern facilitates the development of the equality theory. It is however not at all helpful to the expressiveness study. A self-contained treatment of the theory of the value-passing calculi is carried out in a satellite paper by Fu (2013b).

The search for the λ-calculus of the concurrent computations has resulted in an influential model, the π-calculus of Milner, Parrow and Walker (Milner et al., 1992; Parrow, 2001; Sangiorgi and Walker, 2001b). Due to its name-passing communication mechanism, the conditional processes in the π-calculus can be nicely defined in terms of the match and mismatch operators. The studies in the name-passing paradigm (Parrow and Sangiorgi, 1995; Boreale and De Nicola, 1995; Lin, 1995a,b; Sangiorgi, 1996c; Lin, 1996, 1998, 2003; Fu and Yang, 2003; Fu, 2005) clarify the indispensable role of the localization operator and its tricky presence in the original formulation of the π-calculus. The investigations into the variants of the π-calculus (Honda and Tokoro, 1991a,b; Boudol, 1992; Amadio et al., 1998; Sangiorgi, 1996b; Boreale, 1996; Fu, 1997; Parrow and Victor, 1997, 1998; Fu, 1999; Merro, 2000; Merro and Sangiorgi, 2004; Fu, 2003) brings out the agility of the name-passing mechanism. The proposal of the barbed bisimilarity by Milner and Sangiorgi (1992) for the name-passing calculi is a major step in the study of the theory of equality. The research in the π-calculus has also advanced the theory of expressiveness. Milner's encoding of the lazy λ-calculus (Milner, 1992) has inspired several follow-up works that relate the operational and the observational semantics of the computation model to those of the process models (Sangiorgi, 1994, 1995; Fu, 1999; Merro, 2000; Merro and Sangiorgi, 2004; Cai and Fu, 2011). Sangiorgi's translation of the higher order process calculi (Sangiorgi, 1993, 1996a,b, 2001) into the π-calculus significantly improves our understanding of the relationship between higher order process calculi and the π-calculus. The object oriented style of the π-programming, a notable feature of the works of Milner and Sangiorgi, is emphasized by Walker's interpretation of the object oriented languages (Walker, 1991, 1995). All encodings into the π-calculus have to address the issue of full abstraction. The most recent attempt to attack the problem is the fully abstract encoding of the full λ-calculus by Cai and Fu (2011). The theory of expressiveness has been significantly enriched by the works on the relative expressiveness of the various variants of the π-calculus (Honda and Tokoro, 1991a,b; Amadio et al., 1998; Nestmann and Pierce, 1996; Merro, 2000; Nestmann, 2000; Palamidessi, 2003; Merro and Sangiorgi, 2004; Cacciagrano et al., 2006; Palamiddessi et al., 2006; Cacciagrano et al., 2008) and the works (Gorla, 2008b,a, 2009a,b; Fu, 2007) that compare the π-calculus to other process models, in particularly the variants (Levi and Sangiorgi, 2000; Guan et al., 2001; Phillips and Vigliotti, 2002; M. Bugliesi and Crafa, 2004; Merro and Zappa Nardelli, 2005; Merro and Hennessy, 2006; Fu, 2007) of the Ambient Calculus (Cardelli and Gordon, 2000). In retrospect one cannot help remarking that the equality theory of the π-calculus has been unnecessarily complicated, and the research on the expressiveness theory of the name-passing calculi has been slowed down, by the confusion of the name variables with the names. An elaboration on the problems caused by the confusion can be found in (Fu and Zhu, 2015).

The power of the barbed bisimilarity is duly exhibited in Sangiorgi's study of the higher order calculi (Sangiorgi, 1992, 1996a). There could be several variations when defining a bisimulation equivalence for the process-passing calculi (Thomsen, 1989, 1990; Sangiorgi, 1992; Thomsen, 1993, 1995), but it is the barbed bisimilarity that tells us which one should be preferred. It does not take much long for one to get the feeling that the process-passing mechanism has too weak a control power over the interactions to carry out the fundamental constructions possible in the π-calculus. Sangiorgi's encoding of the lazy λ-calculus in the higher order π-calculus (Sangiorgi, 1993) for example has to resort to the constant definitions parametric on either names or processes. These parametric definitions are not only used locally, they are also passed from one process to another, which necessarily means that typing information has to be provided since different definitions have different numbers of parameter. Both the work of Thomsen on the higher order CCS and Sangiorgi on the higher order π-calculus suggest that the process-passing mechanism is best used along with the name-passing mechanism, which again raises the issue of type system. Having to introduce a type discipline is one way to say that the higher order mechanism belongs to programming theory rather than to model theory.

In the rest of the section, we take a closer look at the related criteria. Our discussions will be guided by the four principles. The practical aspect of the criteria will be commented on in Section 6.4.

6.1. On Equality

The bisimulation and the codivergence are *intensional* conditions. Intensional conditions are concerned with computations. On the other hand, the extensionality and the equipollence are *extensional* conditions. Extensional conditions are to do with interactions. If we completely ignore the intensional conditions, we may come up with the following definition.

Definition 23. The *extensional equality* $=_e^{\mathbb{M}}$ on $\mathbb{M}$ is the largest equivalent, extensional and equipollent relation on $\mathcal{P}_{\mathbb{M}}$.

Definition 23 is the model independent formulation of the trace equivalence. See (Fu and Zhu, 2015) for more on this topic. The so-called observational equivalences are obtained by strengthening Definition 23 with additional intensional requirements. The strong equality and the weak equality are two examples. So is the testing equivalence (Fu and Zhu, 2015). The advantage of this way of thinking about the observational equivalences is that it only produces model independent definitions. It is beyond the scope of this paper to discuss which of the equivalence relations proposed in literature (van Glabbeek, 1993b, 2001) have model independent characterizations. What we do in this subsection is to comment on the major conceptual and technical contributions that relate to the present work.

6.1.1. Bisimulation

In algebraic theory bisimulation is a technique to construct the largest fixpoints. The concept was introduced to concurrency theory by Park (1981) and Milner (1989a). For a long time bisimulation was mainly perceived as a requirement about external actions. Many variations were proposed and studied (van Glabbeek, 2001, 1993b). It is the barbed bisimulation of Milner and Sangiorgi (1992) that brings out the power of the bisimulation of internal actions. This is a first step towards a model independent theory of process. The barbed approach, and the reductional semantics, have become a standard tool in process theory (Sangiorgi and Walker, 2001b). It is emphasized in (Fu and Lu, 2010; Fu, 2015) that the notion of bisimulation is implicit in Church-Turing Thesis (van Emde Boas, 1990). To prove a Turing completeness result amounts to establishing an *effective* bisimulation (Fu, 2015) of computation from a known computation model to the target model. Now the key point is that a bisimulation of internal action is not the same as a bisimulation of deterministic computation. A deterministic computation is a local manipulation that does not change the ability to interact, whereas a nondeterministic computation does change the interactability. The correct definition of the bisimulation of internal action is the essence of the branching bisimulation introduced by van Glabbeek and Weijland (1989). The particular formulation given in Definition 3 is due to Baesten (Baeten, 1996), which has the edge over the original formulation of van Glabbeek and Weijland (1989) in that the composition of two branching bisimulations is still a branching bisimulation. The advantage of the branching bisimulation over the weak bisimulation has been demonstrated in a series of papers by van Glabbeek and his collaborators (van Glabbeek and Weijland, 1989; van Glabbeek, 1993a,b, 1994; van Glabbeek et al., 2009). The branching bisimulation also appears as a more stable property than the weak bisimulation. This is manifested in a number of works.

1. De Nicola et al. (1990) have argued that the branching bisimilarity is the least fixpoints of the operators that admit both forward bisimulation as well as some kind of backward bisimulation.
2. De Nicola and Vaandrager (1995) have shown that there is a nice logical characterization of the branching bisimilarity using an additional until operator,
3. Baier and Hermanns (1997) have pointed out that for the fully probabilistic systems branching bisimulation is the only choice.
4. Czerwiński et al. (2011) and Fu (2013a) have provided convincing examples demonstrating that the branching bisimilarity is a lot more tractable than the weak bisimilarity.

The importance of the branching bisimulation is best appreciated when computations are considered as first class citizens.

One benefit of combining the barbed approach and the branching approach is that one only needs to talk about bisimulations of internal actions. Bisimulations of external actions are derived properties.

6.1.2. Codivergence

The theory of process calculus has been criticized for not paying enough attention to divergence. It was ignored completely in the trace equivalence of CSP (Hoare, 1978, 1985) and the bisimulation equivalence of CCS (Milner, 1989a). In the denotational treatment a process that can only diverge is understood as the bottom element of the domain (Amadio and Curien, 1998) of processes. The denotational approach identifies divergence to zero information or undefinedness. If the information order is mingled with the observational theories, different order relations on processes are produced. In the testing scenario of De Nicola and Hennessy (De Nicola and Hennessy, 1984; Phillips, 1987; Hennessy, 1988), the must equivalence regards divergence as catastrophic. This is rectified in later modifications of the testing equivalence (Brinksma et al., 1995; Natarajan and Cleaveland, 1995; Boreale et al., 1999, 2001). In the bisimulation semantics (Walker, 1990; Aceto and Hennessy, 1992), divergence is respected but not bisimulated. In process algebra (Baeten and Weijland, 1990), divergence is an operator defined (in)equationally. The denotational approach features a preordained predicate, often denoted by $\Uparrow$, that is defined on the syntax of processes, and an inactive process, denoted by $\perp$ or Ω or Δ by various authors and called *explicit divergence*, that induces a satisfaction relationship to the predicate.

Divergence is neither a syntactical concept nor an observational one. Its treatment should be intensional rather than extensional. This is consistent with the view that bisimulation as given in Definition 3 is a computational, and consequently an intensional property. The preferable way of defining divergence is in terms of infinite internal action sequences. An equality is divergence sensitive if it respects divergence. The simplest formulation of divergence sensitivity is the so-called termination preserving property. In (van Glabbeek, 1993b) the author summarizes several ways to strengthen the termination preserving property. The operational approach to divergence had been left without scrutiny for some time. It is Lohrey et al. (2002, 2005) who give for the first time an algebraic characterization of the divergence sensitive equivalences classified by van Glabbeek. Termination preservation is obviously a correct criterion if it is stated for the deterministic computations. It is a dubious one when stated for the general computations including the nondeterministic computations. The right property that goes nicely with the bisimulation property is the codivergence heavily used by Fu and Lu (2010). Independently van Glabbeek, Luttik and Trčka have studied in a recent paper (van Glabbeek et al., 2009) the codivergence property of the branching bisimulations. Codivergence appears in (van Glabbeek et al., 2009) as the weakest condition among several alternative formulations of the same idea. But Bisimulation Lemma clearly implies that it is as effective as the strongest formulation one may think of. As far as we know the codivergence property was stated for the first time by Priese (Priese, 1978), who called it *eventually progressing property.* The equational axioms for codivergence is studied in (Fu, 2015), which is inspired by the work of Lohrey et al. (2005).

6.1.3. Extensionality

The observation theory of process is complicated by our perception that there are an almost unbounded number of choices of the observers. If the observers are external, meaning that they stay outside the model of the observees, then they may vary considerably in strength. But if the observing power of the external observers, chosen for whatever reasons, are too strong or too weak compared to the power of the observees, isn't that suggesting that a wrong model has been chosen? We might as well start with a stronger or weaker model. If a process calculus defines a closed model of processes, which is generally accepted, then the observers ought to be internal ones. The proper assumption is that all processes of a model are observers, which leads to the environment closure property of the barbed equivalence (Milner and Sangiorgi, 1992). If one generalizes this closure property from a relation on one model to a relation from one model to another, one gets the extensionality criterion. The simple step from the environmental closure condition to the extensionality condition is what makes possible the unification of the equality and the expressiveness relationship. The tradeoff is that generally one has to settle for *a* subbisimilarity rather than *the* subbisimilarity.

It has been suggested that a composition process $P \mid Q$ could be interpreted as $C[\llbracket P \rrbracket, \llbracket Q \rrbracket]$ for some nontrivial context $C[\_, \_]$ in target model. Theoretically such a translation is understood as saying that $\llbracket P \rrbracket$ or $\llbracket Q \rrbracket$ has been wrongly interpreted. In applications necessity for this kind of translation occurs. That is a different issue.

6.1.4. Equipollence

Extensionality alone is insufficient to recover an observational equivalence. A counter example is the extensional closure of $\{(!a \mid !\overline{a}, !b \mid !\overline{b})\}$. To capture the full power of observation, one needs to talk about interactability. This is what Milner and Sangiorgi's barbed approach has offered. In the original paper of Milner and Sangiorgi (1992), barbedness means observability. The idea was embraced in a lot of later studies and has clarified the relative roles of some of the equivalences (Sangiorgi and Walker, 2001a). In most of these studies one defines a barb as for example a subject name of an action or an external action, and then define the barbed equivalence accordingly. These variations on the barbed equivalence are not as model independent as Milner and Sangiorgi's original definition. When generalizing relations on one model to relations between two models, it is the observability, not any definition of barbs, that survives.

The barbed approach has promoted a shift from the labeled semantics to the reductional semantics (Berry and Boudol, 1992; Milner, 1992; Honda and Yoshida, 1995). While we acknowledge the importance of the reductional approach, we must point out that it is rather easy to design a reductional semantics that cannot be justified from the perspective of interaction. Sometimes one ends up with just a term rewriting system.

6.2. On Expressiveness

Nestmann (2000), Palamidessi (2003) and Parrow (2006) have discussed several expressiveness criteria proposed in literature. Gorla (2008a) has done significant work on the expressiveness of process calculi. Gorla's model independent approach differs from ours in that his criteria are more application oriented whereas ours are purely theoretical. This point will be further elaborated in Section 6.4. In this subsection we examine a number of criteria in the light of Theory of Interaction. Suppose that there are two translations:

$$\mathbb{M}_0 \xrightarrow{\mathcal{T}} \mathbb{M}_1 \xrightarrow{\mathcal{T}'} \mathbb{M}_2,$$

where $\mathcal{T}$ asserts that $\mathbb{M}_1$ is at least as expressive as $\mathbb{M}_0$ and $\mathcal{T}'$ asserts that $\mathbb{M}_2$ is at least as expressive as $\mathbb{M}_1$. Whatever the criteria are taken for the assertions, it must follow that $\mathbb{M}_2$ is at least as expressive as $\mathbb{M}_0$. In sequel we will reject any criterion that does not enjoy transitivity.

Every criterion for expressiveness should refer to some equivalence relation. For simplicity we shall only use the weak equality, and accordingly weak bisimulations, in the following discussions.

6.2.1. Leader Election Problem

If there is a problem that can be solved in $\mathbb{L}$ but is unsolvable in $\mathbb{L}'$, then the problem separates $\mathbb{L}$ from $\mathbb{L}'$ since the latter cannot be more expressive than the former. The Leader Election of distributed systems (Garcia-Molina, 1982; Stoller, 2000) is one such problem that is often used to tell apart process calculi (Bougé, 1988; Palamidessi, 2003; Phillips and Vigliotti, 2006; Vigliotti et al., 2007; Phillips and Vigliotti, 2008). In the theory of distributed computing an important issue is about reaching a consensus. The problem looks for a method to reach an agreement, say a number less than k, among a set of k processes. It has been proved that such a method does not exist for the general consensus problem if processes may fail. The problem does have a solution if we assume that no participating processes can ever fail or that there is a timeout mechanism. In the restricted scenario, a simple leader election algorithm suffices. In process theory failure is interpreted as divergence. The Leader Election Problem assumes that all the negotiating processes are terminating. It is however not an easy task to prove that a problem is unsolvable in a particular process model.

Solutions to the Leader Election Problem are often required to satisfy additional properties like distribution, symmetry and stability (Palamidessi, 2003). It has been suggested that adding more structures is one way to achieve sharper separation results (Vigliotti et al., 2007). Some of the additional structures are highly syntactical. Results along this line have more implications to distributed computing than to process theory. A proof by counter example, say the Leader Election Problem, can be couched in the following formal terms:

1. Single out a process P in $\mathbb{L}$.
2. Assume that there were a subbisimilarity $\mathfrak{F}$ from $\mathbb{L}$ to $\mathbb{L}'$.
3. Derive a contradiction from the assumption $\exists Q.P\mathfrak{F}Q$.
4. Conclude that $\mathbb{L} \not\sqsubseteq \mathbb{L}'$.

The steps (1-4) are the standard procedure to prove a negative result. When working in a formal setting as strong as Theory of Interaction, the counter examples are often simple and straightforward. For instance it is easy to show that the mixed choice $a + \overline{b}$ cannot be interpreted in the π-calculus with only the separated choices (Fu and Lu, 2010). This counter example is much simpler than the Leader Election Problem.

Let's see another example. Is the random number generator definable in say $\mathbb{VPC}$? Take a look at the $\mathbb{VPC}$-process *Rng* defined as follows:

$$Rng \overset{\text{def}}{=} (c)(\overline{c}(\underline{0}) \mid !c(x).(\tau.\overline{c}(\mathsf{succ}(x)) + \tau.\overline{b}(x))).$$

It is clear that $Rng \overset{\tau}{\Longrightarrow} \xrightarrow{\overline{b}(\underline{k})} = \mathbf{0}$ for every k. The number k is picked up randomly due to the nondeterminism caused by the composition operator. But notice that *Rng* is divergent. So at best it is a divergent implementation of the random number generator. Can we improve upon the solution offered by *Rng*? The answer is negative.

Lemma 6.1. *There does not exist a terminating* $\mathbb{VPC}$*-process RN such that all its action sequences are of the form* $RN \Longrightarrow \xrightarrow{\overline{b}(\underline{k})} = \mathbf{0}$ *and for every* k *one has that* $RN \Longrightarrow \xrightarrow{\overline{b}(\underline{k})} = \mathbf{0}$.

Proof. According to the termination condition, RN cannot contain two replication sub-processes that can interact. But then RN must be finite branching and consequently König Lemma implies that RN is only capable of generating a finite number of natural numbers. □

6.2.2. Operational Correspondence

Two folklore criteria for the operational faithfulness of the translation is given in the next two definitions.

Definition 24. The translation $\mathcal{T}$ is *operationally complete* if $P \xrightarrow{\tau} P'$ implies that $\mathcal{T}(P) \Longrightarrow Q' =_w^{\mathbb{M}_1} \mathcal{T}(P')$ for some $Q' \in \mathcal{P}_{\mathbb{M}_1}$.

Definition 25. The translation $\mathcal{T}$ is *operationally sound* if $\mathcal{T}(P) \xrightarrow{\tau} Q'$ implies that $P \Longrightarrow P'$ and $Q' =_w^{\mathbb{M}_1} \mathcal{T}(P')$ for some $P' \in \mathcal{P}_{\mathbb{M}_0}$.

To put things in perspective, suppose $\mathcal{T}$ and $\mathcal{T}'$ are operationally sound and complete. Let's see what is necessary to establish the operational completeness and soundness of the composition $\mathcal{T};\mathcal{T}'$.

- It is easy to see that in order to prove the completeness of $\mathcal{T};\mathcal{T}'$, we generally need the following soundness property:

  - If $P =_w^{\mathbb{M}_1} Q$ then $\mathcal{T}'(P) =_w^{\mathbb{M}_2} \mathcal{T}'(Q)$.

  We also need to make use of the fact that $=_w^{\mathbb{M}_2}$ is a weak bisimulation.

- The soundness of $\mathcal{T};\mathcal{T}'$ is problematic. Suppose $\mathcal{T}'(\mathcal{T}(A)) \xrightarrow{\tau} A_2$ for some $A \in \mathcal{P}_{\mathbb{M}_0}$. By the soundness of $\mathcal{T}'$, some A_1 exists such that $\mathcal{T}(A) \Longrightarrow A_1$ and $\mathcal{T}'(A_1) =_w^{\mathbb{M}_2} A_2$. Without loss of generality, assume that $\mathcal{T}(A) \Longrightarrow A_1$ consists of two internal steps, say $\mathcal{T}(A) \xrightarrow{\tau} A_1' \xrightarrow{\tau} A_1$. By the soundness of $\mathcal{T}$, some A_0 exists such that $A \Longrightarrow A_0$ and $\mathcal{T}(A_0) =_w^{\mathbb{M}_1} A_1'$. To continue, we need to use the property that $=_w^{\mathbb{M}_1}$ is a weak bisimulation. We get some A_1'' such that $\mathcal{T}(A_0) \Longrightarrow A_1'' =_w^{\mathbb{M}_1} A_1$. We are back to square one since $\mathcal{T}(A_0) \Longrightarrow A_1''$ could contain more than one internal step.

So Definition 25 is not as innocent as it appears. There are two ways to get over the problem with the soundness. One is to turn the maps $\mathcal{T},\mathcal{T}'$ into relations, which leads to the idea of weak subbisimilarity (Fu and Lu, 2010). A weak subbisimilarity generalizes a subbisimilarity in that the former is a weak bisimulation, not necessarily a bisimulation. The other is to strengthen the operational soundness in the manner of the next definition.

Definition 26. The translation $\mathcal{T}$ is *truly operational sound* if $\mathcal{T}(P) \Longrightarrow Q'$ implies that $P \Longrightarrow P'$ for some $P' \in \mathcal{P}_{\mathbb{M}_0}$ such that $Q' =_w^{\mathbb{M}_1} \mathcal{T}(P')$.

We can now formulate the notion of operational correspondence.

Definition 27. The translation $\mathcal{T}$ is *operationally correspondent* if it is operationally complete and truly operationally sound.

The operational correspondence is a basic requirement for any sensible translation. It must be used in combination with other criteria.

6.2.3. Weak Operational Correspondence

In the literature, see for example (Parrow, 2006), a weaker version of the operational soundness appears more popular than the operational soundness we have just discussed.

Definition 28. A translation $\mathcal{T}$ is *weakly operationally sound* if $\mathcal{T}(P) \xrightarrow{\tau} Q'$ implies $P \Longrightarrow P'$ for some $P' \in \mathcal{P}_{\mathbb{M}_0}$ and $Q' \Longrightarrow Q'' =_w^{\mathbb{M}_1} \mathcal{T}(P')$ for some $Q'' \in \mathcal{P}_{\mathbb{M}_1}$.

Using similar argument as given in Section 6.2.2, one easily sees that the transitivity of the weak operational correspondence poses even a bigger problem. One solution is to strengthen Definition 28 in the style of Definition 26.

Definition 29. A translation $\mathcal{T}$ is *truly weak operationally sound* if $\mathcal{T}(P) \Longrightarrow Q'$ implies $P \Longrightarrow P'$ for some $P' \in \mathcal{P}_{\mathbb{M}_0}$ and $Q' \Longrightarrow Q'' =_w^{\mathbb{M}_1} \mathcal{T}(P')$ for some $Q'' \in \mathcal{P}_{\mathbb{M}_1}$.

Assuming that the translation $\mathcal{T}$ is operational complete and preserves the weak equality, it is easy to show that truly weak operational soundness is transitive. A discussion of the advantage of the formulation given in the above definition has appeared in (Nestmann and Pierce, 2000).

Weak operational correspondence is often applied when operational correspondence fails. It happens when an encoding introduces extra names. The role of $Q' \Longrightarrow Q''$ is to do away with unwanted extra names by completing unfinished simulations.

6.2.4. Equivalence Criterion

When the models $\mathbb{M}_0, \mathbb{M}_1$ are close enough, it is possible to apply the *equivalence criterion.* A typical scenario of using the equivalence criterion looks like this:

1. $\mathbb{M}_0$ is syntactically a submodel of $\mathbb{M}_1$.

2. The action set of $\mathbb{M}_0$ is the same as the action set of $\mathbb{M}_1$.
3. To show that $\mathbb{M}_0$ is as expressive as $\mathbb{M}_1$, one shows that for each process P of $\mathbb{M}_1$ there is some process Q of $\mathbb{M}_0$ such that $Q =_w^{\mathbb{M}_1} P$.

The equivalence criterion is often useful when comparing the expressive powers of different variants of a model (Palamidessi, 2003). It can be seen as an application of Definition 18 in a special scenario.

6.2.5. Weak Full Abstraction

A translation fails to satisfy the full abstraction property often satisfies the so-called "weak full abstraction" property (Parrow, 2006).

Definition 30. The translation $\mathcal{T}$ is *weakly full abstract* if the following property holds: $P =_w^{\mathbb{M}_0} Q$ if and only if $\mathcal{T}(P) \approx_{\mathbb{M}_1} \mathcal{T}(Q)$.

In the above definition, $\approx_{\mathbb{M}_1}$ is the equivalence obtained by restricting the set of the observers to the set consisting of the translations of the contexts definable in $\mathbb{M}_0$.

If there is another weak fully abstract translation $\mathcal{T}'$ from some $\mathbb{M}_1$ to $\mathbb{M}_2$, a different equivalence $\approx_{\mathbb{M}_2}$ would be used. Since $\approx_{\mathbb{M}_1}$ is strictly weaker than $=_w^{\mathbb{M}_1}$, there is no way to talk about transitivity for the weak full abstraction.

At theoretical level the weak full abstraction condition must be rejected.

6.2.6. Compositionality

Almost all the translations that have been proposed are structural. If the translation of every operator is structural, then the translation of every process is structural. This motivates the following definition by Gorla (2008b,a, 2009b,a).

Definition 31. The translation $\mathcal{T}$ is *compositional* if for every k-ary operator *op* of $\mathbb{M}_0$ there exists some context $C[\_,\ldots,\_]$ of $\mathbb{M}_1$ with k-holes such that $\mathcal{T}(op(P_1,\ldots,P_k)) =_w^{\mathbb{M}_1} C[\mathcal{T}(P_1),\ldots,\mathcal{T}(P_k)]$.

Compositionality is clearly transitive. In reality a compositional translation may be stronger than what Definition 31 suggests. Take for example Milner's encoding of the lazy λ-calculus:

$$[\![MN]\!](u) \stackrel{\text{def}}{=} (v)([\![M]\!](v) \,|\, \overline{v}u.\overline{v}(n) \,|\, !n(w).[\![N]\!](w)).$$

The translation of the application is not exactly compositional according to Definition 31 since it is parameterized on the names. This example shows that the translation $\mathcal{T}$ in the above definition should be understood in general as a parameterized encoding.

Compositionality is a good property for an encoding to have. From the point of view of expressiveness, it is not clear why a non-compositional interpretation should be ruled out.

6.2.7. Name Invariance

Most translations have the nice property that they are invariant under the change of names. The criterion defined next is part of the *uniformity condition* of Palamidessi (2003). It is also heavily used in the work of Gorla (2008b,a, 2009b,a).

Definition 32. The translation $\mathcal{T}$ is *name invariant* if $\mathcal{T}(P\alpha) \equiv \mathcal{T}(P)\alpha$ for every $P \in \mathcal{P}_{\mathbb{M}_0}$ and every renaming α.

In the statement of Definition 32, the renaming α could be either injective or non-injective. Injective name invariance is a fine property, though it is not that useful technically. On the other hand non-injective renaming is a bit too strong. There is nothing wrong for a translation that maps the π-process $[a{=}b]\overline{d}e$ onto $\mathbf{0}$. But clearly every translation $\mathcal{T}$ should render $\mathcal{T}(([a{=}b]\overline{d}e)\alpha)$ and $\mathbf{0}\alpha$ unequal if α is the non-injective renaming $\{b/a\}$.

6.2.8. Correct Translation

A recent proposal for expressiveness relation makes use of the so called *correct translations* (van Glabbeek, 2012). In this approach a model is presented as a pair $(\mathbb{M}, \mathcal{I}_{\mathbb{M}})$ where $\mathbb{M}$ introduces valid expressions and $\mathcal{I}_{\mathbb{M}}$ is a surjective interpretation map from $\mathbb{M}$ to some domain $D_{\mathbb{M}}$ of meanings. A corret translation from $(\mathbb{M}, \mathcal{I}_{\mathbb{M}})$ to $(\mathbb{N}, \mathcal{I}_{\mathbb{N}})$ is a map $\mathcal{T}$ from $\mathbb{M}$ to $\mathbb{N}$ such that $\mathcal{I}_{\mathbb{M}}(T) \asymp \mathcal{I}_{\mathbb{N}}(\mathcal{T}(T))$ for all T, where $\asymp$ is some appropriate semantic equivalence on $D_{\mathbb{M}} \cup D_{\mathbb{N}}$. According to the definition a correct translation from $(\mathbb{M}, \mathcal{I}_{\mathbb{M}})$ to $(\mathbb{N}, \mathcal{I}_{\mathbb{N}})$ depends on $\mathcal{I}_{\mathbb{M}}$, $\mathcal{I}_{\mathbb{N}}$ and $\asymp$. For simplicity suppose that $\mathcal{I}_{\mathbb{M}}$ and $\mathcal{I}_{\mathbb{N}}$ are given by the labeled transition semantics of the respective models. So $\mathcal{I}_{\mathbb{M}}(T)$ for example is the labeled transition system or the process graph induced by T. Now if $(\mathbb{M}, \mathcal{I}_{\mathbb{M}})$ and $(\mathbb{N}, \mathcal{I}_{\mathbb{N}})$ have quite different action labels, defining the semantic equivalence relation $\asymp$ is just as hard as defining the translation $\mathcal{T}$. One could even argue that defining $\asymp$ is the same thing as defining $\mathcal{T}$. In general the story is not that one has a semantic relation $\asymp$ in advance, and then one defines the translation $\mathcal{T}$. The relation $\asymp$ and the map $\mathcal{T}$ are defined simultaneously since both depend on the semantic mappings $\mathcal{I}_{\mathbb{M}}$ and $\mathcal{I}_{\mathbb{N}}$. Simpler situations occur when $D_{\mathbb{M}} = D_{\mathbb{N}}$.

6.3. On Completeness

There has been no consensus on what it means for a process calculus to be Turing complete. For the restricted question of Turing completeness of the π-calculus, Milner's answer is the most influential one (Milner, 1992). His encoding, a typical application of the apparatus of the π-calculus, is given below:

$$\begin{aligned}
[\![x]\!]_a &\stackrel{\text{def}}{=} \overline{x}a, \\
[\![\lambda x.M]\!]_a &\stackrel{\text{def}}{=} a(x).a(v).[\![\lambda x.M]\!]_v, \\
[\![MN]\!]_a &\stackrel{\text{def}}{=} (c)([\![M]\!]_c \,|\, !\overline{c}(f).\overline{c}a.f(w).[\![N]\!]_w).
\end{aligned}$$

It is proved in (Milner, 1992) that the encodings of the closed λ-terms both preserve and reflect the operational semantics of the lazy λ-calculus of Abramsky (1988). In the followup papers (Sangiorgi, 1994, 1995) Sangiorgi proved that Milner's encoding is fully abstract with respect to the open applicative bisimilarity on the λ-terms. In (Milner, 1992) Milner has also given an interpretation of the call-by-value λ-calculus (Plotkin, 1975) in the π-calculus. Encodings into the variants of the π-calculus have been investigated in (Sangiorgi, 1993; Fu, 1999; Merro and Sangiorgi, 2004). None of these encodings are good for the full λ-calculus (Barendregt, 1984). The fully abstract encoding of the full λ-calculus in π (Cai and Fu, 2011) makes use of the match/mismatch operator and the guarded choice operator.

Over the last few years the Turing completeness property has been studied for a number of process calculi (Palamidessi, 2003; Busi et al., 2004; Maffeis and Phillips, 2005; Fu and Lu, 2010) using the machine model rather than the λ-calculus. Maffeis and Phillips (2005) have summarized several criteria for Turing completeness. The weakest criterion that has appeared in literature can be stated as follows:

> TC-0: Let C be a computation model. A process calculus $\mathcal{M}$ with a canonical equivalence $\asymp$ on the $\mathcal{M}$-processes is Turing complete if there is a function $\mathfrak{e} : \mathsf{C} \to \mathcal{P}_{\mathcal{M}}$ such that the following statements are valid for each $\mathsf{c} \in \mathsf{C}$:

(i) If $\mathsf{c} \rightarrow \mathsf{c}'$ then $\mathfrak{e}(\mathsf{c}) \Longrightarrow P$ for some process P such that $P \asymp \mathfrak{e}(\mathsf{c}')$.
(ii) If $\mathfrak{e}(\mathsf{c}) \xrightarrow{\tau} P$ then $\mathsf{c} \rightarrow^* \mathsf{c}'$ for some c' such that $\mathfrak{e}(\mathsf{c}') \asymp P$.
(iii) c diverges if and only if $\mathfrak{e}(\mathsf{c})$ diverges.

TC-0 is useful for establishing undecidable results. It has been used as a statement that asserts Turing completeness. In our opinion TC-0 is too weak to be a criterion for anything. It is satisfied by some trivial encodings. Let $\mathfrak{b}$ for example be a function defined by the following.

$$\mathfrak{b}(\mathsf{c}) \stackrel{\mathrm{def}}{=} \begin{cases} \mathbf{0}, & \text{if } \mathsf{c} \text{ terminates,} \\ \Omega, & \text{if } \mathsf{c} \text{ diverges.} \end{cases}$$

The function $\mathfrak{b}$ clearly satisfies TC-0. It basically says that all process calculi are Turing complete according to TC-0. The encodings defined in process calculus literature actually satisfy far more properties, say the effectiveness, than TC-0. What Turing completeness is really about is the simulations of functional behaviours. This can be formally stated as follows.

TC-1: Let R be the model of recursive funcitons. A process calculus $\mathcal{M}$ with a canonical equivalence $\asymp$ on the $\mathcal{M}$-processes is Turing complete if there is a function $\mathfrak{e} : \mathsf{R} \rightarrow \mathcal{P}_{\mathcal{M}}$ such that the following statements are valid for all $\mathsf{f}(x_1, \ldots, x_i) \in \mathsf{R}$ and all numerals $\underline{n_1}, \ldots, \underline{n_i}$:
(i) If $\mathsf{f}(\underline{n_1}, \ldots, \underline{n_i}) \rightarrow \mathsf{f}'$ then $\mathfrak{e}(\mathsf{f}(\widetilde{x})) \,|\, \mathfrak{e}(\underline{n_1}) \,|\, \ldots \,|\, \mathfrak{e}(\underline{n_i}) \Longrightarrow P$ for some process P such that $P \asymp \mathfrak{e}(\mathsf{f}')$.
(ii) If $\mathfrak{e}(\mathsf{f}(\widetilde{x})) \,|\, \mathfrak{e}(\underline{n_1}) \,|\, \ldots \,|\, \mathfrak{e}(\underline{n_i}) \xrightarrow{\tau} P$ then $\mathsf{f}(\underline{n_1}, \ldots, \underline{n_i}) \rightarrow^* \mathsf{f}'$ for some f' such that $\mathfrak{e}(\mathsf{f}') \asymp P$.
(iii) $\mathsf{f}(\underline{n_1}, \ldots, \underline{n_i})$ diverges if and only if $\mathfrak{e}(\mathsf{f}(\widetilde{x})) \,|\, \mathfrak{e}(\underline{n_1}) \,|\, \ldots \,|\, \mathfrak{e}(\underline{n_i})$ diverges.

It is not difficult to see that an encoding satisfying TC-1 must not confuse two distinct numerals. The consequence is that the recursive functions cannot be interpreted in a trivial fashion. TC-1 appears to be the strongest criterion one could find in the literature. Other weaker forms are more liberal about (ii) and/or (iii). For example the encoding of a recursive function might get stuck or diverge on some inputs even if the recursive function is defined on these inputs. TC-1 and its weaker forms have been used to establish Turing completeness, or the absence of it, of CCS (Busi et al., 2004; Fu and Lu,

2010) and Ambient Calculi (Hirschkoff et al., 2002; Zimmer, 2003; Busi and Zavattaro, 2004; Maffeis and Phillips, 2005).

From the point of view of interaction, even TC-1 is subject to criticism. Suppose $\mathsf{f}(\underline{n_1},\ldots,\underline{n_i}) \rightarrow^* \underline{n}$ and

$$\mathfrak{e}(\mathsf{f}(\widetilde{x})) \,|\, \mathfrak{e}(\underline{n_1}) \,|\, \ldots \,|\, \mathfrak{e}(\underline{n_i}) \xrightarrow{\tau} P_1 \xrightarrow{\tau} P_2 \xrightarrow{\tau} \ldots \xrightarrow{\tau} P_k \asymp \mathfrak{e}(\underline{n}). \qquad (20)$$

According to definition $\mathsf{f}(\underline{n_1},\ldots,\underline{n_i})$ is computationally equal to $\underline{n}$. This equality is generally not respected by the encoding $\mathfrak{e}$ since it is possible that $P_k \not\asymp P_i$ for all $i < k$. The consequence is that the simulation of the encoding of a recursive function can be easily interrupted by the environment and an environment is never sure of what it is getting.

There are other variant definitions of the Turing completeness. It suffices to say that none of them meet our requirement on completeness. The encodings of the lazy λ-calculus in the π-calculus for example lack of the mechanism of result delivery. Other encodings appear weaker in that they have neither the input nor the output mechanisms. Although computations can be carried out by these encodings, the results of the computations are not usable to any processes. It is in the light of these facts that the completeness of this paper should be judged.

The notion of completeness is so important to Theory of Interaction that a sound formulation of the notion is of paramount importance.

6.4. Natural Criteria

The comparisons carried out in Section 6.1 and Section 6.2 are done with regards to *interpretation* rather than *translation*. For theoretical studies interpretation should definitely be preferred. There are at least two reasons.

1. Results about interpretability are model independent. They have both theoretical significance and practical implication.
2. A solid negative result about expressiveness must be proved by showing the absence of any interpretation. The nonexistence of any translations or encodings satisfying a set of conditions cannot be conclusive.

In practice one is more interested in the effective proofs of positive results. This is the case when one tries to construct a compiler for a programming language using another programming language. In reality there is little point in claiming that $\mathbb{L}'$ is more expressive than $\mathbb{L}$ if one cannot implement the former by the latter. The motivations for the structural criteria proposed in literature, notably those by Nestmann (2000), Palamidessi (2003) and Gorla (2008a), are the belief that if there are translations between two *natural models*, there are *natural translations* between the models. This belief has been widely exploited in computation theory. The criteria discussed in Section 6.2 are natural criteria. Different authors have different understanding of the non-technical word "natural" in different application scenarios, hence the diversity. If one compares for instance two models that fit in the same programming paradigm, one applies compositionality and name invariance conditions. If one is looking for an efficient algorithm for a specific problem, one may impose some structural constraints on the solutions.

In practice one also uses natural equalities for different purposes. One such equality is the testing equivalence De Nicola and Hennessy (1984). The model independent approach can be applied to the testing equivalence, leading to a different theory than what is presented in this paper.

It is from the angle of theoretical necessity and application diversity that the Theory of Interaction and the work of others can be fully appreciated.

7. Conclusion

The most significant contribution of this paper is the model independent formulation of the two most important relationships in Computer Science, the expressiveness relationship between the models and the equality relationship between the objects of a model. The mere fact that the Axiom of Completeness can be formally stated is itself a manifestation. The uniform definition of the equality and the expressiveness is a technical advance, making it clear how the expressiveness relationship can be unique and why the full abstraction is a necessary requirement. Theory of Interaction is open to logic study. In logic, equality is unique, and there is no ambiguity when one talks about one logic being more expressive than another. We have seen the benefit of using logic in the formulation of the Axiom of Completeness. Another example, the Axiom of Computation, is introduced in (Fu, 2015).

Theory of Interaction provides a number of guidelines when introducing a new model $\mathbb{N}$. The following two questions must be addressed:

- Is $\mathbb{N}$ complete?
- Is the absolute equality a congruence?

More information about $\mathbb{N}$ can be obtained by answering harder questions. Theory of Interaction allows to formalize these questions unambiguously.

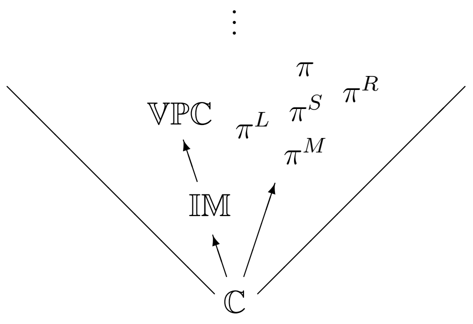

Figure 2: The World of Models

What is carried out in this paper is the starting point for many things to follow. In sequel we point out some problems, unsettled in this paper, and several interesting research directions.

7.1. Unsettled Problem

The picture in Fig. 2 is what we currently know, leaving aside for the moment the question of what π^L, π^S, π^R stand for. In Section 4 we have derived some structural results. More generally one may ask the following.

Problem 1. Is there any useful structural theory about $\mathfrak{Mod}$?

In particular, how do we stratify $\mathfrak{Mod}$? Is there any interesting hierarchy theorem or theory for $\mathfrak{Mod}$? Should there be a final model? To answer the question one would probably introduce something like processes with oracles. It remains to see how to define the notion of oracles for interaction models.

For the models without the choice operator, the problem one comes across when attempting to prove the coincidence is to do with the bisimulations of the input actions. Take the π-calculus for instance. Suppose $P =_\pi Q \xrightarrow{ac} Q'$ and d is a fresh name. Now $Q \,|\, \overline{a}c.d \,|\, \overline{d} \xrightarrow{\tau}\xrightarrow{\tau} Q'$ must be simulated by some $P \,|\, \overline{a}c.d \,|\, \overline{d} \overset{\tau}{\Longrightarrow} P'$. From the latter internal action sequence it is easy to derive $P \overset{ac}{\Longrightarrow} P'$. But so far we have not been able to show that the simulation must take the form $P \rightarrow^* \xrightarrow{ac} P'$. This issue is technically important and practically useful. We would very much like to see any progress on it.

Problem 2. What is the observation theory of π^M?

If the external bisimilarity turns out to be the same as given in Definition 14, one may immediately conclude that $\pi^M \sqsubset \pi$. The external characterization of $=_{\pi^M}$ is important if one studies models that have name-passing features.

Definition 33. A model $\mathbb{M}$ *has name-passing* if $\pi^M \sqsubseteq \mathbb{M}$. A model $\mathbb{M}$ *has value-passing* if $\mathbb{VPC}^- \sqsubseteq \mathbb{M}$.

A external characterization of the absolute equality of $\mathbb{VPC}^-$ is also open. Section 4.4 tells us that there are possibly many incompatible interpretations of a complete model into itself. It also suggests that incompatible interpretations are often caused by different encoding-decoding functions. At the moment $\mathbb{VPC}$ and $\mathbb{IM}$ are the only models we know that have nontrivial self interpretations. The picture of the self interpretations on $\mathbb{VPC}/\mathbb{IM}$ will not be fully understood until we answer the following.

Problem 3. Is every self interpretation on $\mathbb{VPC}/\mathbb{IM}$ induced by a bijective computable function?

7.2. Future Direction

Future studies in Theory of Interaction may stretch in several directions.

1. The first direction is to expand the current picture, given by the diagram in Fig. 2, of the world of interaction models. The only model missing in the picture is the asynchronous π-calculus. We have not discussed the model due to space consideration. The model π^L, called local π by Merro and Sangiorgi (2004), is the π-variant in which a name received from a communication cannot be used as an input channel. Symmetrically π^R (Fu and Zhu, 2015) is the π-variant in which a name received from a communication cannot be used as an output channel. Another π-variant, called π^S in (Fu and Zhu, 2015), is defined by imposing the restriction that the received names can only be used as channel names (subject names). The expressiveness of π^L, π^S, π^R is studied in (Xue et al., 2011). A sensible thing to do is to examine the other major process models not yet discussed in this paper. One could compare along the way the existing expressiveness results against the results obtained by applying the subbisimilarity relationship. There is little doubt that exercises of this sort would bring to us new insight into these process models.
2. The second direction is to apply the model independent philosophy of Theory of Interaction to other frameworks. If we go about the plan in the order of the equality theory, the expressiveness theory and the completeness theory, we should start from the picture described in Fig. 3. All equalities for models of interaction should be observational. In

$$\begin{tikzcd}
= & \text{strongest intensional requirement} \\
=_w \arrow[u] & \\
=_t \arrow[u] & \\
=_e \arrow[u] & \text{no intensional requirement}
\end{tikzcd}$$

Figure 3: The World of Equalities

other words they must satisfy the extensionality and equipollence conditions. So every equality is above the extensional equality introduced by Definition 23. On the other hand, the bisimulation and codivergence conditions are the strongest intensional (computational) requirements one may impose on the observational equalities. Consequently every equality is under the absolute equality. Now suppose $=_t$ is a model independent equality that lies between the extensional equality and the absolute equality. The definition of $=_t$ either introduces some new intensional condition, or weakens the codivergence and bisimulation conditions, or is a mixture of both. Due to its model independent nature, the definition can be routinely generalized to the expressiveness relationship. The rest of the job is to develop the expressiveness theory and the completeness theory that go along with $=_t$. The views and the results obtained along this line of research would certainly improve our understanding of the interaction models.
The model independent approach can also be applied to the study of the pre-orders on processes. A progress along this line of investigation is reported in (He, 2010).
We can try to use the model independent approach to study other communication mechanisms. Some process calculi are based on communication media that support one-many broadcast. Some other process calculi permit a group of processes to synchronize at a particular communication media, like CSP (Hoare, 1985). Still some other process calculi can engage in a one-one interaction based on locations, like Ambient Calculus of Cardelli and Gordon (2000). It remains to see how the basic idea of Theory of Interaction can be applied to these

different communication mechanisms.

3. The third direction is to rework the concurrency theory in the light of Theory of Interaction. There are many aspects one could inspect. One may try to formulate the asynchronous theory (Honda and Tokoro, 1991a,b; Boudol, 1992; Amadio et al., 1998) in Theory of Interaction. Initial progress on this topic is reported in (Fu, 2010). One may apply the principles of Theory of Interaction to the Petri Net Model (Reisig, 1985) so that the problem of compositionality is solved in a satisfactory way. Theory of Interaction provides a method, as well as a toolkit, to shrink the bulk of the concurrency theory by streamlining the models.
4. The fourth direction is to integrate the theory of computation (Dasgupta et al., 2006; Papadimitriou, 1994; Wegener, 2005) into Theory of Interaction. Since every interaction model is complete, one may carry out algorithm design and complexity analysis in an interaction model. The additional help Theory of Interaction might provide is a formal treatment of the distinctions between the determinism and the nondeterminism. The hope is that by expanding the computation theory into the dimension of interaction, nondeterminism can be dealt with at the model level rather than at the algorithmic level.
5. The fifth direction is to explore the tools developed in Theory of Interaction to search for significant negative results. The theory of process calculus is a theory abundant in models. It is also a theory that is by and large flat. The lack of deep negative results supports the flat theory view. Finkel and Schnoebelen explain in (Finkel and Schnoebelen, 2001) how the well quasi order structure (Kruskal, 1972) can be used to derive a number of results in concurrency theory. The same structures are used in obtaining some negative results in process theory (Busi et al., 2003, 2004; Fu and Lu, 2010). Another interesting negative result is Sewell's theorem on the nonexistence of a finite axiomatic system of the finite state processes (Sewell, 1994, 1995, 1997). There are not many such examples. The meta theoretical framework offered by Theory of Interaction ought to be helpful in obtaining general negative results.
6. The sixth direction is to apply the ideas of Theory of Interaction to solving practical issues. With the advance in hardware, interest in concurrent programming has been revived. Studies in this area face many new challenges. In concurrent program analysis for example, one needs a language independent notion of program equivalence to start with.

> To evaluate the virtues of different concurrent programming languages, one cannot avoid using some language independent expressiveness relation. There are other application areas where Theory of Interaction should help in one way or another. Feedbacks from applications will be important to further development of the theory.

So Theory of Interaction not only offers a framework, it also offers problems.

There are a lot to do.

Acknowledgment

I would like to thank the members of BASICS for their interest, discussions and feedbacks. I am grateful to Huan Long and Jianxin Xue who have discovered a number of serious mistakes in several drafts of the paper. In particular they demonstrated to me that my original perception about the self interpretations on the π-calculus was terribly wrong, and that the tricks used in the proof of Theorem 4.12 can be reused to tell apart the polyadic π from the monadic π. I owe a great deal to Xiaojuan Cai, Xiaoju Dong, Huan Long, Hao Lu, Jianxin Xue and Han Zhu for having collaborated on issues related to the topics of this paper. I have also benefited a lot from an informative discussion with professor Mingsheng Ying about axiomatic approach in computer science.

Working on the topics covered in this paper has been a fascinating experience. Towards the end of 2006, after my collaboration with Hao Lu has led to the results described in (Fu and Lu, 2010), I started thinking about a general model independent theory of interaction. I got heavily occupied by it in the years 2007-2008. During the time several frameworks had been investigated and denied. By early 2009 the theory was beginning to take its present shape. I made an announcement of a draft paper in BASICS 2009. The questions and comments I received from the audience of the workshop have kept me working on the subject ever since.

I must thank the two anonymous referees for their detailed comments, suggestions, questions and criticisms. In many ways their insightful advice has helped to improve the quality of the paper in an indispensable fashion.

Over the years I have been supported by NSFC projects one after another. I would like to mention in particular the project number 61033002 and the PACE project number ANR 61261130589. Seriously I owe NSFC a big apology for not being able to deliver in time enough published papers. In retrospect I should have broadened my research scope many years ago.

Appendix A. Input via Output?

The next two facts describe some standard properties of $[\![\_]\!]^{\bowtie}$.

Fact 1. *The following statements are valid.*

1. *If* $T \xrightarrow{\tau} T'$ *then* $[\![T]\!]^{\bowtie} \xrightarrow{\tau}\xrightarrow{\tau} [\![T']\!]^{\bowtie}$.
2. $[\![\_]\!]^{\bowtie}$ *is contained in a minimal extension* $[\![\_]\!]^{\bowtie}_{\Downarrow}$ *that is equipollent, extensional, codivergent and bisimilar.*

Proof. The validity of (1) is obvious. In fact $T \xrightarrow{\tau} T'$ implies $[\![T]\!]^{\bowtie} \xrightarrow{\tau} T_c \xrightarrow{\tau} [\![T']\!]^{\bowtie}$ for some T_c such that $T_c = [\![T']\!]^{\bowtie}$. The equality tells us how to extend the function $[\![\_]\!]^{\bowtie}$ to a minimal relation $[\![\_]\!]^{\bowtie}_{\Downarrow}$ that is equipollent, extensional, codivergent and bisimilar. $\square$

Fact 2. *The composition* $=;\{([\![P]\!]^{\bowtie},[\![Q]\!]^{\bowtie}) \mid P = Q\};=$ *is equipollent, extensional, codivergent and bisimilar.*

Proof. Let $\mathcal{R}$ stand for the relation $\{([\![P]\!]^{\bowtie},[\![Q]\!]^{\bowtie}) \mid P = Q\}$. Now suppose

$$A' \xleftarrow{\iota} A = [\![P]\!]^{\bowtie}\ \mathcal{R}\ [\![Q]\!]^{\bowtie} = B$$

and that $A \xrightarrow{\iota} A'$ is simulated by $[\![P]\!]^{\bowtie} \rightarrow^* P''_c \xrightarrow{\iota} P'_c$ for some P''_c, P'_c such that $A = P''_c$ and $A' = P'_c$. We may as well assume that $P''_c \equiv [\![P'']\!]^{\bowtie}$ for some P'' such that $P \Longrightarrow P''$. Now $P'' \xrightarrow{\tau} P'$ for some P' such that $P'_c = [\![P']\!]^{\bowtie}$. Using the fact $P = Q$, one gets some Q'', Q' such that $Q \Longrightarrow Q'' \xrightarrow{\tau} Q' = P'$ and $Q'' = P''$. Therefore $[\![Q]\!]^{\bowtie} \Longrightarrow [\![Q'']\!]^{\bowtie} \xrightarrow{\tau}= [\![Q']\!]^{\bowtie}\ \mathcal{R}\ [\![P']\!]^{\bowtie}$ and $[\![Q'']\!]^{\bowtie}\ \mathcal{R}\ [\![P'']\!]^{\bowtie}$. Finally some B'', B' exist such that $B \Longrightarrow B'' \xrightarrow{\tau} B' = [\![Q']\!]^{\bowtie}$ and $B'' = [\![Q'']\!]^{\bowtie}$. What we have essentially demonstrated is that $=;\mathcal{R};=$ is equipollent, extensional, codivergent and bisimilar. $\square$

What is missing from Fact 1 and Fact 2? Had we established reflexivity of $=;\mathcal{R};=$, we would have proved that the composition $[\![\_]\!]^{\bowtie};=$ were a subbisimilarity and that $[\![\_]\!]^{\bowtie}$ were a correct encoding. This is impossible. The map $[\![\_]\!]^{\bowtie}$ fails the soundness condition. The process $!a(x).\overline{b}x$ is equal to the process $(d)(a(x).(\overline{b}x \mid \overline{d}) \mid !d.a(x).(\overline{b}x \mid \overline{d}))$. But the interpretation $[\![!a(x).\overline{b}x]\!]^{\bowtie}$ is obviously unequal to

$$[\![(d)(a(x).(\overline{b}x \mid \overline{d}) \mid !d.a(x).(\overline{b}x \mid \overline{d}))]\!]^{\bowtie}$$

since the former can always do input actions at a whereas the latter can only do one such action.

Appendix B. Value-Passing via Name-Passing?

The encodings are based on an injective map of the term variables onto the name variables. For simplicity we pretend that this map is an identity function. In what follows we will make use of the following encoding of the polyadic prefixes in terms of the monadic prefixes (Milner, 1993b):

$$\begin{aligned} a(x_1,\ldots,x_k).T &\overset{\mathrm{def}}{=} a(w).w(x_1).\cdots.w(x_k).T, \\ \overline{a}\langle n_1,\ldots,n_k\rangle.T &\overset{\mathrm{def}}{=} \overline{a}(c).\overline{c}n_1.\cdots.\overline{c}n_k.T, \\ \overline{a}(b_1,\ldots,b_k).T &\overset{\mathrm{def}}{=} \overline{a}(c).\overline{c}(b_1).\cdots.\overline{c}(b_k).T. \end{aligned}$$

The encoding does not give rise to a subbisimilarity. But it is good for our purpose.

The interpretation of every object in the π-calculus must be accessible. In other words it must let environments know its existence. The numeral $\underline{n}$ will be interpreted as a process that can be visited at a name. So there is an infinite number of interpretations of every numeral, each being accessible at a particular name. The numerals can be coded up in many ways. We shall use the following encoding:

$$\begin{aligned} [\![\underline{0}]\!]^{\pi}_{c} &\overset{\mathrm{def}}{=} \overline{c}(e,f).\overline{f}, \\ [\![\underline{n{+}1}]\!]^{\pi}_{c} &\overset{\mathrm{def}}{=} (d)(\overline{c}(e,f).\overline{e}d \mid [\![\underline{n}]\!]^{\pi}_{d}). \end{aligned}$$

It is obvious from the definition that $[\![\underline{i}]\!]^{\pi}_{c} = [\![\underline{j}]\!]^{\pi}_{c}$ if and only if $i = j$. A process interacts with $[\![\underline{n}]\!]^{\pi}_{c}$ would have to consume $[\![\underline{n}]\!]^{\pi}_{c}$ as it were in order to figure out what the numeral is. Once the process has got the numeral, it would probably make use of the numeral several times. For that to be possible, the process must make several copies of the numeral. A better approach is to introduce an operation that duplicates the encoded numeral for a potentially infinite number of times. To describe the operation it is convenient to introduce the following persistent form of the encoding:

$$\begin{aligned} [\![!\underline{0}]\!]^{\pi}_{c} &\overset{\mathrm{def}}{=} !\overline{c}(e,f).\overline{f}, \\ [\![!\underline{n{+}1}]\!]^{\pi}_{c} &\overset{\mathrm{def}}{=} (d)(!\overline{c}(e,f).\overline{e}d \mid [\![!\underline{n}]\!]^{\pi}_{d}). \end{aligned}$$

Now for every π-term T, we would like to introduce a π-term $Rep(x,u).T$ that replicates an input numeral. Operationally it satisfies the following computational property:

$$[\![\underline{n}]\!]^{\pi}_{c} \mid Rep(c,d).T \xrightarrow{\tau}= [\![!\underline{n}]\!]^{\pi}_{d} \mid T.$$

We may define $Rep(x,u).T$ by the following term

$$\begin{aligned}&(fg)(x(y,z).(z.(T\,|\,!\overline{u}(e,f).\overline{f})\,|\,y(x).(d)\overline{f}\langle d,x\rangle.g.!\overline{u}(e,f).\overline{e}d)\\&\qquad|\,!f(v,x).x(y,z).(z.\overline{g}.(T\,|\,!\overline{v}(e,f).\overline{f})\,|\,y(x).(d)\overline{f}\langle d,x\rangle.!\overline{v}(e,f).\overline{e}d)).\end{aligned}$$

The term $Rep(c,d).T$ transforms the encoding $[\![\underline{n}]\!]^{\pi}_{c}$ into a replicated form before T can be fired. The name g is used to prevent the replicated form from being used before it is completely generated. Once we have the process $[\![!\underline{n}]\!]^{\pi}_{c}$ we might want to make a copy of it whenever necessary. This is done by $Copy(x,u).T$ that is defined by the following term

$$\begin{aligned}&(fg)(x(y,z).(z.(T\,|\,\overline{u}(e,f).\overline{f})\,|\,y(x).(d)\overline{f}\langle d,x\rangle.g.\overline{u}(e,f).\overline{e}d)\\&\qquad|\,!f(v,x).x(y,z).(z.\overline{g}.(T\,|\,\overline{v}(e,f).\overline{f})\,|\,y(x).(d)\overline{f}\langle d,x\rangle.\overline{v}(e,f).\overline{e}d)).\end{aligned}$$

It is clear that the following computations are admissible.

$$\begin{aligned}[\![\underline{n}]\!]^{\pi}_{c}\,|\,Copy(c,d).T &\xrightarrow{\tau}= [\![\underline{n}]\!]^{\pi}_{d}\,|\,T,\\ [\![!\underline{n}]\!]^{\pi}_{c}\,|\,Copy(c,d).T &\xrightarrow{\tau}= [\![!\underline{n}]\!]^{\pi}_{c}\,|\,[\![\underline{n}]\!]^{\pi}_{d}\,|\,T.\end{aligned}$$

The encoding of the boolean expressions explores the fact that when a process reaches to a state where a conditional subterm is in a fireable position, its free variables must have all been instantiated. The structural definition of the encoding is as follows:

$$\begin{aligned}[\![\top]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} \overline{c}(e,f).\overline{e},\\
[\![\bot]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} \overline{c}(e,f).\overline{f},\\
[\![p{=}q]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} Equal(p,q,c),\\
[\![p{<}q]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} Less(p,q,c),\\
[\![\neg\varphi]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} (d)([\![\varphi]\!]^{\pi}_{d}\,|\,d(u,v).(u.[\![\bot]\!]^{\pi}_{c}\,|\,v.[\![\top]\!]^{\pi}_{c})),\\
[\![\varphi\wedge\psi]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} (d_1d_2)([\![\varphi]\!]^{\pi}_{d_1}\,|\,[\![\psi]\!]^{\pi}_{d_2}\,|\,d_1(u,v).d_2(u',v').(u.u'.[\![\top]\!]^{\pi}_{c}\\
&\qquad |\,u.v'.[\![\bot]\!]^{\pi}_{c}\,|\,v.u'.[\![\bot]\!]^{\pi}_{c}\,|\,v.v'.[\![\bot]\!]^{\pi}_{c})),\\
[\![\varphi\vee\psi]\!]^{\pi}_{c} &\stackrel{\text{def}}{=} (d_1d_2)([\![\varphi]\!]^{\pi}_{d_1}\,|\,[\![\psi]\!]^{\pi}_{d_2}\,|\,d_1(u,v).d_2(u',v').(u.u'.[\![\top]\!]^{\pi}_{c}\\
&\qquad |\,u.v'.[\![\top]\!]^{\pi}_{c}\,|\,v.u'.[\![\top]\!]^{\pi}_{c}\,|\,v.v'.[\![\bot]\!]^{\pi}_{c})).\end{aligned}$$

The processes $Equal(p,q,c)$ and $Less(p,q,c)$ appeared in the above encoding are defined as follows:

$$\begin{array}{rcl} Equal(p,q,c) & \stackrel{\text{def}}{=} & (d)(\overline{d}\langle p,q\rangle \,|\, !d(u,v).u(x,y).v(x',y').(x(w).x'(w').\overline{d}\langle w,w'\rangle \\ & & \quad |\, y.x'(w').[\![\bot]\!]^\pi_c \,|\, x(w).y'.[\![\bot]\!]^\pi_c \,|\, y.y'.[\![\top]\!]^\pi_c)), \\ Less(p,q,c) & \stackrel{\text{def}}{=} & (d)(\overline{d}\langle p,q\rangle \,|\, !d(u,v).u(x,y).v(x',y').(x(w).x'(w').\overline{d}\langle w,w'\rangle \\ & & \quad |\, y.x'(w').[\![\top]\!]^\pi_c \,|\, x(w).y'.[\![\bot]\!]^\pi_c \,|\, y.y'.[\![\bot]\!]^\pi_c)). \end{array}$$

The correctness of the encoding $[\![\varphi]\!]^\pi_c$ is stated in the next lemma.

Fact 3. *Suppose $x_1,\ldots,x_i$ are free in φ. Then $\vdash \varphi\{\underline{n_1}/x_1,\ldots,\underline{n_i}/x_i\}$ if and only if $(c_1\ldots c_i)([\![\varphi]\!]^\pi_c\{c_1/x_1,\ldots,c_i/x_i\} \,|\, [\![\underline{n_1}]\!]^\pi_{c_1} \,|\, \ldots \,|\, [\![\underline{n_i}]\!]^\pi_{c_i}) = [\![\top]\!]^\pi_c$.*

The encoding of the term expressions is straightforward, using the fact that a term expression is either a numeral, or a term variable, or of the form $\underline{i+x}$.

$$[\![t]\!]^\pi_c \stackrel{\text{def}}{=} \begin{cases} [\![\underline{i}]\!]^\pi_c, & \text{if } t = \underline{i}, \\ Copy(x,c), & \text{if } t = x, \\ (c_1\ldots c_i)Copy(x,c_i).(\overline{c}c_1 \,|\, \overline{c_1}c_2 \,|\, \ldots \,|\, \overline{c_{i-1}}c_i), & \text{if } t = \underline{i+x}. \end{cases}$$

Finally we can define the encoding of the $\mathbb{VPC}$-processes. The translation is defined by the following induction:

$$\begin{array}{rcl} [\![\sum_{1\le i\le k} a(x).T_i]\!]^{vpc\to\pi} & \stackrel{\text{def}}{=} & \sum_{1\le i\le k} a(x).(c)Rep(x,c).[\![T_i]\!]^{vpc\to\pi}\{c/x\}, \\ [\![\sum_{1\le i\le k} \overline{a}(t_i).T_i]\!]^{vpc\to\pi} & \stackrel{\text{def}}{=} & \sum_{1\le i\le k} \overline{a}(c).([\![t_i]\!]^\pi_c \,|\, [\![T_i]\!]^{vpc\to\pi}), \\ [\![\mathit{if}\,\varphi\ \mathit{then}\ T]\!]^{vpc\to\pi} & \stackrel{\text{def}}{=} & (c)([\![\varphi]\!]^\pi_c \,|\, c(u,v).u.[\![T]\!]^{vpc\to\pi}), \\ [\![!a(x).T]\!]^{vpc\to\pi} & \stackrel{\text{def}}{=} & !a(x).(c)Rep(x,c).[\![T]\!]^{vpc\to\pi}\{c/x\}, \\ [\![!\overline{a}(t).T]\!]^{vpc\to\pi} & \stackrel{\text{def}}{=} & !\overline{a}(c).([\![t]\!]^\pi_c \,|\, [\![T]\!]^{vpc\to\pi}). \end{array}$$

Compared to the map $[\![\_]\!]^{\bowtie}$, the translation $[\![\_]\!]^{vpc\to\pi}$ is much better. We will not go into details to demonstrate how this encoding satisfies the good properties stated in the next lemma. It suffices to say that the long proof is routine.

Fact 4. *The composition $[\![\_]\!]^{vpc\to\pi};=_\pi$ is equipollent, extensional, codivergent and bisimilar.*

Fact 4 should not be overvalued since $[\![\_]\!]^{vpc\to\pi};=_\pi$ fails to be sound.

References

Abramsky, S., 1988. The Lazy Lambda Calculus. In: Turner, D. (Ed.), Declarative Programming. Addison-Wesley, 65–116.

Abramsky, S., 2006. What are the Fundamental Structures of Concurrency? We still do not know. Electronic Notes in Theoretical Computer Science 162, 37–41.

Aceto, L., Fokkink, W., Verhoef, C., 2001. Structural operational semantics. In: Bergstra, J., Ponse, A., Smolka, S. (Eds.), Handbook of Process Algebra. North-Holland, 197–292.

Aceto, L., Hennessy, M., 1992. Termination, deadlock, and divergence. Journal of ACM 39, 147–187.

Amadio, R., 1993. On the reduction of chocs bisimulation to π-calculus bisimulation. In: CONCUR'93. Lecture Notes in Computer Science 715, 112–126.

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

Amadio, R., Curien, P., 1998. Domains and Lambda-Calculi. Cambridge Tracts in Theoretical Computer Science. CUP.

Arora, S., Barak, B., 2009. Computational Complexity, a modern approach. CUP.

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

Baeten, J., van Glabbeek, R., 1987. Another look at abstraction in process algebra. In: ICALP'87. Lecture Notes in Computer Science 267, 84–97.

Baeten, J., Weijland, W., 1990. Process Algebra. Vol. 18 of Cambridge Tracts in Theoretical Computer Science. CUP.

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

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

Berry, G., Boudol, G., 1992. The chemical abstract machine. Theoretical Computer Science 96, 217–248.

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

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

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

Boreale, M., De Nicola, R., 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.

Bougé, L., 1988. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. Acta Informatica 25, 179–201.

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

Burkart, O., Caucal, D., Moller, F., Steffen, B., 2001. Verification on infinite structures. In: Bergstra, J., Ponse, A., Smolka, S. (Eds.), Handbook of Process Algebra. North-Holland, 545–623.

Busi, N., Gabbrielli, M., Zavattaro, G., 2003. Replication vs recursive definitions in channel based calculi. In: ICALP'03. Lecture Notes in Computer Science 2719, 133–144.

Busi, N., Gabbrielli, M., Zavattaro, G., 2004. Comparing recursion, replication and iteration in process calculi. In: ICALP'04. Lecture Notes in Computer Science 3142, 307–319.

Busi, N., Zavattaro, G., 2004. On the expressive power of movement and restriction in pure mobile ambients. Theoretical Computer Science 322, 477–515.

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

Cai, X., Fu, Y., 2011. The λ-calculus in the π-calculus. Mathematical Structure in Computer Science 21, 943–996.

Cardelli, L., Gordon, A., 2000. Mobile ambients. Theoretical Computer Science 240, 177–213.

Czerwiński, W., Hofman, P., Lasota, S., 2011. Decidability of branching bisimulation on normed commutative context-free processes. In: CONCUR'11. Lecture Notes in Computer Science 6901, 528–542.

Dasgupta, A., Papamiditriou, C., Vazirani, U., 2006. Algorithm. MaGraw-Hill.

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

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

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

Fournet, C., Gonthier, G., 1996. The reflexive chemical abstract machine and the join-calculus. In: POPL'96. ACM, 372–385.

Fournet, C., Gonthier, G., Lévy, J., Maranget, L., Rémy, D., 1996. A Calculus of Mobile Agents. In: CONCUR'96. Lecture Notes in Computer Science 1119, 406-421.

Finkel, A., Schnoebelen, P., 2001. Well-structured transition system everywhere. Theoretical Computer Science 256, 63–92.

Fu, Y., 1997. A proof theoretical approach to communications. In: ICALP'97. Lecture Notes in Computer Science 1256, 325–335.

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

Fu, Y., 2003. Bisimulation congruences of chi calculus. Information and Computation 184, 201–226.

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

Fu, Y., 2007. Fair ambients. Acta Informatica 43, 535–594.

Fu, Y., 2010. Theory by process. In: CONCUR'10. Lecture Notes in Computer Science 6296, 403–416.

Fu, Y., 2013a. Checking equality and regularity for normed BPA with silent moves. In: ICALP'13. Lecture Notes in Computer Science 7966, 244–255.

Fu, Y., 2013b. The value-passing calculus. In: Theories of Programming and Formal Methods. Lecture Notes in Computer Science 8051, 166–195.

Fu, Y., 2015. Nondeterministic structure of computation. Mathematical Structures in Computer Science 25, 1295–1338.

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

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

Fu, Y., Zhu, H., 2015. The name-passing calculus. ArXiv:1508.00093.

Garcia-Molina, H., 1982. Elections in distributed computing systems. IEEE Transactions on Computers 31, 48–59.

Giambiagi, P., Schneider, G., Valencia, F., 2004. On the expressiveness of infinite behavior and name scoping in process calculi. In: FOSSACS'04. Lecture Notes in Computer Science 2987, 226–240.

Gorla, D., 2008a. Comparing communication primitives via their relative expressive power. Information and Computation 206, 931–952.

Gorla, D., 2008b. Towards a unified approach to encodability and separation results for process calculi. In: CONCUR'08. Lecture Notes in Computer Science 5201, 492–507.

Gorla, D., 2009a. On the relative power of ambient-based calculi. In: TGC'08. Lecture Notes in Computer Science 5474, 141–156.

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

Guan, X., Yang, Y., You, J., 2001. Typing evolving ambients. Information Processing Letters 80, 265–270.

He, C., 2010. Model independent order relations for processes. In: APLAS'10. Lecture Notes in Computer Science 6461, 408–423.

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., 2007. A Distributed Pi-Calculus. CUP.

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

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

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

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

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

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

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

Hirschkoff, D., Lozes, E., Sangiorgi, D., 2002. Separability, expressiveness, and decidability in the ambient logic. In: LICS'02. IEEE Computer Society, 423–432.

Hirshfeld, Y., Jerrum, M., 1999. Bisimulation equivalence is decidable for normed process algebra. In: ICALP'99. Lecture Notes in Computer Science 1644, 72–73.

Hoare, C., 1978. Communicating sequential processes. Communications of ACM 21, 666–677.

Hoare, C., 1985. Communicating Sequential Processes. Prentice Hall.

Honda, K., Tokoro, M., 1991a. An object calculus for asynchronous communications. In: ECOOP'91. Lecture Notes in Computer Science 512, 133–147.

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

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

Hopcroft, J., Ullman, J., 1979. Introduction to Automata Theory, Languages and Computation. Addison-Wesley Publishing Company.

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

Jančar, P., Srba, J., 2008. Undecidability of bisimilarity by defender's forcing. Journal of ACM 55 (1).

Kruskal, J., 1972. The theory of well ordering: A frequently discovered concept. Journal of Combinatorial Theory, Series A 13, 297–305.

Kučera, A., Jančar, P., 2006. Equivalence-checking on infinite state systems: Techniques and results. Theory and Practice of Logic Programming 6, 227–264.

Lanese, I., Perez, J., Sangiorgi, D., Schmitt, A., 2010. On the expressiveness of polyadic and synchronous communication in higher-order process calculi. In: ICALP'10. Lecture Notes in Computer Science 6199, 442–453.

Levi, F., Sangiorgi, D., 2000. Controlling interference in ambients. In: POPL'00. ACM, 352–364.

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 915, 187–201.

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

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

Lin, H., 1998. Complete proof systems for observation congruences in finite-control π-calculus. In: ICALP'98. Lecture Notes in Computer Science 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., Hermanns, H., 2002. Axiomatising divergence. In: ICALP'02. Lecture Notes in Computer Science 2380, 585–596.

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

M. Bugliesi, G. C., Crafa, S., 2004. Access control for mobile agents: the calculus of boxed ambients. ACM Transactions on Programming Languages and Systems 26, 57–124.

Maffeis, S., Phillips, I., 2005. On the computational strength of pure ambient calculi. Theoretical Computer Science 330, 501–551.

Mayr, R., 2000. Process rewrite systems. Information and Computation 156, 264–286.

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

Merro, M., Hennessy, M., 2006. A bisimulation-based semantic theory of safe ambients. ACM Transactions on Programming Languages and Systems 28, 290–330.

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

Merro, M., Zappa Nardelli, F., 2005. Behavioural theory for mobile ambients. Journal of ACM 52, 961–1023.

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

Milner, R., 1981. Modal characterisation of observable machine behaviour. In: CAAP'81. Lecture Notes in Computer Science 112, 25–34.

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., 1992. Functions as processes. Mathematical Structures in Computer Science 2, 119–146.

Milner, R., 1993a. Elements of interaction. Communication of 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.

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

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

Natarajan, V., Cleaveland, R., 1995. Divergence and fair testing. In: ICALP'95. Lecture Notes in Computer Science 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: CONCUR'06. Lecture Notes in Computer Science 4137, 52–63.

Nestmann, U., Pierce, B., 1996. Decoding choice encodings. In: CONCUR'96. Lecture Notes in Computer Science 1119, 179–194.

Nestmann, U., Pierce, B., 2000. Decoding choice encodings. Information and Computation 156, 287–319.

Palamiddessi, C., Saraswat, V., Valencia, F., Victor, B., 2006. On the expressiveness of linearity vs. persistence in the asynchronous pi calculus. In: LICS'06. IEEE Computer Society, 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. In: Theoretical Computer Science. Lecture Notes in Computer Science 104, 167–183.

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

Parrow, J., 2006. Expressiveness of process algebras. In: LIX Colloquium'06.

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

Parrow, J., Victor, B., 1997. The update calculus. In: AMAST'97. Lecture Notes in Computer Science 1119, 389–405.

Parrow, J., Victor, B., 1998. The fusion calculus: Expressiveness and symmetry in mobile processes. In: LICS'98. IEEE Computer Society, 176–185.

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

Phillips, I., Vigliotti, M., 2002. On reduction semantics for the push and pull ambient calculus. In: TCS'02, IFIP 17th World Computer Congress, Montreal. Kluwer, 550–562.

Phillips, I., Vigliotti, M., 2006. Leader election in rings of ambient processes. Theoretical Computer Science 356, 468–494.

Phillips, I., Vigliotti, M., 2008. Symmetric electoral systems for ambient calculi. Information and Computation 206, 34–72.

Plotkin, G., 1975. Call-by-name, call-by-value, and the λ-calculus. Theoretical Computer Science 1, 125–159.

Plotkin, G., 1981. A structural approach to operational semantics. Tech. rep., Computer Science Department, Aarhus University.

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: LICS'97. IEEE Computer Society, 140–148.

Reisig, W., 1985. Petri Nets, An Introduction. EATCS Monograph on Theoretical Computer Science. Springer.

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

Roscoe, A., 1997. The Theory and Practice of Concurrency. Prentice Hall.

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: TAPSOFT'93. Lecture Notes in Computer Science 668, 151–166.

Sangiorgi, D., 1994. The lazy λ-calculus in a concurrency scenario. Information and Computation 111, 120–153.

Sangiorgi, D., 1995. Lazy functions and mobile processes. Tech. Rep. 2515, INRIA Sophia-Antipolis.

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., 2001. Asynchronous process calculi: The first-order and higher-order paradigms (tutorial). Theoretical Computer Science 253, 311–350.

Sangiorgi, D., 2009. On the origin of bisimulation and coinduction. Transactions on Programming Languages and Systems 31 (4).

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

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

Sangiorgi, D., Walker, D., 2001b. The π Calculus: A Theory of Mobile Processes. CUP.

Sewell, P., 1994. Bisimulation is not finitely (first order) equationally axiomatisable. In: LICS'94. IEEE Computer Society, 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.

Soare, R., 1987. Recursively Enumerable Sets and Degrees, a study of computable functions and computably generated sets. Springer.

Srba, J., 2004. Roadmap of infinite results. In: Formal Models and Semantics, II. World Scientific Publishing Co.

Stoller, D., 2000. Leader election in asynchronous distributed systems. IEEE Transactions on Computers 49, 283–284.

Thomsen, B., 1989. A calculus of higher order communicating systems. In: POPL'89. ACM, 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 Emde Boas, P., 1990. Machine models and simulations. In: van Leeuwen, J. (Ed.), Handbook of Theoretical Computer Science: Algorithm and Complexity, volume A. Elservier, 65–116.

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

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

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

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

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

van Glabbeek, R., 2012. Musings on Encodings and Expressiveness. In: EXPRESS/SOS'12. 81–98.

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

Vigliotti, M., Phillips, I., Palamidessi, C., 2007. Tutorial on separation results in process calculi via leader election. Theoretical Computer Science 388, 267–289.

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

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

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

Wegener, I., 2005. Complexity Theory. Springer.

Xu, X., 2012. Distinguishing and relating higher-order and first-order processes by expressiveness. Acta Informatica 49, 445–484.

Xue, J., Long, H., Fu, Y., 2011. Remark on some pi variants. Draft paper.

Yin, Q., Fu, Y., He, C., Huang, M., Tao, X. Branching bisimilarity checking for PRS. In: ICALP'14. Lecture Notes in Computer Science 8573, 363–374.

Zimmer, P., 2003. On the expressiveness of pure safe ambients. Mathematical Sturcutres in Computer Science 13, 721–770.