

Classifying Interactions via Subbisimilarity

Yuxi Fu Hao Lu

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

Abstract

Subbisimilarity is proposed as a uniform tool to classify the expressive power of the process calculi. The relative expressiveness of the variants of the Pi Calculus is studied in terms of the subbisimilarity. Some issues concerning the expressiveness of the Pi are clarified. Several open problems are solved along the way.

1 Computation and Interaction

A fundamental issue in the process theory is the expressiveness of the process calculi. The first problem one encounters when studying the expressiveness is how to pin down a reasonable notion of expressiveness. What does it mean by one process calculus being (strictly) more expressive than another? We start with an overview of some of the criteria used in the literature and some expressiveness results on π -Calculus [13].

1.1 Criteria for Expressiveness

One may start looking for the expressiveness criteria by asking the following question: What is the essential difference between the expressiveness of the process calculi and the expressiveness of the Turing computing models? Process calculi are the operational models of the *open* computing systems that formalize the interactions between systems. In this sense process theory is a theory of interaction [12]. On the other hand the Turing computing models like the λ -calculus study the *closed* computing systems whose only interactions with the environments are the input and output actions. Two Turing computable functions are compared by their impacts on the environments, which are nothing but their input-output behaviors. Generalizing this view, a process is identified by the effect it may inflict on the environments. Since processes normally interact with the environments continuously, the behaviors of the processes have to be compared in a co-inductive way.

The bisimulation approach was introduced precisely for that purpose [11, 17].

Most current results about the expressiveness of the process calculi are actually concerned with the relative expressive powers of the variants of a concurrent model, say the π -Calculus. For the purpose of this paper, we say that two process calculi \mathcal{L}_1 and \mathcal{L}_2 , whose operational semantics are defined by the labeled transition systems $(\mathcal{S}_1, \mathcal{T}_1, \longrightarrow_1)$ and $(\mathcal{S}_2, \mathcal{T}_2, \longrightarrow_2)$ respectively, are variants to each other if $\mathcal{T}_1 = \mathcal{T}_2$. In our view two variants may differ in (the definitions of) the operators they have. But they must have the same set of action labels to qualify for being variants of each other. For the variants \mathcal{L}_1 and \mathcal{L}_2 the following criteria have been used to compare their expressive powers [16, 14]:

- *Interaction Preservation* is the minimal criterion for ‘being more expressive’. Process calculi are operational models. For \mathcal{L}_2 to be at least as expressive as \mathcal{L}_1 , the former must be able to simulate the operational behaviors of the latter. This normally means that there is a translation $\llbracket - \rrbracket$ from \mathcal{L}_1 to \mathcal{L}_2 such that whenever $P_1 \xrightarrow{\lambda}_1 P_2$ then $\llbracket P_1 \rrbracket \xrightarrow{\lambda}_2 \asymp \llbracket P_2 \rrbracket$, where \asymp is an observational equivalence. From the viewpoint of interactions, the Interaction Preservation alone is just not strong enough. It may well be that $\llbracket P \rrbracket$ can engage in an action to which P does not have any counterpart action. And there is no guarantee that equivalent processes are translated to equivalent processes.
- *Full Abstraction* is often used to remove the second problem mentioned above. The property, $P \asymp Q$ if and only if $\llbracket P \rrbracket \asymp \llbracket Q \rrbracket$, meets all the criteria from an algebraic point of view. The advantage of Full Abstraction is its wide applications. It can be applied to compare two process calculi with quite different action sets, say the π -calculus [13] and the ambient calculus [5]. From the viewpoint of interaction, Full Abstraction is not of much use without Interaction Preservation. A translation from \mathcal{L}_1 to \mathcal{L}_2 may still suffer from the first problem even if it satisfies the Full Abstraction and Interaction Preservation. Full Abstraction is, in our view, best seen as a fallout property rather than a criterion.

- *Interaction Reflection* requires that whatever $\llbracket P \rrbracket$ can do are the translations of the actions of P . Formally $\llbracket P \rrbracket \xrightarrow{\lambda}_2 P_2$ implies that $P \xrightarrow{\lambda}_1 P_1$ for some P_1 such that $\llbracket P_1 \rrbracket \simeq P_2$. Interaction Reflection guarantees that $\llbracket P \rrbracket$ can not place more effect on an environment than P . It is easy to see that *Interaction Criterion* (Interaction Preservation + Interaction Reflection) normally subsumes the Full Abstraction with respect to \simeq .
- *Equivalence Criterion* is used in [16] to compare two process calculi which are variants to each other. If for every P in \mathcal{L}_1 there is some P' in \mathcal{L}_2 such that $P \simeq P'$, then \mathcal{L}_2 is at least as expressive as \mathcal{L}_1 . Like Full Abstraction, it is subsumed by the Interaction Criterion. But unlike Full Abstraction, it can not be applied to compare process calculi with different action sets.

Other criteria have been used in the previous studies. In [16], Palamidessi used *Uniformity* and *Reasonability* as criteria to compare the expressiveness of the π -calculi. These criteria are weaker than the Interaction Criterion from the viewpoint of interaction. In [3, 4], Busi, Gabbriellini and Zavattaro made use of decidability to tell apart process calculi. How can computational properties be used to classify interactions? Well in a theory of interaction, computations are internal interactions of systems. If a process is exclusively engaged in an infinite computation, the process will never get a chance to interact externally. So computations may interfere with interactions by not giving the latter any chance! An encoding $\llbracket _ \rrbracket$ is said to meet *Computation Criterion* if it satisfies the property that P is terminating if and only if $\llbracket P \rrbracket$ is terminating, where P is terminating if it does not have any infinite sequence of τ actions. The importance of this criterion has been emphasized in the previous work.

1.2 Results and Open Issues

We now sketch some of the relative expressiveness results about the variants of the π -calculus. Let π be the π -Calculus with the binary choice, π^m the π with the mixed choice, π^s the π with the separated choice, π^i the π with the input choice, π^- the π without the choice, and π^a the asynchronous π . In [16], Palamidessi summarized the existing relative expressiveness of these variants by the diagram in Figure 1. The sub-calculus relationship is indicated by the hooked arrow \hookrightarrow . The plain arrow \rightarrow indicates the existence of a uniform and reasonable encoding. Boudol, who proposed π^a , gave in [2] a translation from π^- to π^a that preserves a Morris style contextual equivalence. Honda and Tokoro provided a different encoding whose correctness is proved for some weak bisimilarity [10]. Their notion of the asynchronous bisimilarity has been subsequently adopted as the standard equivalence for the asynchronous calculi. Nestmann and Pierce studied in [14] two encodings from

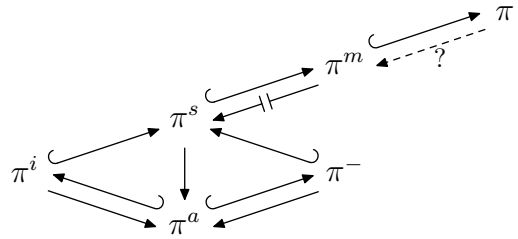


Figure 1. Encodings Between Pi Variants

π^i to π^a . One is fully abstract with respect to the asynchronous bisimilarity but not termination preserving. The other is termination preserving but not fully abstract. Nestmann has examined some encodings of π^s in π^a [15] and argued without a proof that no fully abstract translation from the former to the latter exists. Palamidessi proved in [16] that there is no uniform and reasonable encoding from π^m to π^s .

How forceful are the results stated in Figure 1? One fact that must be pointed out is that two uniform and reasonable encodings could exhibit quite different interactional behaviors. For example there is an encoding from π^i to π^a that satisfies the Interaction Criterion [14]. On the other hand the known encoding from π^s to π^a is much weaker [15]. There are certainly processes in π^s whose interactive effect on the environments is different from any process in π^a . The second point is that the subcalculus relationships indicated in Figure 1 are not all trivial. For the identity encoding from π^a to π^- , how do we reconcile the asynchronous equivalence of the former to the (synchronous) equivalence of the latter? What does the Interaction Criterion mean in this case? Another important point is that the negative results, crucial to the classification of the expressive hierarchy of the process calculi, depend on our choice of the expressiveness criteria. The selection of a set of criteria could be subtle because both the positive results and the negative results should speak with the same potency. For instance, the nonexistence of a uniform and reasonable encoding from π^m to π^s does not rule out the existence of a fully abstract encoding from π^m to π^s with respect to the weak bisimilarity. On the other hand it seems that the arrow from π^m to π^a , as indicated in Figure 1, can be understood as anything but a fully abstract encoding with respect to the weak bisimilarity. Finally there is a point about the Equivalence Criterion. One way to understand the negative result of Figure 1 is that there is no encodings from π^m to π^s that satisfy the Equivalence Criterion and the Computation Criterion. Even if we relax that the arrows $\pi^s \rightarrow \pi^i$, $\pi^s \rightarrow \pi^-$, $\pi^s \rightarrow \pi^a$ and $\pi^- \rightarrow \pi^a$ fail the Equivalence Criterion. Using the simple arguments, as we shall see later, one can show that

the ‘tossing-the-coin algorithm’, which can be coded up in π^s by $\bar{c}f + \bar{c}t$, can not be programmed in any of the π^i , π^a and π^- . These negative results are in a sense even stronger than the negative result offered by the Leader Election System [16].

According to the above analysis, there is really a need to clarify the picture about the relative expressiveness of the π -variants. It would be fruitful to look at the models from the point of view of interactability. The computational factor should also be considered since they interfere with the interactions. In other words we expect to strengthen the results stated in Figure 1 by answering the following:

Question 1: What are the relationships among the variants of π -Calculus if the Interaction Criterion is adopted? What if the Computation Criterion is also adopted?

In Figure 1 the asynchronous π -calculi are compared against the synchronous π -calculi. The equivalences used in the full abstraction results are the asynchronous bisimilarity in the former and the various (synchronous) bisimilarities in the latter. A better understanding of the relationships can be achieved by considering the following:

Question 2: What is the uniform framework to study the expressiveness of the synchronous π -calculi and the asynchronous π -calculi?

To answer the two questions, some open problems need be solved. The relative expressiveness of π over π^m has stood unanswered for quite some time. From the viewpoint of interactions, the problem can be formalized as follows:

Problem 1: Is there a (termination preserving) encoding $\llbracket - \rrbracket$ from π to π^m such that $P \approx \llbracket P \rrbracket$?

The answer is not just practically interesting, it is also theoretically important. Similarly the relative expressiveness of π^m over π^s is not completely clear, despite the result in [16]. Hence

Problem 2: Is there a (termination preserving) encoding $\llbracket - \rrbracket$ from π^m to π^s such that $P \approx \llbracket P \rrbracket$?

The above two problems are about the relative powers of the choices. There is also a question on whether any form of the external choice is redundant.

Problem 3: Is there an encoding $\llbracket - \rrbracket$ from π^s to π^- such that $P \approx \llbracket P \rrbracket$?

The choice operator is often used in specification [11], whereas the concurrent composition ‘|’ is an implementation operator. Problem 3 can be equivalently stated as follows: Is there a specification that can not be implemented without the choice operator? The asynchronous π -calculus

is often presented without the choice. But intuitively π^a is strictly less expressive than all the synchronous π -calculi interactively since the output actions do not have any continuations. So we pose the following:

Problem 4: Is π^a , and π^i as well, strictly less expressive than π^- interactively?

Sometimes two variants only differ in the ways the infinite behaviors are introduced. The well known recursion mechanism used in the π -Calculus are the fixpoint operations, recursive constant definitions, replications etc.. It would be interesting to see if different recursion mechanisms make a difference interactively

Problem 5: What are the comparative expressiveness of the various recursion mechanisms from the viewpoint of interaction?

Solutions to the above five questions should allow us to have an improved understanding of the relative expressiveness of the π -calculi appeared in Figure 1.

1.3 Contributions

This paper sets out to answer the two questions and solve the five problems stated in the previous subsection. Our contributions are both methodological and technical.

- We will propose a methodology to compare the expressive powers of the process calculi in a uniform manner. This methodology is inspired by the Interaction Criterion. The advantage of our approach is that it fits well with the observational theory. It can also be easily adapted if different observational theories are adopted.
- Technically we introduce subbisimilarity and asynchronous subbisimilarity as tools to measure the relative expressiveness of the process calculi. The former works for the synchronous calculi and the latter for both the synchronous and the asynchronous calculi. Computation Criterion can be easily incorporated. Using the asynchronous (termination preserving) subbisimilarity, we clarify all the relationships among the variants of the π -Calculus present in Figure 1. The theoretical framework also allows us to study the independence of the operators of the process calculi. We show that all the operators of the π -calculus are independent to each other.

The paper is organized as follows: The subbisimilarities for CCS variants are introduced in Section 2 to study the expressiveness of the choice operators and the recursion. The approach is extended to the setting of the π -calculus in Section 3 to investigate the relative expressiveness of the synchronous variants of the π -calculus. It is further extended in Section 4 to cover the asynchronous π -calculi. Some comments are made in Section 5.

2 Interaction Hierarchy of CCS

As an operational model of interaction [12], CCS [11] is a test bed for studying the expressiveness of process calculi. Apart from serving as a technical preamble, the results in this section are also interesting on their own.

We shall assume that there is a set of names \mathcal{N} ranged over by the small letters. The set of action labels $\mathcal{N} \cup \bar{\mathcal{N}} \cup \{\tau\}$ is ranged over by λ . The syntax of CCS is as follows:

$$E ::= 0 \mid X \mid \lambda.E \mid E \mid E' \mid (x)E \mid E+E' \mid \mu X.E$$

We have left out the relabeling operation for the following reason: Our interest in CCS is to obtain results that can be easily transplanted to the π -Calculus. For that purpose the relabeling operation is not necessary.

The standard operational semantics of CCS is inductively generated by the following rules:

Prefix, Restriction

$$\frac{}{\lambda.E \xrightarrow{\lambda} E} \quad \frac{E \xrightarrow{\lambda} E' \quad x \notin n(\lambda)}{(x)E \xrightarrow{\lambda} (x)E'}$$

Composition

$$\frac{E \xrightarrow{\lambda} E'}{E \mid F \xrightarrow{\lambda} E' \mid F} \quad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{\bar{a}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

Choice, Fixpoint

$$\frac{E \xrightarrow{\lambda} E'}{E+F \xrightarrow{\lambda} E'} \quad \frac{E\{\mu X.E/X\} \xrightarrow{\lambda} E'}{\mu X.E \xrightarrow{\lambda} E'}$$

Notice that the symmetric rules are omitted. We will write \Longrightarrow and $\xRightarrow{\lambda}$ for their standard interpretations. The α -conversion applies to this version of CCS. The standard reference book on CCS [11] covers the fundamental theory about the model. We will assume that the reader is familiar with the labeled transition semantics and the basic operators in process calculi. We will use the standard notations in process calculi. For instance we write σ to stand for substitutions of names like $\{y_1/x_1, \dots, y_n/x_n\}$. We write $\{E_1/X_1, \dots, E_n/X_n\}$ for substitutions of variables. When applying the substitution to a process expression E , notation $E\{E_1/X_1, \dots, E_n/X_n\}$, we get a process expression obtained by replacing X_1, \dots, X_n by E_1, \dots, E_n respectively. Bound names need be renamed to avoid name capture. For instance $(x)(y)(\bar{c}x \mid \bar{d}y \mid X)\{\bar{a}x \mid \bar{b}y/X\}$ is the expression $(v)(w)(\bar{c}v \mid \bar{d}w \mid \bar{a}x \mid \bar{b}y)$. Sometimes, especially when studying dynamic binding calculi, we need substitutions that does not admit α -conversion. For that purpose we introduce the dynamic substitution of variables $[E_1/X_1, \dots, E_n/X_n]$. For instance the syntactic object $(x)(y)(\bar{c}x \mid \bar{d}y \mid X)[\bar{a}x \mid \bar{b}y/X]$ is the expression

$(x)(y)(\bar{c}x \mid \bar{d}y \mid \bar{a}x \mid \bar{b}y)$. The substitutions of variables will be ranged over by ς .

A descendant of a process expression E is an E' such that $E \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_n} E'$, where $n \geq 0$. Let $\mathcal{Dscd}(E)$ be the set of descendants of E . By definition $E \in \mathcal{Dscd}(E)$. A computation of E is either an infinite internal action sequence $E \xrightarrow{\tau} \dots \xrightarrow{\tau} E_i \xrightarrow{\tau} \dots$ or a finite internal action sequence $E \xrightarrow{\tau} \dots \xrightarrow{\tau} E'$ such that E' can not perform any internal actions. Let $\mathcal{Comp}(E)$ be the set of all computations of E . The expression E is *terminating* if $\mathcal{Comp}(E)$ does not contain any infinite computation. It is *divergent* otherwise.

In order to compare the expressive powers of the variants of CCS, we introduce the subbisimilarity between the variants. For that purpose we need the following definition:

Definition 1. A binary relation $\mathcal{R} \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ is left surjective if $\forall S_1 \in \mathcal{S}_1. \exists S_2 \in \mathcal{S}_2. S_1 \mathcal{R} S_2$. It is right surjective if $\forall S_2 \in \mathcal{S}_2. \exists S_1 \in \mathcal{S}_1. S_1 \mathcal{R} S_2$.

Intuitively a subbisimilarity interprets every process of a variant by (at least) one process in another variant in such a way that the related processes have the same operational impacts on the environments. In other words the related processes satisfy the Interaction Criterion. In Milner's terminology [11, 17], a subbisimilarity is a left surjective bisimulation between *two* process calculi. The following definition is almost the same as the definition of bisimulation in [9]. Throughout this paper we confine our attention to the relations on the processes without the free process variables.

Definition 2. A subbisimilarity from the variant CCS^1 to the variant CCS^2 is a left surjective relation \mathcal{R} such that the following bisimulation property holds whenever $S_1 \mathcal{R} S_2$:

$$\text{If } S_1 \xrightarrow{\lambda} S'_1 \text{ then } S_2 \xRightarrow{\lambda} S'_2 \mathcal{R}^{-1} S'_1 \text{ for an } S'_2;$$

$$\text{If } S_2 \xrightarrow{\lambda} S'_2 \text{ then } S_1 \xRightarrow{\lambda} S'_1 \mathcal{R} S'_2 \text{ for some } S'_1.$$

We shall say that CCS^1 is subbisimilar to CCS^2 , notation $\text{CCS}^1 \sqsubseteq^{ccs} \text{CCS}^2$, if there is a subbisimilarity from CCS^1 to CCS^2 . We write $\text{CCS}^1 \sqsubset^{ccs} \text{CCS}^2$ if $\text{CCS}^1 \sqsubseteq^{ccs} \text{CCS}^2$ and $\text{CCS}^2 \not\sqsubseteq^{ccs} \text{CCS}^1$.

The subbisimilarity relationship is transitive. A subbisimilarity from a CCS variant to itself is a special bisimulation [11]. The largest such bisimulation is called *bisimilarity*, denoted by \approx . Alternatively $P \approx Q$ if there is a right surjective subbisimilarity from $\mathcal{Dscd}(P)$ to $\mathcal{Dscd}(Q)$.

In sequel we often say that an encoding $\llbracket _ \rrbracket$ from CCS^1 to CCS^2 is a subbisimilarity. This should be understood that $\llbracket _ \rrbracket$ is contained in a subbisimilarity. Such an encoding clearly satisfies the Full Abstraction with respect to the bisimilarity \approx .

To prove $\text{CCS}^2 \sqsubset^{ccs} \text{CCS}^1$, one only has to show that some P_2 of CCS^2 exists such that for every P_1 of CCS^1 the pair P_2, P_1 do not satisfy the bisimulation property.

2.1 Computation and Bisimulation

In process theory, what are the rules that govern the behaviors of the computations? From the viewpoint of interaction, what can be said about the processes in the sequence $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} P_2 \dots P_i \xrightarrow{\tau} P_{i+1} (\xrightarrow{\tau} \dots)$? A not well known fact is that if some P_i is equivalent to the initial state P_0 then all the intermediate states are equivalent. The following lemma has already been stated in [8].

Lemma 1 (O-Lemma). *If $P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots P_n \xrightarrow{\tau} P_0$ then $P_0 \approx P_1 \approx P_2 \approx \dots \approx P_n$.*

According to the O-Lemma a computation is carried out in phases, which can be depicted as follows:

$$P_0 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_{i_1} \xrightarrow{\tau} P_{i_1+1} \xrightarrow{\tau} \dots \xrightarrow{\tau} P_{i_2} \xrightarrow{\tau} \dots$$

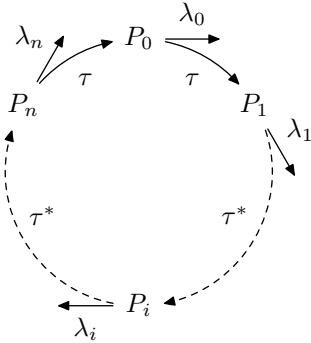
The phase one consists of all the states from P_0 to P_{i_1} . These states are all equivalent from the point of view of interaction. From the point of a computing system, what phase one accomplishes is some internal adjustment. It has not made any irretrievable decision. The computing step from P_{i_1} to P_{i_1+1} is fundamental. Once the system has done that, there is no going back. The second phase starting from P_{i_1+1} and ending at P_{i_2} consists of the equivalent states again. These states are *not* equivalent to *any* of the previous states. Generally none of the states in a phase is equivalent to any state in the previous phases.

In practice the O-Lemma is used in the following form:

$$\text{If } P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \dots \xrightarrow{\tau} P_n \asymp P_0 \text{ then } P_0 \asymp P_1 \asymp P_2 \asymp \dots \asymp P_n.$$

The symbol \asymp stands for an observational equivalence.

The O-Lemma is of crucial importance to some fully abstract encodings studied in this paper. A typical scenario of using the O-Lemma is the following:



Imagine that there is an agent traveling around the states of the circle. All the states on the circle are equivalent by the O-Lemma. So as long as the agent is staying on the circle, it retains all the powers to kick off any of the possible actions doable by P_i . But at any particular state, say P_i , the agent might decide to kick off the action λ_i . Once it does that, the system moves to the next step.

2.2 Choice, or No Choice

Instead of the binary choice $+$, one could have the *mixed choice* $\sum_{i \in I} \lambda_i.E_i$, or the *separated choice* $\sum_{i \in I} a_i.E_i$ and $\sum_{i \in I} \bar{a}_i.E_i$, where I is finite. The operational semantics of the mixed/separated choice is defined as follows:

$$\overline{\sum_{i \in I} \lambda_i.E_i} \xrightarrow{\lambda_i} E_i$$

We write CCS^m and CCS^s for the CCS with mixed choice, respectively separated choice. In both CCS^m and CCS^s the prefix operator could be dropped.

Let CCS^- be the CCS without the choice operator. It is not surprising that CCS^- is less expressive than CCS^s in terms of interactability. What is surprising is that CCS^s and CCS^m are equivalent interactively.

Theorem 1. $\text{CCS}^- \sqsubseteq^{\text{ccs}} \text{CCS}^m \sqsubseteq^{\text{ccs}} \text{CCS}^s$.

Proof. It is clear that $\text{CCS}^- \sqsubseteq^{\text{ccs}} \text{CCS}^m$. We have to show that $\text{CCS}^m \not\sqsubseteq^{\text{ccs}} \text{CCS}^-$. Suppose that P is a process containing no choice operations and that $a+b \approx P$ for distinct a, b . To match up the action $a+b \xrightarrow{a} 0$ there must be some P' such that $P \Longrightarrow P' \xrightarrow{a}$. Now $P' \approx a+b$. Therefore P'' exists such that $P' \Longrightarrow P'' \xrightarrow{b}$. Using the fact that P' contains no choice operator, it is easy to show that $P'' \xrightarrow{a}$ (see Lemma 2 below). But then it follows from Lemma 2 that $P'' \xrightarrow{a} \xrightarrow{b}$, which contradicts to the fact that $P'' \approx a+b$.

Next we show that $\text{CCS}^m \sqsubseteq^{\text{ccs}} \text{CCS}^s$. It is enough to see how to simulate the mixed choice $a.P+\bar{b}.Q$ by process A , or B , defined as follows:

$$\begin{aligned} A &= a.P+\tau.B \\ B &= \bar{b}.Q+\tau.A \end{aligned}$$

Using the fixpoint operator one could get the solution to the above equations:

$$\begin{aligned} A &\stackrel{\text{def}}{=} a.P+\tau.\mu Y.(\bar{b}.Q+\tau.(a.P+\tau.Y)) \\ B &\stackrel{\text{def}}{=} \mu Y.(\bar{b}.Q+\tau.(a.P+\tau.Y)) \end{aligned}$$

Obviously $A \approx a.P+\bar{b}.Q \approx B$ and A, B are in CCS^s . Clearly this is a simple application of the O-Lemma. \square

Lemma 2. *Suppose E is a CCS^- expression. If $E \xrightarrow{\lambda_1} E'$ and $E \xrightarrow{\lambda_2} E''$ such that $\lambda_1 \neq \lambda_2$, then $E \xrightarrow{\lambda_1} E' \xrightarrow{\lambda_2} E'''$ and $E \xrightarrow{\lambda_2} E'' \xrightarrow{\lambda_1} E'''$ for some E''' .*

Proof. This is a simple induction on the height of derivations. \square

2.3 Termination Preserving Encoding

The encoding given in the previous subsection is assuring from an external point of view. No environments can ever detect any difference between P and $\llbracket P \rrbracket$. There is however something lost over the translation. The process $a.P + \bar{b}.Q$ is translated to $a.P + \tau.\mu Y.(\bar{b}.Q + \tau.(a.P + \tau.Y))$ which is divergent. The point is that although the divergent computations introduced by the encoding only go through states that are equivalent to the initial state, there is arguably some difference between a never ending computation and a terminating computation. One is interested in knowing if Theorem 1 can be strengthened to meet the Computation Criterion.

Definition 3. A subbisimilarity \mathcal{R} from CCS^1 to CCS^2 is termination preserving if it meets the Computation Criterion in the following sense:

If PRQ then P is terminating if and only if Q is terminating.

We write $\text{CCS}^1 \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^2$ if a termination preserving subbisimilarity from CCS^1 to CCS^2 exists, and $\text{CCS}^1 \sqsubset_{\downarrow}^{\text{ccs}} \text{CCS}^2$ if $\text{CCS}^1 \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^2$ yet $\text{CCS}^2 \not\sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^1$.

The next theorem says that one must be ready to sacrifice Computation Criterion if the use of the choice is restricted.

Theorem 2. $\text{CCS} \not\sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^m \not\sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^s$.

Proof. We prove that there are no termination preserving subbisimilarities from CCS, respectively CCS^m , to CCS^m , respectively CCS^s .

Suppose that P is a terminating process with only separated choice. We argue that P is not bisimilar to $a + \bar{b}$. If not, then for each P' such that $P \Longrightarrow P'$ and $\neg(P' \xrightarrow{\tau})$, one would have that $P' \approx a + \bar{b}$, which would imply that $P' \xrightarrow{a}$ and $P' \xrightarrow{\bar{b}}$. This is an obvious contradiction.

Next we prove that the process $\mu X.(a + b | X)$, which is in CCS and is terminating, is not bisimilar to any terminating process in CCS^m . Notice that $\mu X.(a + b | X)$ is terminating and infinite branching and that it turns into a process of the form $\underbrace{b | \dots | b}_{j \text{ times}}$ after performing an a action.

If $\mu X.(a + b | X)$ were bisimilar to some terminating P of CCS^m , then according to the finite branching property of CCS^m , there are only finitely many P_i , for $i \in I$, such that $P \xrightarrow{a} P_i$. If P were bisimilar to $\mu X.(a + b | X)$ then for each $i \in I$ there would exist some $j_i < \omega$ such that

$$\mu X.(a + b | X) \xrightarrow{a} \underbrace{b | \dots | b}_{j_i \text{ times}} \approx P_i$$

Let j_m be such that $j_m > \sup_{i \in I} \{j_i\}$. It is clear that the action $\mu X.(a + b | X) \xrightarrow{a} \prod_{1 \leq j \leq j_m} b$ can not be matched up by P . \square

2.4 Static Binding, or Dynamic Binding

The infinite behaviors of CCS could be defined in a number of ways. Using the fixpoint operator μ , interesting process behaviors can be defined. Consider the process $A \stackrel{\text{def}}{=} \mu X.(a + b | X)$. A finite action trace of the process takes the following shape

$$A \xrightarrow{b} \dots \xrightarrow{b} A \xrightarrow{a} A \xrightarrow{b} \dots \xrightarrow{b} \mathbf{0}$$

$i \text{ times} \qquad \qquad \qquad j \text{ times}$

At first glance the interactional behavior of A has a lot to do with the fact of A being not finitely branching. But what really causes the particular pattern of the infinite behavior is the fact that X is unguarded. For the CCS we have defined, finite branching property is available if one restricts to CCS^m .

Another way to achieve the finite branching property is to use the replicator with its semantics formulated *a la* Giambiagi, Schneider and Valencia [9]:

$$\frac{E \xrightarrow{\lambda} E'}{!E \xrightarrow{\lambda} E' | !E} \qquad \frac{E \xrightarrow{a} E' \quad E \xrightarrow{\bar{a}} E''}{!E \xrightarrow{\tau} E' | E'' | !E}$$

Let $\text{CCS}^!$ denote the variant of CCS whose infinite behaviors are defined by the replicator. Intuitively one has that $!P \approx P | !P$. One obvious advantage of using the replication instead of the fixpoint operation is that one could give a purely first order presentation of CCS. Giambiagi, Schneider and Valencia have shown that the guarded μ and the replicator $!$ are equivalent interactively. This result is better stated in terms of CCS^m .

Fact 1 (Giambiagi, Schneider, Valencia [9]). *The following holds:* $\text{CCS}^m \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^! \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^m$.

Proof. Intuitively $!P$ of $\text{CCS}^!$ can be interpreted in CCS^m as $\mu X.(P | X)$. Conversely $\mu X.E$ of CCS^m is simulated in $\text{CCS}^!$ by $(m)(E\{\bar{m}.\mathbf{0}/X\} | !m.E\{\bar{m}.\mathbf{0}/X\})$, where m does not occur in E . \square

The above equivalence still leaves us with the open problem: What is the relationship between CCS and $\text{CCS}^!$?

Problem 1. $\text{CCS} \sqsubseteq^{\text{ccs}} \text{CCS}^!$?

The infinite behaviors can also be admitted through *constant definition* [11] and *parametric definition* [9]. We shall spend a minute to explain the mechanism of constant definition. Suppose E_1, \dots, E_n are expressions with free process variables X_1, \dots, X_n . The following form a finite set of constant definitions:

$$\begin{aligned} C_1 &= E_1\{C_1/X_1, \dots, C_n/X_n\} \\ &\vdots \\ C_n &= E_n\{C_1/X_1, \dots, C_n/X_n\} \end{aligned}$$

We mention that we do not allow recursively defined definitions using an infinite set of equations. A prominent feature of the constant mechanism is that it disallows the α -conversion. Take for instance the constants defined below:

$$\begin{aligned} D_1 &= \bar{x}y \mid (x)(y(z).\bar{z}x \mid D_2) \\ D_2 &= \bar{y}x \mid (y)(x(z).\bar{z}y \mid D_1) \end{aligned}$$

The free y in the definition of D_2 is captured by the localization operator (x) in the definition of D_1 dynamically. One has that

$$\begin{aligned} D_1 &\xrightarrow{\tau} \bar{x}y \mid (x)(\bar{x}x \mid (y)(x(z).\bar{z}y \mid D_1)) \\ &\xrightarrow{\tau} \bar{x}y \mid (x)(\bar{x}x \mid (y)(\bar{y}y \mid (x)(y(z).\bar{z}x \mid D_2))) \end{aligned}$$

These internal actions are possible only if bound names are never renamed.

Yet another way of introducing the infinite behaviors is by using the *dynamic fixpoint*. For the dynamic binding μ -operator, free names might get bound dynamically while unfolding the recursion. For instance $\mu X.x \mid (x)(\bar{x} \mid X)$ may not do any action if the static binding is adopted. It is divergent if the dynamic binding is admitted. For the static fixpoint operations the α -conversion is admitted, whereas for the dynamic fixpoint operations the α -conversion is banned.

Let CCS^{def} be the finite CCS with the constant definition, CCS^{pdef} be the finite CCS with the parametric definition, and $\text{CCS}^{d\mu}$ be the finite CCS with the dynamic μ -operator. It should be pointed out that the α -conversion do not apply to these three variants. For uniformity, we sometimes write CCS^μ for CCS with the static μ -operator.

Fact 2 (Giambiagi, Schneider, Valencia [9]). *One has that $\text{CCS}^{d\mu} \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{\text{def}} \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{\text{pdef}} \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{d\mu}$.*

Proof. For a proof of $\text{CCS}^{\text{def}} \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{\text{pdef}} \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{\text{def}}$ the reader is referred to [9]. The conversions between $\text{CCS}^{d\mu}$ and CCS^{def} are relevant to this paper. So we take some time to explain them. Given a process P in $\text{CCS}^{d\mu}$, we may convert it into a finite set of constant definitions in the following manner:

- To start with we introduce the constant $C = P$.
- Suppose the conversion has introduced the following set of constant definitions:

$$\begin{aligned} C &= E_0 \\ C_1 &= E_1 \\ &\vdots \\ C_n &= E_n \end{aligned}$$

If $\mu X.E$ is a subexpression in one of E_1, \dots, E_n such that E does not contain any occurrence of the μ -operator, then we replace the above set of definitions by the following set of definitions

$$\begin{aligned} C &= E_0\{C_{n+1}/\mu X.E\} \\ C_1 &= E_1\{C_{n+1}/\mu X.E\} \\ &\vdots \\ C_n &= E_n\{C_{n+1}/\mu X.E\} \\ C_{n+1} &= E_{n+1}\{C_{n+1}/X\} \end{aligned}$$

- The second step is repeated until there is no occurrence of μ -operator in any of the definitions.

It is easy to see that P is strongly bisimilar to C .

Next we explain the converse conversion. Without loss of generality, suppose we have in CCS^{def} the following set of constant definitions:

$$\begin{aligned} C_1 &= E_1\{C_1/X_1, C_2/X_2\} \\ C_2 &= E_2\{C_1/X_1, C_2/X_2\} \end{aligned}$$

Now C_1, C_2 form a solution to the following equations

$$\begin{aligned} X_1 &= E_1 \\ X_2 &= E_2 \end{aligned}$$

Using the dynamic μ -operator of $\text{CCS}^{d\mu}$, one could construct the process expression $\mu X_1.E_1$. Substituting $\mu X_1.E_1$ for X_1 in the second equation gives rise to the equation

$$X_2 = E_2\{\mu X_1.E_1/X_1\}$$

from which we could construct $\mu X_2.E_2\{\mu X_1.E_1/X_1\}$. It is easy to see that

$$\begin{aligned} C_1 &\sim \mu X_1.E_1\{\mu X_2.E_2\{\mu X_1.E_1/X_1\}/X_2\} \\ C_2 &\sim \mu X_2.E_2\{\mu X_1.E_1/X_1\} \end{aligned}$$

We are done. \square

Let CCS^{d-} be the variant of CCS^- with the dynamic fixpoint operator. Similarly we have CCS^{dm} and CCS^{ds} . What is the relationship between CCS^m and CCS^{dm} ? It is easily seen that $\text{CCS}^m \sqsubseteq_{\downarrow}^{\text{ccs}} \text{CCS}^{dm}$. Busi, Gabbrielli and Zavattaro have shown in [3, 4] that the termination property is decidable in CCS^m but undecidable in CCS^{dm} . It is important to notice that this result *per se* is *not* about the interactability! However the dynamic recursion is indeed strictly stronger interactively than the static recursion in the setting of CCS. In other words, CCS^{dm} can not be encoded in CCS^m .

Theorem 3. *The following properties hold:*

- (i) $CCS^- \sqsubseteq^{ccs} CCS^{d-}$;
- (ii) $CCS^s \sqsubseteq^{ccs} CCS^{ds}$;
- (iii) $CCS^m \sqsubseteq^{ccs} CCS^{dm}$.

Proof. Consider the counter Z defined in CCS^{dm} :

$$\begin{aligned} Z &= z.Z + a.(x)(O | x.Z) \\ O &= \bar{a}.\bar{x} + a.(y)(E | y.O) \\ E &= \bar{a}.\bar{y} + a.(x)(O | x.E) \end{aligned}$$

This is a simplification of the counter defined in [3]. It has the following operational behavior

$$Z \xrightarrow{\underbrace{a \rightarrow \dots \rightarrow a}_{i \text{ times}}} \xrightarrow{\underbrace{\bar{a} \rightarrow \dots \rightarrow \bar{a}}_{i \text{ times}}} \xrightarrow{z}$$

for every natural number i . The process Z is capable of remembering the difference between the number of a 's it has performed and the number of \bar{a} 's it has done. After Z has done consecutive n a 's, it must do at least n \bar{a} 's before it can do a z . This dynamic behavior can not be captured by any process in CCS^m with the static fixpoint operator.

In view of Fact 1 we only have to show that Z is not bisimilar to any process in $CCS^!$. The proof of this fact bears resemblance to the decidability proof of [3, 4], making use of the theory developed in [6]. See Appendix A for a proof.

It is easy to see how to modify the above proof to show that $CCS^s \sqsubseteq^{ccs} CCS^{ds}$. In CCS^{d-} , and CCS^- as well, one could define the *internal choice* $E \dot{+} F$ as follows:

$$E \dot{+} F \stackrel{\text{def}}{=} (m)(m.E | m.F | \bar{m})$$

where m does not appear in $E | F$. Using the internal choice one could define a weak form of counter in CCS^{d-} in the following manner

$$\begin{aligned} Z' &= \tau.z.Z' \dot{+} \tau.a.(x)(O' | x.Z') \\ O' &= \tau.\bar{a}.\bar{x} \dot{+} \tau.a.(y)(E' | y.O') \\ E' &= \tau.\bar{a}.\bar{y} \dot{+} \tau.a.(x)(O' | x.E') \end{aligned}$$

Although Z' does not always do what the environments want, it does exhibit the same memory capacity as Z . For this reason the approach used in Appendix A can be applied to prove (i). \square

The fact that the dynamic fixpoint operations, or the constant definitions, are much stronger than the static fixpoint operations, or the replications, is reinforced by another fact. The CCS with dynamic fixpoint operations and the mixed choices is strong enough to code up the CCS with arbitrary choices. It is a phenomenal result that not only points out the power of the constant definitions, but also pinpoints the difference between guarded choices and arbitrary choices.

$$\begin{aligned} C_{\mathbf{0}}^b &= \bar{x}_b \\ C_{\lambda.E}^b &= \bar{x}_b + \lambda.(x_b)C_E^b \\ C_{(a)E}^b &= (a)C_E^b \\ C_{E_1+E_2}^b &= \bar{x}_b + \tau.(x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.C_{E_1+E_2}^b) \\ &\quad + \tau.(x_{\bar{b}})(C_{E_2}^{\bar{b}} | x_{\bar{b}}.C_{E_1+E_2}^b) \\ C_{E_1|E_2}^b &= \bar{x}_b + \tau.(u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b) \\ V_{E_1|E_2}^b &= \tau.(x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.V_{E_1|E_2}^b) + u.C_{E_1|E_2}^b \\ W_{E_1|E_2}^b &= \tau.(x_{\bar{b}})(C_{E_2}^{\bar{b}} | x_{\bar{b}}.W_{E_1|E_2}^b) + \bar{u} \end{aligned}$$

where $b \in \{\top, \perp\}$, $\bar{b} = \neg b$ and x_{\top}, x_{\perp} are fresh names.

Figure 2. Translation from $CCS^{d\mu}$ to CCS^{dm}

Theorem 4. $CCS^{d\mu} \sqsubseteq^{ccs} CCS^{dm} \sqsubseteq^{ccs} CCS^{ds}$.

Proof. The proof $CCS^{dm} \sqsubseteq^{ccs} CCS^{ds}$ can be readily copied from the proof of Theorem 1. To establish $CCS^{d\mu} \sqsubseteq^{ccs} CCS^{dm}$ we demonstrate how to translate a $CCS^{d\mu}$ process to a CCS^{dm} process. The translation consists of four steps:

1. First a translation from the $CCS^{d\mu}$ expressions that contain no μ -operations to the finite sets of constant definitions in CCS^{dm} is defined as given in Figure 2. For each $CCS^{d\mu}$ expression E without μ -operator, a constant C_E^b is introduced. The translation is indexed by the boolean value b , which is either truth \top or false \perp , and \bar{b} is the negation of b . The name x_{\top} and x_{\perp} are used to trace back to the starting point of an internal wandering. The alternating use of the bound names x_{\top}, x_{\perp} is essential.
2. Secondly each $CCS^{d\mu}$ process P with n occurrences of the μ -operator is unrolled to a set of n equations

$$\begin{aligned} X_1 &= E_P^1 \\ &\vdots \\ X_n &= E_P^n \end{aligned}$$

in the way as explained in the proof of Fact 2. Let E_P be the $CCS^{d\mu}$ expression obtained by replacing the n fixpoints by the corresponding process variables X_1, \dots, X_n . We then have a new equation

$$X = E_P$$

for a fresh process variable X .

3. Thirdly we have a set of constant definitions in

CCS^{dm}. This set consists of the following definitions

$$\begin{aligned} C_X^\top &= C_E^\top \\ C_{X_1}^\top &= C_{E_P^1}^\top \\ &\vdots \\ C_{X_n}^\top &= C_{E_P^n}^\top \\ C_{X_1}^\perp &= C_{E_P^1}^\perp \\ &\vdots \\ C_{X_n}^\perp &= C_{E_P^n}^\perp \end{aligned}$$

plus the constant definitions introduced by the constants $C_{E_P^1}^\top, \dots, C_{E_P^n}^\top, C_{E_P^1}^\perp, \dots, C_{E_P^n}^\perp$.

4. Finally the encoding $\llbracket P \rrbracket$ is defined as follows:

$$\llbracket P \rrbracket \stackrel{\text{def}}{=} (x_\top)C_X^\top$$

The encoding $\llbracket - \rrbracket$ so defined makes full use of the dynamic binding mechanism. It is very much like the counter example defined in the proof of Theorem 3.

To understand the encoding, the reader is advised to work out the encoding of the process $P \equiv \mu X.(a+b|X)$. The equations derived from this process are

$$\begin{aligned} X &= Y \\ Y &= a + b | Y \end{aligned}$$

which can be simplified to a single equation

$$X = a + b | X \quad (1)$$

To improve readability let's write P_E for C_E^\top and Q_E for C_E^\perp . The constant definitions that correspond to (1) are the following:

$$\begin{aligned} P_X &= P_{a+b|X} \\ P_{a+b|X} &= \bar{x} + \tau.(y)(Q_a|y.P_{a+b|X}) \\ &\quad + \tau.(y)(Q_{b|X}|y.P_{a+b|X}) \\ Q_a &= \bar{y} + a.(x)P_0 \\ Q_{b|X} &= \bar{y} + \tau.(u)(Q_{b|X}^1|Q_{b|X}^2) \\ Q_{b|X}^1 &= \tau.(x)(P_b|x.Q_{b|X}^1) + u.Q_{b|X} \\ Q_{b|X}^2 &= \tau.(x)(P_X|x.Q_{b|X}^2) + \bar{u} \\ P_b &= \bar{x} + b.(x)P_0 \\ P_0 &= \bar{x} \end{aligned}$$

The proof of the correctness of the encoding can be found in Appendix B. \square

Theorem 1 and Theorem 4 imply that there are specifications in CCS that can not be implemented by CCS processes without the choice operators.

3 Interaction Hierarchy of Pi

In this section we apply the idea of the subbisimilarity to investigate the interactability of the variants of the π -Calculus. Most results we have established for CCS can be easily transplanted to the π -Calculus. We start with the core of the π -Calculus whose abstract grammar is as follows:

$$E ::= \mathbf{0} \mid a(x).E \mid \bar{a}x.E \mid E \mid E \mid (x)E \mid \mu X.E$$

We shall let π^- denote this calculus. The semantic rules are given below:

Prefix

$$\frac{}{a(x).E \xrightarrow{ay} E\{y/x\}} \quad \frac{}{\bar{a}x.E \xrightarrow{\bar{a}x} E}$$

Restriction

$$\frac{E \xrightarrow{\bar{a}x} E'}{(x)E \xrightarrow{\bar{a}(x)} E'} \quad \frac{E \xrightarrow{\lambda} E' \quad x \notin n(\lambda)}{(x)E \xrightarrow{\lambda} (x)E'}$$

Composition, Fixpoint

$$\frac{E \xrightarrow{\lambda} F'}{E \mid F \xrightarrow{\lambda} E' \mid F'} \quad \frac{E \xrightarrow{ax} E' \quad F \xrightarrow{\bar{a}x} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$$

$$\frac{E \xrightarrow{ax} E' \quad F \xrightarrow{\bar{a}(x)} F'}{E \mid F \xrightarrow{\tau} (x)(E' \mid F')} \quad \frac{E\{\mu X.E/X\} \xrightarrow{\lambda} E'}{\mu X.E \xrightarrow{\lambda} E'}$$

The semantics of the μ -operator is defined as in CCS. The variants with the choice, mixed choice and the separated choice are denoted respectively by π (or π^μ), π^m and π^s . We will write $\pi^!$ for the variant of π obtained by replacing the μ -operator by the replicator “!”. ”.

To compare the interactional powers of the variants of the π , we need to define the subbisimilarity relations for the π variants. Among all the equivalence relations proposed for the π -Calculus [21], the quasi open bisimilarity stands out as the bisimulation equivalence with the right distinguishing power [20, 7]. A quasi bisimulation is a family of relations $\{\mathcal{R}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ indexed by the set of the finite set of names (\subseteq_f being the finite subset relation). Intuitively $P \mathcal{R}^{\tilde{x}} Q$ means that P and Q are quasi open bisimilar under the assumption that the names \tilde{x} were local names that have been opened up by the bounded output actions. To make sure that the quasi open bisimilarity is closed under the prefix operation, it is required that each $\mathcal{R}^{\tilde{x}}$ is closed under the respectful substitutions. A substitution σ respects \tilde{x} if $(\forall x \in \tilde{x}.\sigma(x) = x) \wedge (\forall z \notin \tilde{x}.\sigma(z) \notin \tilde{x})$.

The following definition extends that of the quasi open bisimulations, in which $bn(\lambda)$ is the set of bound names in λ .

Definition 4. Suppose that π^1 and π^2 are two π variants. A subbisimilarity from π^1 to π^2 is a family $\{\mathcal{R}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ such that the following properties hold:

- (i) For each $\tilde{x} \subseteq_f \mathcal{N}$, $\mathcal{R}^{\tilde{x}}$ is a left surjective and is closed under the substitutions that respect \tilde{x} .
- (ii) The bisimulation property holds whenever $S_1 \mathcal{R}^{\tilde{x}} S_2$:

If $S_1 \xrightarrow{\lambda}_1 S'_1$ then $S_2 \xRightarrow{\tilde{\lambda}}_2 S'_2$ for some S'_2 such that $S'_1 \mathcal{R}^{\tilde{x}bn(\lambda)} S'_2$;

If $S_2 \xrightarrow{\lambda}_2 S'_2$ then $S_1 \xRightarrow{\tilde{\lambda}}_1 S'_1$ for some S'_1 such that $S'_1 \mathcal{R}^{\tilde{x}bn(\lambda)} S'_2$.

We say that π^1 is subbisimilar to π^2 , notation $\pi^1 \sqsubseteq^{pi} \pi^2$, if there is a subbisimilarity from the former to the latter. A subbisimilarity $\{\mathcal{R}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ satisfies the Computation Criterion if $\mathcal{R}^{\tilde{x}}$ satisfies the Computation Criterion for each $\tilde{x} \subseteq_f \mathcal{N}$. The notations \sqsubseteq^{pi} , $\sqsubseteq_{\downarrow}^{pi}$, $\sqsubseteq_{\downarrow}^{pi}$ are defined accordingly.

A subbisimilarity from a (synchronous) variant of π to itself is a special quasi open bisimulation. The largest of such quasi open bisimulation is the quasi open bisimilarity denoted by \approx .

The results obtained in the previous section can be redone in the π framework. In what follows we make use of the subbisimilarity to classify the interactability of the name passing interactions.

3.1 Choice and Recursion for Mobile Process

The relative expressiveness of the choice operators in the π -Calculus is the same as in CCS. The following theorems confirm.

Theorem 5. $\pi^- \sqsubseteq_{\downarrow}^{pi} \pi^m \sqsubseteq_{\downarrow}^{pi} \pi^s$.

Proof. A reiteration of the proof of Theorem 1 suffices. \square

If the Computation Criterion must be met, the picture remains the same as in CCS.

Theorem 6. $\pi^s \sqsubseteq_{\downarrow}^{pi} \pi^m \sqsubseteq_{\downarrow}^{pi} \pi$.

Proof. Both the proof of $\pi^s \sqsubseteq_{\downarrow}^{pi} \pi^m$ and the proof of $\pi^m \sqsubseteq_{\downarrow}^{pi} \pi$ can be copied from the proof of Theorem 2. \square

Next we take a look at the various recursion mechanisms in the setting of π -Calculus. For the sake of bookkeeping, we repeat the following well known fact [9]:

Fact 3. $\pi^m \sqsubseteq_{\downarrow}^{pi} \pi^1 \sqsubseteq_{\downarrow}^{pi} \pi^m$.

As it turns out, the relationship between π^1 and π is best seen via the corresponding variants using the dynamic fixpoint. In other words, we need to work with $\pi^{d\mu}$, the π

with the dynamic μ -operator, and π^{dm} , the π^m with the dynamic μ -operator. We will postulate that in $\pi^{d\mu}$ and π^{dm} , free names can get bound by both the localization operators and the input prefix binders. For instance after unfolding $\mu X.(\bar{x}z | y(z).(x)(X | \bar{z}z))$, one gets the process

$$\bar{x}z | y(z).(x)(\mu X.(\bar{x}z | y(z).(x)(X | \bar{z}z)) | \bar{z}z)$$

Here the free name z in $\mu X.(\bar{x}z | y(z).(x)(X | \bar{z}z))$ is bound by the input prefix operator $y(z)$ and the free name x is bound by the localization operator (x) . Clearly the dynamic fixpoint subsumes the static fixpoint.

Theorem 7. $\pi^m \sqsubseteq_{\downarrow}^{pi} \pi^{dm}$ and $\pi \sqsubseteq_{\downarrow}^{pi} \pi^{d\mu}$.

Proof. Using α -conversion, one may assume that all the bound names in a π process P are distinct and are different from all the free names of P . The interactional behaviors of such a P remain unchanged if the fixpoint operator becomes dynamic. Hence $\pi^m \sqsubseteq_{\downarrow}^{pi} \pi^{dm}$ and $\pi \sqsubseteq_{\downarrow}^{pi} \pi^{d\mu}$. \square

The above theorem would not be interesting if the existences of the reverses of the two subbisimilarities are unknown. The next theorem reveals that π^m and π^{dm} are completely equivalent.

Theorem 8. $\pi^{dm} \sqsubseteq_{\downarrow}^{pi} \pi^m$.

Proof. Now suppose P is in π^{dm} . Let $\mu X_1.E_1$ be a subexpression of P that does not contain any μ -operator inside it. We can translate the expression $\mu X_1.E_1$ into the π^m expression

$$(m_X)(\overline{m_X}(c).\bar{c}x_1 \dots x_n.\mathbf{0} | \mu X.m_X(c).c(x_1 \dots x_n).E\zeta)$$

where

- m_X, c are fresh names;
- x_1, x_2, \dots, x_n are the free names in E_1 ;
- $c(x_1 \dots x_n).E\zeta$ is $c(x_1).c(x_2). \dots .c(x_n).E\zeta$;
- $\bar{c}x_1 \dots x_n.\mathbf{0}$ is $\bar{c}x_1.\bar{c}x_2. \dots .\bar{c}x_n.\mathbf{0}$;
- ζ is $[(\overline{m_X}(c).\bar{c}x_1 \dots x_n.\mathbf{0}) | X/X]$, a dynamic substitution of variable.

The intuition about the translation is that every time the fixpoint is unfolded, the free names of the expression E are updated with the local information. So there is actually no name capture when unfolding the fixpoint. By proceeding from the inside to outside, we can do the transformation to all the subexpressions (of P) in the fixpoint form. The resulting π^m process is then the encoding $\llbracket P \rrbracket$ of P . Notice that for this encoding to work the guarded choice condition is crucial.

Formally the encoding from π^{dm} process expressions to π^m process expressions is defined as follows:

$$\begin{aligned}
\llbracket \mathbf{0} \rrbracket &\stackrel{\text{def}}{=} \mathbf{0} \\
\llbracket X \rrbracket &\stackrel{\text{def}}{=} X \\
\llbracket (x)E \rrbracket &\stackrel{\text{def}}{=} (x)\llbracket E \rrbracket \\
\llbracket \sum_{i \in I} \lambda_i.E_i \rrbracket &\stackrel{\text{def}}{=} \sum_{i \in I} \lambda_i.\llbracket E_i \rrbracket \\
\llbracket E_1 \mid E_2 \rrbracket &\stackrel{\text{def}}{=} \llbracket E_1 \rrbracket \mid \llbracket E_2 \rrbracket \\
\llbracket \mu X.E \rrbracket &\stackrel{\text{def}}{=} (m_X)(\overline{m_X}(c).\overline{c}\vec{x} \\
&\quad \mid \mu X.m_X(c).c(\vec{x}).\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid X/X])
\end{aligned}$$

where $m_X, c, \vec{x} = x_1, \dots, x_n, \varsigma$ are defined as above. The π^{dm} process P is encoded by $\llbracket P \rrbracket$. Let's take a look at the operational aspect of the encoding. First of all notice that in π^m one has that

$$\llbracket \mu X.E \rrbracket \Longrightarrow (m_X)\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$$

and

$$\llbracket \mu X.E \rrbracket \approx (m_X)\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$$

where $\mu X \equiv \mu X.m_X(c).c(\vec{x}).\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid X/X]$. It might occur to the reader whether something would go wrong if $\overline{m_X}(c).\overline{c}\vec{x}$ communicate with another copy of μX rather than the neighboring μX . The fact that such a communication would do no harm is stated in Lemma 3 established right after this proof. It follows immediately from Lemma 3 that $\llbracket \mu X.E \rrbracket \approx \llbracket E \rrbracket[\llbracket \mu X.E \rrbracket/X]$.

Now define the relation \mathcal{R} as follows:

$$\mathcal{R} \stackrel{\text{def}}{=} \{(P, \llbracket P \rrbracket) \mid P \text{ is a } \pi^{dm} \text{ process}\}$$

Using Lemma 4 it is routine to show that $\mathcal{R} \approx$ is a subbisimilarity from π^{dm} to π^m . Now suppose $P_1 \mathcal{R} \llbracket P_1 \rrbracket \approx Q_1$. If $P_1 \xrightarrow{\lambda} P_2$ then $\llbracket P_1 \rrbracket \xrightarrow{\lambda} P' \approx \llbracket P_2 \rrbracket$. By definition some P'' exists such that $Q_1 \xrightarrow{\hat{\lambda}} Q'_1 \approx P'$. Therefore $\llbracket P_2 \rrbracket \mathcal{R} \approx Q'_1$. In the other direction if $Q_1 \xrightarrow{\lambda} Q'_1$ then P' exists such that $\llbracket P_1 \rrbracket \xrightarrow{\hat{\lambda}} P' \approx Q'_1$. By (ii) of Lemma 4, some P_2 exists such that $P_1 \xrightarrow{\hat{\lambda}} P_2$ for some P_2 such that $\llbracket P_2 \rrbracket \approx P'$. Hence $P'_1 \mathcal{R} \approx Q'_1$. \square

Lemma 3. Suppose $P \equiv \mu X.E$ is in π^{dm} . Let μX abbreviate the process $\mu X.m_X(c).c(\vec{x}).\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid X/X]$. Then $(m_X)\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X] \approx \llbracket E \rrbracket[\llbracket \mu X.E \rrbracket/X]$ in π^m .

Proof. Let E be an expression in π^m . It suffices to show that $(m_X)E[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X] \approx E[\llbracket \mu X.E \rrbracket/X]$. We prove that $(m_X)E[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$ and $E[\llbracket \mu X.E \rrbracket/X]$

bisimulate. If an action is caused by E then the bisimulation is trivial. If an action of $E[\llbracket \mu X.E \rrbracket/X]$ is caused by

$$\llbracket \mu X.E \rrbracket \xrightarrow{\tau} (c)(\overline{c}\vec{x} \mid c(\vec{x}).\llbracket E \rrbracket[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X])$$

In the same place, $(m_X)E[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$ can do the same action. If an action of $(m_X)E[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$ is caused by a communication of a $\overline{m_X}(c).\overline{c}\vec{x}$ and a μX , then structurally speaking the communication is the same as the communication between $\overline{m_X}(c).\overline{c}\vec{x}$ and its neighboring μX . So this action of $(m_X)E[\overline{m_X}(c).\overline{c}\vec{x} \mid \mu X/X]$ is simulated by the same action of $E[\llbracket \mu X.E \rrbracket/X]$. \square

To state the following lemma, we introduce a notation:

$P \xrightarrow{\hat{\lambda}} P'$ is either $P \xrightarrow{\lambda} P'$ or $P \equiv P'$ when $\lambda = \tau$. In other words $P \xrightarrow{\hat{\tau}} P'$ could be either $P \xrightarrow{\tau} P'$ or $P \equiv P'$.

Lemma 4. The following properties hold:

- (i) If $P_1 \xrightarrow{\lambda} P_2$ then $\llbracket P_1 \rrbracket \xrightarrow{\hat{\lambda}} \approx \llbracket P_2 \rrbracket$.
- (ii) If $\llbracket P_1 \rrbracket \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_n} P'$ then $P_1 \xrightarrow{\hat{\lambda}_1} \dots \xrightarrow{\hat{\lambda}_n} P_2$ for some P_2 such that $\llbracket P_2 \rrbracket \approx P'$.

Proof. (i) The proof is by structural induction. If P is not a fixpoint then use the induction hypothesis. Otherwise use Lemma 3. (ii) If $\lambda_1 \dots \lambda_n$ is a sequence of internal communications between either some m_X or some c , then $\llbracket P_1 \rrbracket \approx P'$. If $\lambda_{i_1}, \dots, \lambda_{i_n}$ are not actions between either some m_X or some c and all the rest actions are communications between either some m_X or some c , then $P_1 \xrightarrow{\lambda_{i_1}} \dots \xrightarrow{\lambda_{i_m}} P_2$ such that $\llbracket P_2 \rrbracket \approx P'$. \square

Our efforts spent in the proof of Theorem 4 pays off by the following result.

Theorem 9. $\pi^{d\mu} \sqsubseteq^{pi} \pi^{dm}$ and $\pi^{d\mu} \not\sqsubseteq_{\downarrow}^{pi} \pi^{dm}$.

Proof. Using the technique developed in the proof of Theorem 4, one can prove that $\pi^{d\mu} \sqsubseteq^{pi} \pi^{dm}$. The counter example in the proof of Theorem 2 can be reused here to show that $\pi^{d\mu} \not\sqsubseteq_{\downarrow}^{pi} \pi^{dm}$. \square

Using Theorem 7, Theorem 8 and Theorem 9 a characterization of the relationships between π^m and π in terms of subbisimilarity is achieved.

Theorem 10. $\pi^{d\mu} \sqsubseteq^{pi} \pi \sqsubseteq^{pi} \pi^m$.

The results of this subsection about the mobile processes can be summarized as follows: All forms of choice are interactively equivalent; and all mechanisms of introducing the infinite behaviors are interactively equivalent.

The only unrevealed relationship is to do with $\pi^{d\mu}$ and π . Is there an encoding from $\pi^{d\mu}$ to π that satisfies the Computation Criterion? This is the second open problem of the paper left for further investigation.

Problem 2. $\pi^{d\mu} \sqsubseteq_{\downarrow}^{pi} \pi$?

3.2 Independence Result

The name passing communication mechanism of the π -Calculus facilitates the internalization of the conditionals useful in programming. It is interesting to see how the conditionals enhance the interactability of the π . The syntaxes of the match and the mismatch operators are $[x=y]E$ and $[x\neq y]E$ respectively. The semantics is defined by the following rules:

$$\frac{E \xrightarrow{\lambda} E'}{[x=y]E \xrightarrow{\lambda} E'} \quad \frac{E \xrightarrow{\lambda} E'}{[x\neq y]E \xrightarrow{\lambda} E'}$$

We use the notations $\pi^=$, π^\neq and π^c to stand for the π with respectively the match operator $=$, the mismatch operator \neq and both match and mismatch operators. It is clear that $\pi \sqsubseteq^{pi} \pi^=$ (π^\neq) $\sqsubseteq^{pi} \pi^c$. More about the interactability of the conditionals are given in the next theorem.

Theorem 11. *The following negative results hold:*

- (i) $\pi^= \not\sqsubseteq^{pi} \pi$ and $\pi^\neq \not\sqsubseteq^{pi} \pi^\neq$.
- (ii) $\pi^\neq \not\sqsubseteq^{pi} \pi$ and $\pi^\neq \not\sqsubseteq^{pi} \pi^=$.
- (iii) $\pi^c \not\sqsubseteq^{pi} \pi^=$ and $\pi^c \not\sqsubseteq^{pi} \pi^\neq$.

Proof. (i) Consider the process $a(x).[x=y]\bar{b}b$. Suppose that $a(x).[x=y]\bar{b}b \mathcal{R} P$ for a process P of π and a subbisimilarity \mathcal{R} from $\pi^=$ to π . The two consecutive observable actions $a(x).[x=y]\bar{b}b \xrightarrow{ay} [y=y]\bar{b}b \xrightarrow{\bar{b}b} \mathbf{0}$ must be simulated by P in the manner $P \Longrightarrow P_1 \xrightarrow{ay} P_2 \Longrightarrow P_3 \xrightarrow{\bar{b}b} P_4 \Longrightarrow \approx \mathbf{0}$. We claim that for a fresh z some P'_2, P'_3, P'_4 exist such that

$$P_1 \xrightarrow{az} P'_2 \Longrightarrow P'_3 \xrightarrow{\bar{b}b} P'_4 \quad (2)$$

This fact can be proved by induction. The tricky part is to do with the prefix operator. Suppose $P_1 \equiv a(x).P'_1$. Then clearly $P'_1\{y/x\} \equiv P_2$ and $P'_1\{z/x\} \equiv P'_2$. It should be clear that $P_2 \approx \bar{b}b$. This equivalence implies that the internal computations in $P_2 \Longrightarrow P_3$ must be via local names. Since P_2 does not contain any match operations, the substitution $\{y/x\}$ does not enable any internal actions in P'_1 .

Therefore $P'_1 \Longrightarrow \xrightarrow{\bar{b}b}$, from which (2) follows. But (2) contradicts the fact that $a(x).[x=y]\bar{b}b \xrightarrow{az} [z=y]\bar{b}b$. Hence $\pi^= \not\sqsubseteq^{pi} \pi$. The argument for $\pi^\neq \not\sqsubseteq^{pi} \pi^\neq$ is similar.

(ii) Suppose the process $a(x).[x\neq y]\bar{b}b$ were equivalent to a process P of π ($\pi^=$). For a fresh z the two consecutive actions $a(x).[x\neq y]\bar{b}b \xrightarrow{az} [x\neq y]\bar{b}b \xrightarrow{\bar{b}b} \mathbf{0}$ must be simulated by $P \xrightarrow{az} \xrightarrow{\bar{b}b}$. But then $P \xrightarrow{ay} \xrightarrow{\bar{b}b}$, which can not be simulated by $a(x).[x\neq y]\bar{b}b$ of course.

(iii) The proofs are similar. \square

An important application of the subbisimilarity approach is to formally establish the independence of the operators.

Let π_1 be a variant of π and π_1^{-op} be obtained from π_1 by removing the operator op of π_1 and the associated operational rules. We say that the operator op is *independent* to the other operators of π_1 if $\pi_1 \not\sqsubseteq^{pi} \pi_1^{-op}$.

Theorem 12. *All the operators of π^c are independent.*

Proof. Theorem 11 says that the match, and the mismatch as well, is independent from the rest of the operators of π^c . The prefix operator and the replicator are easily seen to be independent from the other operators. The proof of Theorem 5 shows that the choice operator is independent. The localization operator is independent because the restricted output actions are interactively different from the input actions and the free output actions. \square

4 Asynchronous Subbisimilarity

Asynchronous communications are those in which the parties that send messages immediately go ahead without waiting for any acknowledgements from the receiving parties. In the setting of the π -Calculus the asynchronous output can be defined by the following semantic rules

$$\frac{}{\bar{a}x.P \xrightarrow{\tau} \bar{a}x \mid P} \quad \frac{}{\bar{a}x \xrightarrow{\bar{a}x} \mathbf{0}}$$

It is apparent that $\bar{a}x.P$ is bisimilar to $\bar{a}x \mid P$ under this semantics. This fact tells us that if we want to study the asynchronous communications in the setting of the π -Calculus, we might as well focus on the following syntax:

$$P ::= \mathbf{0} \mid a(x).P \mid \bar{a}x \mid P \mid P \mid (x)P \mid !P$$

This is the well known *asynchronous* π , often denoted by π^a . The variants of π^a include the followings:

- π^i is the π^a extended with the input choice operation $\sum_{i \in I} a_i(x).P_i$, where I is finite;
- $\pi^a_{=}$ is the π^a extended with the equality conditional prefix $\varphi a(x).P$; $\pi^i_{=}$ is the π^i extended with the equality conditional input choice $\sum_{i \in I} \varphi_i a_i(x).P_i$;
- π^a_{\neq} is the π^a extended with the inequality conditional prefix $\varphi a(x).P$; π^i_{\neq} is the π^i extended the inequality conditional input choice $\sum_{i \in I} \varphi_i a_i(x).P_i$;
- π^a_c is the π^a extended with the conditional prefix $\varphi a(x).P$; π^i_c is the π^i extended with the conditional input choice $\sum_{i \in I} \varphi_i a_i(x).P_i$.

In the asynchronous calculi, $\bar{a}x$ is a message that has been sent by some process but has not yet been received by any process. It does not really make sense to place any condition right in front of $\bar{a}x$. The algebraic theory of π^a has been studied by several researchers [2, 10, 14, 16, 15].

These studies have reached in a consensus on how the asynchronous processes should be observed.

An asynchronous observer differs in observing power from a synchronous one in that the former can not really see the input actions performed by the observees. Consequently the subbisimilarity relations between two asynchronous π variants do not explicitly bisimulate the input actions. The abilities to perform input actions are compared by the effects they exert on the neighboring processes. The following definition of the asynchronous subbisimilarity borrows the ideas from [10, 1].

Definition 5. Suppose that π_1 and π_2 are two π variants. An asynchronous subbisimilarity from π_1 to π_2 is a family $\{\mathcal{R}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ such that the following properties hold:

- (i) For each $\tilde{x} \subseteq_f \mathcal{N}$, $\mathcal{R}^{\tilde{x}}$ is left surjective and is closed under the substitutions that respect \tilde{x} ;
- (ii) For each $\tilde{x} \subseteq_f \mathcal{N}$, $\mathcal{R}^{\tilde{x}}$ is closed under composition;
- (iii) The bisimulation property holds whenever $S_1 \mathcal{R}^{\tilde{x}} S_2$:

If $S_1 \xrightarrow{\lambda}_1 S'_1$ for some $\lambda \in \{\bar{a}x, \bar{a}(x) \mid a, x \in \mathcal{N}\} \cup \{\tau\}$, then $S_2 \xrightarrow{\hat{\lambda}}_2 S'_2$ for some S'_2 such that $S'_1 \mathcal{R}^{\tilde{x}bn(\lambda)} S'_2$;

If $S_2 \xrightarrow{\lambda}_2 S'_2$ for some $\lambda \in \{\bar{a}x, \bar{a}(x) \mid a, x \in \mathcal{N}\} \cup \{\tau\}$, then $S_1 \xrightarrow{\hat{\lambda}}_1 S'_1$ for some S'_1 such that $S'_1 \mathcal{R}^{\tilde{x}bn(\lambda)} S'_2$.

We say that π_1 is asynchronously subbisimilar to π_2 , notation $\pi_1 \sqsubseteq^{asy} \pi_2$, if there is some asynchronous subbisimilarity from the former to the latter. The notations \sqsubseteq^{asy} , $\sqsubseteq_{\downarrow}^{asy}$, $\sqsubseteq_{\downarrow}^{asy}$ are defined accordingly.

There are several alternatives and simplifications to the above definition, see [10, 1] for more details. But Definition 5 is formulated to reconcile the asynchrony with the synchrony. The following proposition explains.

Proposition 1. Suppose π^1 and π^2 are two synchronous variants of π . Then (i) $\pi^1 \sqsubseteq^{asy} \pi^2$ if and only if $\pi^1 \sqsubseteq^{pi} \pi^2$; and (ii) $\pi^1 \sqsubseteq_{\downarrow}^{asy} \pi^2$ if and only if $\pi^1 \sqsubseteq_{\downarrow}^{pi} \pi^2$.

Proof. Suppose π^1 and π^2 are two synchronous variants of the π -Calculus. Let $\mathcal{R} = \{\mathcal{R}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ be a subbisimilarity from π^1 to π^2 in the sense of Definition 4. For each $\tilde{x} \subseteq_f \mathcal{N}$, we define an infinite sequence of relations from π^1 to π^2 by the following induction:

- $\sqsubseteq_0^{\tilde{x}} \stackrel{\text{def}}{=} \mathcal{R}^{\tilde{x}}$;
- $\sqsubseteq_{n+1}^{\tilde{x}} \stackrel{\text{def}}{=} \left\{ \begin{array}{l} ((z)S_1, (z)S_2), \\ (S_1 | S'_1, S_2 | S'_2) \end{array} \left| \begin{array}{l} S_1 \sqsubseteq_n^{\tilde{x}} S_2 \\ S'_1 \sqsubseteq_n^{\tilde{x}} S'_2 \\ z \notin \tilde{x} \end{array} \right. \right\}$.

Now let $\sqsubseteq_{\infty}^{\tilde{x}}$ be $\bigcup_{n \in \omega} \sqsubseteq_n^{\tilde{x}}$ and \sqsubseteq^{∞} be $\{\sqsubseteq_{\infty}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$. It is a standard exercise to show that \sqsubseteq^{∞} satisfies the bisimulation

property of Definition 5. By construction it is closed under composition. Hence $\pi^1 \sqsubseteq^{asy} \pi^2$. If \mathcal{R} satisfies the Computation Criterion, then it follows from the bisimulation property that \sqsubseteq^{∞} also satisfies the Computation Criterion.

Let $\mathcal{A} = \{\mathcal{A}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ be an asynchronous subbisimilarity from π^1 to π^2 . Suppose $P \mathcal{A}^{\tilde{x}} Q \xrightarrow{\bar{a}z} Q'$. For names b, c that appear neither in P nor Q , there must be some process B in π^2 such that $\bar{a}z.\bar{b}c \mathcal{A}^{\tilde{x}} B$. It should be not difficult to see that $B \xrightarrow{\bar{a}z} \xrightarrow{\bar{b}c} \sim \mathbf{0}$. Consequently

$$P | \bar{a}z.\bar{b}c \mathcal{A}^{\tilde{x}} Q | B \xrightarrow{\tau} \xrightarrow{\bar{b}c} \sim Q' | \mathbf{0}$$

It follows from the bisimulation property that $P \xrightarrow{\bar{a}z} P'$ and $P' | \mathbf{0} \mathcal{A}^{\tilde{x}} Q' | \mathbf{0}$. We conclude that $\{\sim \mathcal{A}^{\tilde{x}} \sim\}_{\tilde{x} \subseteq_f \mathcal{N}}$ is a subbisimilarity from π^1 to π^2 in the sense of Definition 4. Obviously $\{\sim \mathcal{A}^{\tilde{x}} \sim\}_{\tilde{x} \subseteq_f \mathcal{N}}$ satisfies the Computation Criterion if and only if $\{\mathcal{A}^{\tilde{x}}\}_{\tilde{x} \subseteq_f \mathcal{N}}$ satisfies the Computation Criterion. \square

Proposition 1 asserts that the asynchronous subbisimilarity provides a uniform criterion for comparing the interactability of both the asynchronous π as well as the synchronous π . Thanks to Proposition 1 the asynchronous subbisimilarity relationships imply the full abstractions that relate the asynchronous bisimilarities to the (synchronous) bisimilarities. In the rest of the section we use the asynchronous subbisimilarity to characterize the interactability. The next theorem says that the conditionals also add substantial expressive power in the asynchronous setting.

Theorem 13. The following relationships hold:

- (i) $\pi^a \sqsubseteq^{asy} \pi_{\neq}^a \sqsubseteq^{asy} \pi_c^a$; $\pi^a \sqsubseteq^{asy} \pi_{\neq}^a \sqsubseteq^{asy} \pi_c^a$;
- (ii) $\pi^i \sqsubseteq^{asy} \pi_{\neq}^i \sqsubseteq^{asy} \pi_c^i$; $\pi^i \sqsubseteq^{asy} \pi_{\neq}^i \sqsubseteq^{asy} \pi_c^i$;

Proof. The inclusion maps are all asynchronous subbisimilarities. The processes $a(x).(c)(\bar{c}c | [x=y]c(z).\bar{b}b)$ and $a(x).(c)(\bar{c}c | [x \neq y]c(z).\bar{b}b)$ are in π_{\neq}^a and π_c^a respectively. They can be used to show that there are no asynchronous subbisimilarities in the opposite directions. Suppose there were an asynchronous subbisimilarity \mathcal{R} from π_{\neq}^a to π^a and that $a(x).(c)(\bar{c}c | [x=y]c(z).\bar{b}b) \mathcal{R} P$ for some P in π^a . Suppose further that $\bar{a}y \mathcal{R} A$ and $\bar{a}z \mathcal{R} B$. Since \mathcal{R} is closed under composition, one must have

$$\bar{a}y | a(x).(c)(\bar{c}c | [x=y]c(z).\bar{b}b) \mathcal{R} A | P \quad (3)$$

$$\bar{a}z | a(x).(c)(\bar{c}c | [x=y]c(z).\bar{b}b) \mathcal{R} B | P \quad (4)$$

By definition the only observable action A can do is $\bar{a}y$. For the same reason B can not do any observable actions apart from $\bar{a}z$. It follows from (3) and (4) that

$$P' \xleftarrow{\bar{a}z} \longleftarrow P \xrightarrow{\bar{a}y} \xrightarrow{\bar{b}b} \quad (5)$$

for some P' that can not do $\bar{b}b$. As in the proof of Theorem 11, the two sequences of actions in (5) contradict. \square

The relationship between π^a and π^i has been studied by Nestmann and Pierce in [14]. The remarkable result of [14] is that these two calculi are interactively equivalent.

Fact 4 (Nestmann, Pierce [14]). $\pi^i \sqsubseteq^{asy} \pi^a$.

The technique used to prove the above fact can be easily adapted to a proof of the following fact.

Fact 5. $\pi^i \sqsubseteq^{asy} \pi^a; \pi^i \sqsubseteq^{asy} \pi^a; \pi^i \sqsubseteq^{asy} \pi^a$.

However the above equivalences fail if the computational factor is taken into account. It is conjectured in [14] that there is no termination preserving fully abstract encoding from π^i to π^a . The next theorem confirms the conjecture.

Theorem 14. *The following relationships are valid:*
 $\pi^a \sqsubseteq_{\downarrow}^{asy} \pi^i; \pi^a \sqsubseteq_{\downarrow}^{asy} \pi^i; \pi^a \sqsubseteq_{\downarrow}^{asy} \pi^i; \pi^a \sqsubseteq_{\downarrow}^{asy} \pi^i$.

Proof. If $\pi^i \sqsubseteq_{\downarrow}^{asy} \pi^a$ were true, there would be an asynchronous subbisimilarity \mathcal{R} from π^i to π^a that satisfies the Computation Criterion. Then there would be C in π^a such that $a(z)+b(z) \mathcal{R} C$. Since $a(z)+b(z)$ is terminating, there must be some C' such that $C \Longrightarrow C'$ and the next action of C' can not be τ . By definition $C \Longrightarrow C'$ must be matched up by $a(z)+b(z)$. But the latter can not do any internal actions. Therefore $a(z)+b(z) \mathcal{R} C'$. So we may as well assume that the next action of C can not be τ in the first place. Similarly A and B exist such that $\bar{a}a \mathcal{R} A$ and $\bar{b}b \mathcal{R} B$. Again we may assume that the next action of A (B) can only be $\bar{a}a$ ($\bar{b}b$). Since \mathcal{R} is closed under composition, one would have that $\bar{a}a | \bar{b}b | (a(z)+b(z)) \mathcal{R} A | B | C$, from which it follows that C could do immediately an input action on a and an input action on b . But then C could do an input action on a followed by an input action on b , which would lead to a contradiction. \square

Nestmann looked at the encodings from π^s to π^a [15]. The adequacy of the encodings is established for some equivalences considerably weaker than the bisimilarity. As a matter of fact there is little room for improvement.

Theorem 15. $\pi^a \sqsubseteq^{asy} \pi^-; \pi^i \sqsubseteq^{asy} \pi^s$.

Proof. Suppose A is in π^a . There is no asynchronous subbisimilarity \mathcal{R} from π^- to π^a to which $(\bar{a}x.\bar{b}y, A)$ belongs. This can be argued as follows: Suppose there were an asynchronous subbisimilarity \mathcal{R} such that $(\bar{a}x.\bar{b}y, A) \in \mathcal{R}$. Now the two consecutive actions $\bar{a}x.\bar{b}y \xrightarrow{\bar{a}x} \bar{b}y \xrightarrow{\bar{b}y} \mathbf{0}$ had to be simulated by $A \Longrightarrow A_1 \xrightarrow{\bar{a}x} A_2 \Longrightarrow A_3 \xrightarrow{\bar{b}y} A'$. By the definition of asynchrony, the action $A_1 \xrightarrow{\bar{a}x} A_2$ can be delayed until all the internal actions from A_2 to A_3 have been executed. In other words, there is an A'_2 such that $A_1 \Longrightarrow A'_2 \xrightarrow{\bar{a}x} A_3$. Using the asynchronous property

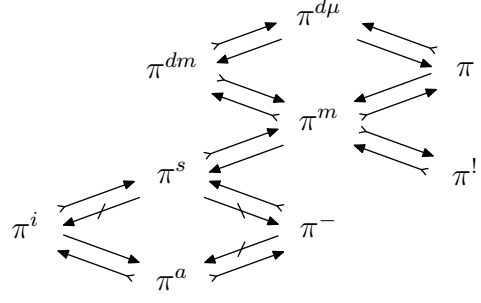


Figure 3. Subbisimilarities of Pi

again, one should have $A'_2 \xrightarrow{\bar{b}y} \bar{a}x$. This action sequence is impossible for $\bar{a}x.\bar{b}y$ to simulate.

The argument for $\pi^i \sqsubseteq^{asy} \pi^s$ is the same. \square

Theorem 15 can be interpreted as saying that the asynchronous variants of π are strictly less expressive than the synchronous counterparts from the viewpoint of interaction.

Finally we state some results that tighten up most of the loose ends.

Theorem 16. $\pi^- \not\sqsubseteq^{asy} \pi^i; \pi^s \not\sqsubseteq^{asy} \pi^-; \pi^i \not\sqsubseteq_{\downarrow}^{asy} \pi^-$.

Proof. The arguments used in the proof of Theorem 15 are good for $\pi^- \not\sqsubseteq^{asy} \pi^i$. The arguments for Theorem 1 can be modified to prove $\pi^s \not\sqsubseteq^{asy} \pi^-$ and $\pi^i \not\sqsubseteq_{\downarrow}^{asy} \pi^-$. \square

5 Final Remark

The paper clarifies some issues on the relative expressive power of the variants of the π -Calculus. The results of this paper about the π -calculi are summarized in Figure 3. In the diagram, the tailed arrow \succrightarrow is $\sqsubseteq_{\downarrow}^{asy}$; the plain arrow \rightarrow represents \sqsubseteq^{asy} ; and \leftrightarrow indicates \sqsubseteq^{asy} . The existence of the plain arrow \rightarrow from π^1 to π^2 should also be understood to rule out the existence of the tailed arrow \succrightarrow from π^1 to π^2 , except in the case of $\pi^{d\mu} \rightarrow \pi$. The question of whether $\pi^{d\mu} \succrightarrow \pi$ holds is yet to be answered. The picture of Figure 3 is not compatible with that of Figure 1. For example there is an arrow from π^s to π^a in Figure 1. In Figure 3 there is definitely not a plain arrow from π^s to π^a for otherwise it would contradict to $\pi^s \not\rightarrow \pi^-$. As a matter of fact in Figure 1 there is an arrow from π^s to π^- by composing two arrows.

The approach of using subbisimilarity to compare the expressiveness of process calculi as advocated in this paper is an attempt towards a general theory of interactability. An avenue that must be further pursued is about how to compare process calculi with different sets of actions. Suppose $\mathcal{T}_1, \mathcal{T}_2$ are the sets of the actions of \mathcal{L}_1 and \mathcal{L}_2 respectively. A subbisimilarity \mathcal{R} from \mathcal{L}_1 to \mathcal{L}_2 must be accompanied by

an injective map $\iota : \mathcal{T}_1 \rightarrow \mathcal{T}_2$ such that $\iota(\tau) = \tau$. The role of the injective map is to identify the actions of \mathcal{L}_1 to the associated actions of \mathcal{L}_2 so that the bisimulation property can be defined in a manner that looks like the following:

If $S_2 \mathcal{R}^{-1} S_1 \xrightarrow{\lambda} S'_1$ then $S_2 \xrightarrow{\widehat{\lambda}'} S'_2 \mathcal{R}^{-1} S'_1$ for some λ', S'_2 such that $\iota(\lambda) = \lambda'$;

If $S_1 \mathcal{R} S_2 \xrightarrow{\lambda'} S'_2$ then $S_1 \xrightarrow{\widehat{\lambda}} S'_1 \mathcal{R} S'_2$ for some λ, S'_1 such that $\iota(\lambda) = \lambda'$.

A simple example of two calculi with the different sets of actions is given by πI [18] and π . First notice that the inclusion map from πI to π is not a subbisimilarity. The process $a(x).(x(z) | \overline{y}(z))$ in πI is interactively different from the same process in π . Constructing a subbisimilarity from π to πI is even more problematic since the free output actions of the former are not available in the latter. The situation is similar if the relationship between π and the higher order π [22, 19] is to be studied in terms of subbisimilarity. A positive subbisimilarity relationship between two quite different calculi is offered in [8], where a variant of the Ambient Calculus [5], called FA, is proposed and studied. It is shown that there is an injective map from the set of the actions of the π -calculus to the set of the actions of FA that supports a subbisimilarity from π to FA, implying that π is actually a subcalculus of FA even from the viewpoint of interactability.

References

- [1] R. Amadio, I. Castellani and D. Sangiorgi. On Bisimulations for the Asynchronous π -Calculus. U. Montanari, V. Sassone (eds.): *CONCUR 1996*, Lecture Notes in Computer Science 1119, 147-162, 1996.
- [2] G. Boudol. Asynchrony and the π -Calculus. Technical Report RR-1702, INRIA Sophia-Antipolis, 1992.
- [3] N. Busi, M. Gabbriellini and G. Zavattaro. Replication vs Recursive Definitions in Channel Based Calculi. *ICALP'03*, Lecture Notes in Computer Science 2719: 133-144, 2003.
- [4] N. Busi, M. Gabbriellini and G. Zavattaro. Comparing Recursion, Replication and Iteration in Process Calculi. *ICALP'04*, Lecture Notes in Computer Science 3142: 307-319, 2004.
- [5] L. Cardelli and A. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240: 177-213, 2000.
- [6] A. Finkel and Ph. Schnoebelen. Well-Structured Transition System Everywhere. *Theoretical Computer Science*, 256: 63-92, 2001.
- [7] Y. Fu. On Quasi Open Bisimulation. *Theoretical Computer Science*, 338: 96-126, 2005.
- [8] Y. Fu. Fair Ambients. to appear in *Acta Informatica*.
- [9] P. Giambiagi, G. Schneider and F. Valencia. On the Expressiveness of Infinite Behavior and Name Scoping in Process Calculi. *FOSSACS 2004*, Lecture Notes in Computer Science 2987: 226-240, 2004.
- [10] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communications. M. Tokoro, O. Nierstrasz, P. Wegner and A. Yonezawa (eds.), *ECOOP 1991*, Geneva, Switzerland, Lecture Notes in Computer Science 512: 133-147, 1991.
- [11] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- [12] R. Milner. Elements of Interaction. *Communication of ACM*: 78-89, 1993.
- [13] R. Milner, J. Parrow and D. Walker. A Calculus of Mobile Processes. *Information and Computation*, 100: 1-40 (Part I), 41-77 (Part II), Academic Press, 1992.
- [14] U. Nestmann and B. Pierce. Decoding Choice Encodings. In: U. Montanari, V. Sassone (eds.): *CONCUR 1996*, Lecture Notes in Computer Science 1119: 179-194, 1996.
- [15] U. Nestmann. What is a Good Encoding of Guarded Choices? *Information and computation*, 156(1-21-2): 287-319, 2000.
- [16] C. Palamidessi. Comparing the Expressive Power of the Synchronous and the Asynchronous π -Calculus. *Mathematical Structures in Computer Science* 13(5): 685-719, 2003.
- [17] D. Park. Concurrency and Automata on Infinite Sequences. Lecture Notes in Computer Science 104: 167-183, Springer, 1981.
- [18] D. Sangiorgi. π -Calculus, Internal Mobility and Agent-Passing Calculi. *Theoretical Computer Science*, 167: 235-274, 1996.
- [19] D. Sangiorgi. Bisimulation for Higher Order Process Calculi. *Information and Computation*, 131(2): 141-178, 1996.
- [20] D. Sangiorgi and D. Walker. On Barbed Equivalence in π -Calculus, In: K. Larsen and M. Nielsen (eds.): *CONCUR 2001*, 292-304, Lecture Notes in Computer Science 2154, 2001.
- [21] D. Sangiorgi and D. Walker. *The Pi Calculus—A Theory of Mobile Processes*, CUP, 2001.

[22] B. Thomsen. A Theory of Higher Order Communicating Systems. *Information and Computation* 116: 38-57, 1995.

A The Proof of Theorem 3

The proof of Theorem 3 could be given by a head-on approach. We will however make use of the materials developed by Busi, Gabbriellini and Zavattaro. The following definitions and propositions are from [3], in which \equiv is the structural congruence defined in [3].

Definition 6. A well-quasi-ordering (wqo) is a quasi-ordering \leq over a set X such that, for any infinite sequence x_0, x_1, x_2, \dots in X , there exist indexes $i < j$ such that $x_i \leq x_j$.

Definition 7. A well-structured transition system with strong compatibility is a transition system $TS = (S, \rightarrow)$, equipped with a quasi-ordering \leq on S , such that the two following conditions hold:

1. **well-quasi-ordering:** \leq is a wqo over S , and
2. **strong compatibility:** \leq is (upward) compatible with \rightarrow , i.e., for all $s_1 \leq t_1$ and all transitions $s_1 \rightarrow s_2$, there exists a state t_2 such that $t_1 \rightarrow t_2$ and $s_2 \leq t_2$.

Definition 8. Let $P \in \text{CCS}^!$. With $\text{Deriv}(P)$ we denote the set of processes reachable from P with a sequence of τ actions.

$$\text{Deriv}(P) = \{Q \mid P \xrightarrow{\tau}^* Q\}$$

Proposition 2. Let $P, Q \in \text{CCS}^!$. If $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \equiv Q'$.

Definition 9. Let $P, Q \in \text{CCS}^!$. We write $P \preceq Q$ if and only if there exist $n, x_1, \dots, x_n, P', R, P_1, \dots, P_n, Q_1, \dots, Q_n$ such that $P \equiv P' \mid \prod_{i=1}^n (x_i)P_i$, $Q \equiv P' \mid R \mid \prod_{i=1}^n (x_i)Q_i$, and $P_i \preceq Q_i$, for $i = 1, \dots, n$.

Theorem 17. Let $P \in \text{CCS}^!$. Then the transition system $(\text{Deriv}(P), \xrightarrow{\tau}, \preceq)$ is a well-structured transition system with strong compatibility.

Using the above results, we may establish the following lemma.

Lemma 5. Let $P, Q \in \text{CCS}^!$. If $P \preceq Q$ and $P \xrightarrow{\lambda} P'$ then there exists Q' such that $Q \xrightarrow{\lambda} Q'$ and $P' \preceq Q'$.

Proof. Suppose $P \equiv C' \mid \prod_{i=1}^n (x_i)C_i$ and $Q \equiv C' \mid R \mid \prod_{i=1}^n (x_i)D_i$ with $C_i \preceq D_i$.

- If $C' \mid \prod_{i=1}^n (x_i)C_i \xrightarrow{\lambda} C'' \mid \prod_{i=1}^n (x_i)C_i \equiv P'$ is caused by $C' \xrightarrow{\lambda} C''$, then $C' \mid R \mid \prod_{i=1}^n (x_i)D_i \xrightarrow{\lambda} C'' \mid R \mid \prod_{i=1}^n (x_i)D_i$ and $Q \xrightarrow{\lambda} Q' \equiv C'' \mid R \mid \prod_{i=1}^n (x_i)D_i$. Thus $P' \preceq Q'$ by definition.

- If $C' \mid \prod_{i=1}^n (x_i)C_i \xrightarrow{\lambda} C' \mid (x_1)C'_1 \mid \prod_{i=2}^n (x_i)C_i \equiv P'$ is caused by $(x_1)C_1 \xrightarrow{\lambda} (x_1)C'_1$, then by structural induction, we have $D_1 \xrightarrow{\lambda} D'_1$ and $C'_1 \preceq D'_1$. Thus $C' \mid R \mid \prod_{i=1}^n (x_i)D_i \xrightarrow{\lambda} C' \mid R \mid (x_1)D'_1 \mid \prod_{i=2}^n (x_i)D_i$. That is $Q \xrightarrow{\lambda} Q' \equiv C' \mid R \mid (x_1)D'_1 \mid \prod_{i=2}^n (x_i)D_i$. Clearly $P' \preceq Q'$ by definition.

- Suppose $\lambda = \tau$ and

$$C' \mid \prod_{i=1}^n (x_i)C_i \xrightarrow{\tau} C'' \mid (x_1)C'_1 \mid \prod_{i=2}^n (x_i)C_i \equiv P'$$

is caused by $C' \xrightarrow{\alpha} C''$ and $C_1 \xrightarrow{\bar{\alpha}} C'_1$. Then by structural induction, $D_1 \xrightarrow{\bar{\alpha}} D'_1$ and $C'_1 \preceq D'_1$. Thus $C' \mid R \mid \prod_{i=1}^n (x_i)D_i \xrightarrow{\tau} C'' \mid R \mid (x_1)D'_1 \mid \prod_{i=2}^n (x_i)D_i$. That is $Q \xrightarrow{\tau} Q' \equiv C'' \mid R \mid (x_1)D'_1 \mid \prod_{i=2}^n (x_i)D_i$. And $P' \preceq Q'$ by definition.

- If $\lambda = \tau$ is caused by a communication between C_i and C_j , the proof is similar.

We are done. \square

Theorem 18. $\text{CCS}^! \sqsubseteq_{\text{ccs}} \text{CCS}^{\delta\mu}$.

Proof. We are to prove that there is no process in $\text{CCS}^!$ that is bisimilar to Z in $\text{CCS}^{\delta\mu}$ defined as follows:

$$\begin{aligned} Z &= z.Z + a.(x)(O \mid x.Z) \\ O &= \bar{a}.\bar{x} + a.(y)(E \mid y.O) \\ E &= \bar{a}.\bar{y} + a.(x)(O \mid x.E) \end{aligned}$$

Assume that there was a P in $\text{CCS}^!$ such that (P, Z) was in a subbisimilarity. Since $Z_0 \stackrel{\text{def}}{=} Z \xrightarrow{a} Z_1 \xrightarrow{a} Z_2 \xrightarrow{a} \dots$, we must have $P_0 \stackrel{\text{def}}{=} P \xrightarrow{a} P_1 \xrightarrow{a} P_2 \xrightarrow{a} \dots$ such that $P_i \approx Z_i$. Let P'_i be defined as follows:

$$P'_i \stackrel{\text{def}}{=} P_i \mid (x)(!\bar{a} \mid \mathbf{0}^i)$$

Then clearly $P'_0 \xrightarrow{\tau} P'_1 \xrightarrow{\tau} P'_2 \xrightarrow{\tau} \dots$. Since \preceq is a well quasi order in $\text{Deriv}(P'_0)$, there must be i, j such that $i < j$ and $P'_i \preceq P'_j$. By the definition of \preceq and P'_i , we have $P_i \mid (x)(!\bar{a} \mid \mathbf{0}^i) \equiv C' \mid \prod_{i=1}^n (x_i)C_i$ and $P_j \mid (x)(!\bar{a} \mid \mathbf{0}^j) \equiv C' \mid R \mid \prod_{i=1}^n (x_i)D_i$ with $C_i \preceq D_i$. Since $P_j \approx Z_j, P_j$

can only do the action \bar{a} for j times. Without loss of generality we may assume that $C_n \equiv !\bar{a} | \mathbf{0}^i$ and $D_n \equiv !\bar{a} |^j$. Then $P_i \equiv C' | \prod_{i=1}^{n-1} (x_i)C_i$ and $P_j \equiv C' | R | \prod_{i=1}^{n-1} (x_i)D_i$ with $C_i \preceq D_i$. Thus $P_i \preceq P_j$.

It follows from $P_i \approx Z_i$ that $P_i \xrightarrow{\bar{a}} \dots \xrightarrow{\bar{a}} \xrightarrow{z}$. Then by Lemma 5, we have $P_j \xrightarrow{\bar{a}} \dots \xrightarrow{\bar{a}} \xrightarrow{z}$. Again since $P_j \approx Z_j$, we have $Z_j \xrightarrow{\bar{a}} \dots \xrightarrow{\bar{a}} \xrightarrow{z}$. This is an obvious contradiction. \square

B The Proof of Theorem 4

Let $\mathcal{D}\text{rv}(E)$ be the set of derivatives of E formally defined as $\{E' | E \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_n} E' \text{ for } n \geq 0\}$. For a CCS $^{d\mu}$ expression E let F, F' be two elements of $\mathcal{D}\text{rv}(\llbracket E \rrbracket)$. We shall write $F \xrightarrow{\tau_0} F'$ if $F \xrightarrow{\tau} F'$ such that the τ -action is either caused by the explicit τ prefix introduced by the encoding in Figure 2, or by a communication between the local names x_\top, x_\perp or u introduced by the encoding in Figure 2. We use the standard notation $\xrightarrow{\tau_0}^*$ for the reflexive and transitive closure of $\xrightarrow{\tau_0}$, and $\xrightarrow{\tau_0}^n$ for a sequence of n $\xrightarrow{\tau_0}$ concatenated one after another.

Using the notations introduced in Figure 2, one could define a relation \mathcal{R} by the following

$$\left\{ (E_1 | \dots | E_n, T_1 | \dots | T_n) \left| \begin{array}{l} (x_b)C_{E_i}^b \xrightarrow{\tau_0}^* T_i \\ \text{or } (u)V_{E_i|E}^b \xrightarrow{\tau_0}^* T_i \\ \text{or } (u)W_{E|E_i}^b \xrightarrow{\tau_0}^* T_i \end{array} \right. \right\}$$

We are to prove that $\mathcal{R} \sim$ is a subbisimilarity from $\pi^{d\mu}$ to π . We need to prove several technical lemmas.

Lemma 6. $(x_{\bar{b}})C_E^b \sim (u)C_E^b \sim C_E^b$.

Proof. The equivalences are obvious due to the following facts: In $C_E^b, V_{E_1|E_2}^b$ and $W_{E_1|E_2}^b$, every occurrence of $C_{E'}^b$ or $x_{\bar{b}}$ is restricted by $(x_{\bar{b}})$; and in C_E^b , every occurrence of $V_{E_1|E_2}^b$ or $W_{E_1|E_2}^b$ is restricted by (u) . \square

Lemma 7. *If either $(x_b)C_E^b \xrightarrow{\tau_0}^* T$ or $(u)V_{E|F}^b \xrightarrow{\tau_0}^* T$ or $(u)W_{E|F}^b \xrightarrow{\tau_0}^* T$, then $(x_\top)T \sim (x_\perp)T \sim (u)T \sim T$.*

Proof. Straightforward from the definition. Notice that every occurrence of $C_{E|F}^b$ in $V_{E|F}^b$ is prefixed by u . \square

Lemma 8. *The following properties hold:*

- (i) *If $C_E^b \xrightarrow{\tau_0}^n F \xrightarrow{\bar{x}_b}$ then $F \sim C_E^b$ and $F \xrightarrow{\bar{x}_b} \sim \mathbf{0}$.*
- (ii) *If $V_{E_1|E_2}^b \xrightarrow{\tau_0}^n F \xrightarrow{u}$ then $F \sim V_{E_1|E_2}^b$ and $F \xrightarrow{u} \sim C_{E_1|E_2}^b$.*

(iii) *If $W_{E_1|E_2}^b \xrightarrow{\tau_0}^n F \xrightarrow{\bar{u}}$ then $F \sim W_{E_1|E_2}^b$ and $F \xrightarrow{\bar{u}} \sim \mathbf{0}$.*

Proof. By the definition of strong bisimilarity and the definition of the encoding, one has the followings: (i) if $F \sim C_E^b$ then $F \xrightarrow{\bar{x}_b} \sim \mathbf{0}$; (ii) if $F \sim V_{E_1|E_2}^b$ then $F \xrightarrow{u} \sim C_{E_1|E_2}^b$; and (iii) if $F \sim W_{E_1|E_2}^b$ then $F \xrightarrow{\bar{u}} \sim \mathbf{0}$. We prove the lemma by induction on n . The case $n = 0$ is trivial. Suppose that the lemma is true for all $n < k$, we show that it is true for $n = k$.

Let F_0 be either C_E^b , or $V_{E_1|E_2}^b$, or $W_{E_1|E_2}^b$ and that

$$F_0 \xrightarrow{\tau_0} F_1 \xrightarrow{\tau_0} \dots \xrightarrow{\tau_0} F_k \equiv F \xrightarrow{\lambda}$$

where correspondingly λ is either \bar{x}_b , or u , or \bar{u} . If there exist i and j such that $i < j \leq k$ and $F_i \sim F_j$, then $F \sim F_0$. This can be proved as follows: If $j = k$ then $F \sim F_i \sim F_0$ by induction. Otherwise $F_j \xrightarrow{\tau_0}^{(k-j)} F_k$ and the definition of the strong bisimilarity imply that we have $F_i \xrightarrow{\tau_0}^{(k-j)} F'_k \sim F_k \equiv F$ for some F'_k . Then $F_0 \xrightarrow{\tau_0}^{(i+k-j)} F'_k \xrightarrow{\bar{x}_b}$. Since $i + k - j < k$, by induction we have $F \sim F'_k \sim F_0$.

We now prove that the i and j in the above argument must exist:

- $F_0 \equiv C_{\mathbf{0}}^b$ or $F_0 \equiv C_{\lambda, E_1}^b$: In this case k must be 0. There is nothing to prove.
- $F_0 \equiv C_{(a)E_1}^b$: In this case we simply consider $C_{E_1}^b$.
- $F_0 \equiv C_{E_1+E_2}^b$: Consider that

$$C_{E_1+E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_1}^b | x_{\bar{b}}.C_{E_1+E_2}^b) \equiv F_1$$

Since $C_{E_1}^b \sim (x_b)C_{E_1}^b$ by Lemma 6, we must have that

$$C_{E_1}^b \xrightarrow{\tau_0}^* \not\xrightarrow{\bar{x}_b}$$

$$C_{E_1}^b \xrightarrow{\tau_0}^{n_1} \xrightarrow{\bar{x}_b} G_1$$

for some G_1 and

$$F_1 \xrightarrow{\tau_0}^{(n_1+1)} (x_{\bar{b}})(G_1 | C_{E_1+E_2}^b) \equiv F_{(n_1+2)} \xrightarrow{\tau_0}^{(k-n_1-2)} F$$

Since $n_1 < k$, we have $G_1 \sim \mathbf{0}$ by induction. Thus $F_{(n_1+2)} \equiv (x_{\bar{b}})(G_1 | C_{E_1+E_2}^b) \sim C_{E_1+E_2}^b \equiv F_0$ by Lemma 6.

If $C_{E_1+E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_2}^b | x_{\bar{b}}.C_{E_1+E_2}^b) \equiv F_1$, then the argument is the same.

- $F_0 \equiv C_{E_1|E_2}^b$: Consider that

$$C_{E_1|E_2}^b \xrightarrow{\tau_0} (u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b) \equiv F_1$$

Since $C_{E_1}^{\bar{b}} \sim (x_b)C_{E_1}^{\bar{b}}$, we have $C_{E_1}^{\bar{b}} \xrightarrow{\tau_0} \not\sim^*$ and thus $V_{E_1|E_2}^b \xrightarrow{\tau_0} \not\sim^*$. The same property holds of $W_{E_1|E_2}^b$. By induction we must have

$$V_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n_1} u \rightarrow G_1 \sim C_{E_1|E_2}^b$$

$$W_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n_2} \bar{u} \rightarrow G_2 \sim \mathbf{0}$$

for some G_1, G_2 and consequently

$$F_1 \xrightarrow{\tau_0} \xrightarrow{(n_1+n_2+1)} F_{(n_1+n_2+2)} \sim C_{E_1|E_2}^b \equiv F_0$$

- $F_0 \equiv V_{E_1|E_2}^b$: The first internal action must be

$$V_{E_1|E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.V_{E_1|E_2}^b)$$

Since $C_{E_1}^{\bar{b}} \sim (u)C_{E_1}^{\bar{b}} \xrightarrow{\tau_0} \not\sim^*$, we must have by induction that $C_{E_1}^{\bar{b}} \xrightarrow{\tau_0} \xrightarrow{n_1} \xrightarrow{\bar{x}_{\bar{b}}} G_1 \sim \mathbf{0}$ for some G_1 . Therefore

$$\begin{aligned} F_0 &\xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.V_{E_1|E_2}^b) \\ &\xrightarrow{\tau_0} \xrightarrow{(n_1+1)} F_{n_1+2} \sim V_{E_1|E_2}^b \equiv F_0 \end{aligned}$$

- $F_0 \equiv W_{E_1|E_2}^b$: The proof is almost the same as for $V_{E_1|E_2}^b$.

We are done. \square

Lemma 9. *The following properties hold:*

- (i) If $C_E^b \xrightarrow{\tau_0} \xrightarrow{n} F$ then $F \xrightarrow{\tau_0} \xrightarrow{m} G \sim C_E^b$ for some m and G ;
- (ii) If $V_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n} F$ then $F \xrightarrow{\tau_0} \xrightarrow{m} G \sim V_{E_1|E_2}^b$ for some m and G ;
- (iii) If $W_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n} F$ then $F \xrightarrow{\tau_0} \xrightarrow{m} G \sim W_{E_1|E_2}^b$ for some m and G .

Proof. We prove the lemma by induction on n . The $n = 0$ case is trivial. Suppose the lemma is true for all $n < k$, we show that it is true for $n = k$ as well.

Let F_0 be either C_E^b , or $V_{E_1|E_2}^b$, or $W_{E_1|E_2}^b$ and that

$$F_0 \xrightarrow{\tau_0} F_1 \xrightarrow{\tau_0} \dots \xrightarrow{\tau_0} F_k \equiv F$$

If there exist i, j such that $i < j \leq k$ and $F_i \sim F_j$, then there exist m and G such that $F \xrightarrow{\tau_0} \xrightarrow{m} G$ and $G \sim F_0$. This can be proved as follow. If $j = k$ then we may simply use

the induction. Otherwise it follows from $F_j \xrightarrow{\tau_0} \xrightarrow{(k-j)} F_k$ and the definition of strong bisimilarity that $F_i \xrightarrow{\tau_0} \xrightarrow{(k-j)} F'_k \sim F_k \equiv F$. Then $F_0 \xrightarrow{\tau_0} \xrightarrow{(k+i-j)} F'_k$. Since $k+i-j < k$, by induction we have $F'_k \xrightarrow{\tau_0} \xrightarrow{m} G'$ and $G' \sim F_0$ for some m and G' . Since $F'_k \sim F$, we have $F \xrightarrow{\tau_0} \xrightarrow{m} G$ and $G \sim G' \sim F_0$ for some G .

Next we now prove the i, j in the above argument do exist:

- $F_0 \equiv C_{\mathbf{0}}^b$ or $F_0 \equiv C_{\lambda.E}^b$: In this case there is nothing to prove.
- $F_0 \equiv C_{(a)E_1}^b$: In this case we consider $C_{E_1}^b$.
- $F_0 \equiv C_{E_1+E_2}^b$: Suppose that

$$C_{E_1+E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.C_{E_1+E_2}^b)$$

is the first internal action. If $C_{E_1}^{\bar{b}} \xrightarrow{\tau_0} \xrightarrow{j} \xrightarrow{\bar{x}_{\bar{b}}} \sim \mathbf{0}$ for some $j < k$ then

$$F_0 \xrightarrow{\tau_0} \xrightarrow{j+1} \sim (x_{\bar{b}})C_{E_1+E_2}^b \sim F_0$$

If

$$\begin{aligned} &(x_{\bar{b}})(C_{E_1}^{\bar{b}} | x_{\bar{b}}.C_{E_1+E_2}^b) \\ &\xrightarrow{\tau_0} \xrightarrow{n-1} (x_{\bar{b}})(H' | x_{\bar{b}}.C_{E_1+E_2}^b) \equiv F \end{aligned}$$

then by induction we have $H' \xrightarrow{\tau_0} \xrightarrow{m'} G'$ and $G' \sim C_{E_1}^{\bar{b}}$ for some m' and G' . But then

$$F \xrightarrow{\tau_0} \xrightarrow{m'} (x_{\bar{b}})(G' | x_{\bar{b}}.C_{E_1+E_2}^b) \sim F_1$$

and since $F_1 \xrightarrow{\tau_0} (x_{\bar{b}})(\mathbf{0} | C_{E_1+E_2}^b)$, there must exist some G such that $F \xrightarrow{\tau_0} \xrightarrow{m'+1} G$ and $G \sim (x_{\bar{b}})(\mathbf{0} | C_{E_1+E_2}^b) \sim F_0$.

- $F_0 \equiv C_{E_1|E_2}^b$: Then obviously

$$C_{E_1|E_2}^b \xrightarrow{\tau_0} (u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b) \equiv F_1$$

If, for some $1 \leq j < k$, $F_j \equiv (u)(P|Q)$, and F_{j+1} is obtained by the synchronization between P and Q through u , then it follows from Lemma 8 that $P \sim V_{E_1|E_2}^b$, $Q \sim W_{E_1|E_2}^b$ and $F_j \sim F_0$. So we have found the i, j . If there is no synchronization on u between P and Q , then $F \equiv (u)(P|Q)$, $V_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n_1} P$ and $W_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{n_2} Q$. By induction we have $P \xrightarrow{\tau_0} \xrightarrow{m_1} \sim V_{E_1|E_2}^b$ and $Q \xrightarrow{\tau_0} \xrightarrow{m_2} \sim W_{E_1|E_2}^b$ for some m_1 and m_2 . Then $F \xrightarrow{\tau_0} \xrightarrow{m_1+m_2+1} G \sim F_0$ for some G .

- $F_0 \equiv V_{E_1|E_2}^b$ or $F_0 \equiv W_{E_1|E_2}^b$: The proof is almost the same as for $C_{E_1+E_2}^b$.

We are done. \square

Lemma 10. *The followings hold if $\lambda \notin \{\bar{x}_b, u, \bar{u}, \tau_0\}$:*

- (i) If $C_E^b \xrightarrow{\tau_0} T \xrightarrow{\lambda} T'$, then there exists some E' such that $E \xrightarrow{\lambda} E'$ and $E'\mathcal{R} \sim T'$.
- (ii) If $V_{E_1|E_2}^b \xrightarrow{\tau_0} T \xrightarrow{\lambda} T'$, then there exists some E' such that $E_1 \xrightarrow{\lambda} E'$ and $E'\mathcal{R} \sim T'$.
- (iii) If $W_{E_1|E_2}^b \xrightarrow{\tau_0} T \xrightarrow{\lambda} T'$, then there exists some E' such that $E_2 \xrightarrow{\lambda} E'$ and $E'\mathcal{R} \sim T'$.

Proof. We prove the lemma by induction on n . Let F_0 be either C_E^b , or $V_{E_1|E_2}^b$, or $W_{E_1|E_2}^b$. When $n = 0$, the only possibility is $F_0 \equiv C_{\lambda.E}^b$. In this case $C_{\lambda.E}^b \xrightarrow{\lambda} (x_b)C_E^b$. Clearly $\lambda.E \xrightarrow{\lambda} E$ and $E\mathcal{R}(x_b)C_E^b$ by the definition of \mathcal{R} . Suppose that the lemma is true for $n < k$, we prove that it is also true for $n = k$.

- $F_0 \equiv C_{(a)E_1}^b$: Consider $C_{E_1}^b$.
- $F_0 \equiv C_{E_1+E_2}^b$: Assume that the first internal action is

$$C_{E_1+E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(C_{E_1}^b | x_{\bar{b}}.C_{E_1+E_2}^b)$$

Suppose that there is a synchronization on the explicit $x_{\bar{b}}$ at the i -th internal action. In other words the assumption takes the following form for some F :

$$C_{E_1+E_2}^b \xrightarrow{\tau_0} F \xrightarrow{\tau_0} T$$

Then by Lemma 8 we have $F \sim C_{E_1+E_2}^b$. Thus by the definition of strong bisimilarity, one gets that

$$C_{E_1+E_2}^b \xrightarrow{\tau_0} F \xrightarrow{\lambda} F' \sim T'$$

for some F' . By induction some E' exists such that

$$E_1+E_2 \xrightarrow{\lambda} E'\mathcal{R} \sim F' \sim T'$$

If such synchronization does not exist, then $T' \equiv (x_{\bar{b}})(T_1' | x_{\bar{b}}.C_{E_1+E_2}^b)$ and $C_{E_1}^b \xrightarrow{\tau_0} T_1'$. By induction, $E_1 \xrightarrow{\lambda} E_1'\mathcal{R} \sim T_1'$ for some E_1' . Therefore $E_1+E_2 \xrightarrow{\lambda} E_1'\mathcal{R} \sim T'$.

- $F_0 \equiv C_{E_1|E_2}^b$: The first internal action is

$$C_{E_1|E_2}^b \xrightarrow{\tau_0} (u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b)$$

If there is a synchronization on the explicit u at the i -th internal action like $C_{E_1|E_2}^b \xrightarrow{\tau_0} F \xrightarrow{\tau_0} T$ for some F , then by Lemma 8 we have $F \sim$

$C_{E_1|E_2}^b$. Thus by the definition of strong bisimilarity, $C_{E_1|E_2}^b \xrightarrow{\tau_0} F \xrightarrow{\lambda} F' \sim T'$ for some F' . It follows by induction that $E_1|E_2 \xrightarrow{\lambda} E'\mathcal{R} \sim F' \sim T'$.

If $T' \equiv (u)(T_1'|T_2')$, $V_{E_1|E_2}^b \xrightarrow{\tau_0} T_1 \xrightarrow{\lambda} T_1'$ for some T_1 and $W_{E_1|E_2}^b \xrightarrow{\tau_0} T_2 \xrightarrow{\lambda} T_2'$, then by induction one has that $E_1 \xrightarrow{\lambda} E_1'\mathcal{R} \sim T_1'$ for some E_1' . It follows from $T' \equiv (u)(T_1'|T_2') \sim T_1'|T_2'$ and $(u)W_{E_1|E_2}^b \xrightarrow{\tau_0} (u)T_2$ that $E_1|E_2 \xrightarrow{\lambda} E_1'|E_2'\mathcal{R} \sim T'$.

If $T' \equiv (u)(T_1'|T_2')$, $V_{E_1|E_2}^b \xrightarrow{\tau_0} T_1 \xrightarrow{a} T_1'$ and $W_{E_1|E_2}^b \xrightarrow{\tau_0} T_2 \xrightarrow{a} T_2'$ for some T_1, T_1', T_2, T_2' , then we have by induction that $E_1 \xrightarrow{a} E_1'\mathcal{R} \sim T_1'$ for some E_1' and $E_2 \xrightarrow{a} E_2'\mathcal{R} \sim T_2'$ for some E_2' . It is then clear that $E_1|E_2 \xrightarrow{\tau} E_1'|E_2'\mathcal{R} \sim T_1'|T_2' \sim (u)(T_1'|T_2') \equiv T'$.

- $T_0 \equiv V_{E_1|E_2}^b$ or $T_0 \equiv W_{E_1|E_2}^b$: The proof is almost the same as for $C_{E_1+E_2}^b$.

We are done. \square

Lemma 11. *The followings hold if $E \xrightarrow{\lambda} E'$:*

- (i) $C_E^b \xrightarrow{\tau_0} T' \xrightarrow{\lambda} T''$ for some T' such that $E'\mathcal{R} \sim T''$;
- (ii) $V_{E|F}^b \xrightarrow{\tau_0} T' \xrightarrow{\lambda} T''$ for some T' such that $E'\mathcal{R} \sim T''$;
- (iii) $W_{F|E}^b \xrightarrow{\tau_0} T' \xrightarrow{\lambda} T''$ for some T' such that $E'\mathcal{R} \sim T''$.

Proof. If $C_E^b \xrightarrow{\tau_0} T' \xrightarrow{\lambda} T''$ and $E'\mathcal{R} \sim T''$, then by definition $V_{E|F}^b \xrightarrow{\tau_0} T' \xrightarrow{\lambda} (x_{\bar{b}})(T' | x_{\bar{b}}.V_{E|F}^b) \sim T''$. And similarly for $W_{F|E}^b$. So we only have to prove (i). This is done by induction on the inference tree of $E \xrightarrow{\lambda} E'$.

- $\lambda.E \xrightarrow{\lambda} E$: Then $C_{\lambda.E}^b \xrightarrow{\lambda} (x_b)C_E^b$. It is obvious that $E'\mathcal{R}(x_b)C_E^b$.
- $E \xrightarrow{\lambda} E' \Rightarrow (x)E \xrightarrow{\lambda} (x)E'$: By induction we have $C_E^b \xrightarrow{\tau_0} T'' \xrightarrow{\lambda} T'''$ and $E'\mathcal{R} \sim T'''$. Since $C_{(x)E}^b \sim (x)C_E^b$, we have $C_{(x)E}^b \xrightarrow{\tau_0} T'' \xrightarrow{\lambda} T'''$ and it is easy to see that $(x)E'\mathcal{R} \sim (x)T''' \sim T''$.
- $E_1 \xrightarrow{\lambda} E_1' \Rightarrow E_1 + E_2 \xrightarrow{\lambda} E_1'$: By induction we have $C_{E_1}^b \xrightarrow{\tau_0} T_1' \xrightarrow{\lambda} T_1''$ and $E_1'\mathcal{R} \sim T_1''$. Then we have $C_{E_1+E_2}^b \xrightarrow{\tau_0} (x_{\bar{b}})(T_1' | x_{\bar{b}}.C_{E_1+E_2}^b) \sim T_1''$.
- $E_1 \xrightarrow{\lambda} E_1' \Rightarrow E_1|E_2 \xrightarrow{\lambda} E_1'|E_2$: By induction we have $V_{E_1|E_2}^b \xrightarrow{\tau_0} T_1' \xrightarrow{\lambda} T_1''$ and $E_1'\mathcal{R} \sim T_1''$. Then we have $C_{E_1|E_2}^b \xrightarrow{\tau_0} (u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b) \xrightarrow{\tau_0} T_1'' \xrightarrow{\lambda} T_1'''$ and $E_1'|E_2 \xrightarrow{\lambda} E_1'|E_2 \xrightarrow{\lambda} T_1'''$.

$(u)(T'_1 | W_{E_1|E_2}^b) \sim T'_1 | (u)W_{E_1|E_2}^b$. It is easily seen that $E'_1 | E_2 \mathcal{R} \sim T'_1 | (u)W_{E_1|E_2}^b$.

- $E_1 \xrightarrow{a} E'_1 \wedge E_2 \xrightarrow{\bar{a}} E'_2 \Rightarrow E_1 | E_2 \xrightarrow{\tau} E'_1 | E'_2$: By induction we have $V_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{*} \xrightarrow{a} T'_1$ and $E'_1 \mathcal{R} \sim T'_1$ for some T'_1 and likewise $W_{E_1|E_2}^b \xrightarrow{\tau_0} \xrightarrow{*} \xrightarrow{\bar{a}} T'_2$ and $E'_2 \mathcal{R} \sim T'_2$ for some T'_2 . Consequently $C_{E_1|E_2}^b \xrightarrow{\tau_0} (u)(V_{E_1|E_2}^b | W_{E_1|E_2}^b) \xrightarrow{\tau_0} \xrightarrow{*} \xrightarrow{\tau} (u)(T'_1 | T'_2) \sim T'_1 | T'_2$. It is obvious that $E'_1 | E'_2 \mathcal{R} \sim T'_1 | T'_2$.

This completes the induction. \square

We are in a position to prove the following proposition.

Proposition 3. $\mathcal{R} \sim$ is a subbisimilarity.

Proof. Assume $E_1 | E_2 | \dots | E_k \mathcal{R} T_1 | T_2 | \dots | T_k \sim T$.

- Suppose $E_1 | E_2 | \dots | E_k \xrightarrow{\lambda} E'$ for some E' . Without loss of generality there are basically two cases: either

$$E_1 | E_2 | \dots | E_k \xrightarrow{\lambda} E'_1 | E_2 | \dots | E_k$$

is caused by $E_1 \xrightarrow{\lambda} E'_1$, or

$$E_1 | E_2 | \dots | E_k \xrightarrow{\tau} E'_1 | E'_2 | \dots | E_k$$

is caused by a communication between E_1 and E_2 . In the first case, we have by Lemma 9 that $F_1 \xrightarrow{\tau_0} T_1$, where F_1 is either $(x_b)C_{E_1}^b$, or $(u)V_{E_1|E}^b$, or $(u)W_{E|E_1}^b$ for some E . By Lemma 11, we have $T_1 \xrightarrow{\tau_0} \xrightarrow{n} \xrightarrow{\lambda} \sim T'_1$ and $E'_1 \mathcal{R} \sim T'_1$ for some T'_1 . Thus $T \xrightarrow{\tau_0} \xrightarrow{(m+n)} \xrightarrow{\lambda} T'$ for some T' such that

$$E'_1 | E_2 | \dots | E_k \mathcal{R} \sim T'_1 | T_2 | \dots | T_k \sim T'$$

For the second case, suppose $E_1 \xrightarrow{a} E'_1$ and $E_2 \xrightarrow{\bar{a}} E'_2$. By Lemma 11, we have that

$$F_1 \xrightarrow{\tau_0} \xrightarrow{m_1} T_1 \xrightarrow{\tau_0} \xrightarrow{n_1} \xrightarrow{a} T'_1 \sim \mathcal{R}^{-1} E'_1$$

and

$$F_2 \xrightarrow{\tau_0} \xrightarrow{m_2} T_2 \xrightarrow{\tau_0} \xrightarrow{n_2} \xrightarrow{\bar{a}} T'_2 \sim \mathcal{R}^{-1} E'_2$$

where F_i , for $i \in \{1, 2\}$, is either $(x_b)C_{E_i}^b$, or $(u)V_{E_i|E}^b$, or $(u)W_{E|E_i}^b$, and $E'_i \mathcal{R} \sim T'_i$.

It is then clear that $T \xrightarrow{\tau_0} \xrightarrow{(m_1+n_1+m_2+n_2)} \xrightarrow{\tau} \sim T'_1 | T'_2 | \dots | T_k \sim \mathcal{R}^{-1} E'_1 | E'_2 | \dots | E_k$.

- If $T \xrightarrow{\lambda} T'$ then by the definition of strong bisimilarity, we have without loss of generality that either

$$T_1 | T_2 | \dots | T_k \xrightarrow{\lambda} T'_1 | T_2 | \dots | T_k \sim T'$$

caused by $T_1 \xrightarrow{\lambda} T'_1$, or

$$T_1 | T_2 | \dots | T_k \xrightarrow{\tau} T'_1 | T'_2 | \dots | T_k \sim T'$$

caused by $T_1 \xrightarrow{a} T'_1$ and $T_2 \xrightarrow{\bar{a}} T'_2$. For the first case we have by definition that $F_1 \xrightarrow{\tau_0} \xrightarrow{n} T_1 \xrightarrow{\lambda} T'_1$ where F_1 is either $(x_b)C_{E_1}^b$, or $(u)V_{E_1|E}^b$, or $(u)W_{E|E_1}^b$. According to Lemma 10, there exists E'_1 such that $E_1 \xrightarrow{\lambda} E'_1 \mathcal{R} \sim T'_1$. Thus $E'_1 | E_2 | \dots | E_k \mathcal{R} \sim T'_1 | T_2 | \dots | T_k \sim T'$. For the second case, we have $F_1 \xrightarrow{\tau_0} \xrightarrow{n_1} T_1 \xrightarrow{a} T'_1$ and $F_2 \xrightarrow{\tau_0} \xrightarrow{n_2} T_2 \xrightarrow{\bar{a}} T'_2$ for some T'_1, T'_2 . By Lemma 10, we have $E_1 \xrightarrow{a} E'_1 \mathcal{R} \sim T'_1$ and $E_2 \xrightarrow{\bar{a}} E'_2 \mathcal{R} \sim T'_2$ for some E'_1, E'_2 . Thus $E_1 | E_2 | \dots | E_k \xrightarrow{\tau} E'_1 | E'_2 | \dots | E_k \mathcal{R} \sim T'_1 | T'_2 | \dots | T_k \sim T'$.

This completes the proof. \square