


Open Bisimilarity for the π -Calculus with Mismatch

Tiange Liu 

School of Computing, The Australian National University, Canberra, Australia

Alwen Tiu 

School of Computing, The Australian National University, Canberra, Australia

Ross Horne 

University of Strathclyde, Glasgow, UK

Abstract

Open bisimilarity is an equivalence relation for the π -calculus that is also congruence, making it suitable to use in compositional reasoning for mobile processes and communication protocols. The original definition of open bisimilarity, due to Sangiorgi, does not account for the mismatch operator, that is crucial in modelling real-world protocols. When mismatch is present, the congruence property no longer holds for open bisimilarity. In a LICS 2018 paper, Horne et al. proposed an extension of open bisimilarity, using a history-indexed class of relations, to address this problem. That definition, however, turns out to be non-compositional as we shall demonstrate in this paper. This paper presents a new definition of open bisimilarity in the π -calculus that incorporates mismatch. This is achieved by augmenting the transition semantics of the π -calculus with an explicit assumption about name distinctions, and by requiring that open bisimulation to be closed under an arbitrary extension of the name distinctions assumption. We then prove that the resulting open bisimilarity is both an equivalence relation and a congruence.

2012 ACM Subject Classification Theory of computation \rightarrow Process calculi

Keywords and phrases mismatch, open bisimilarity, pi calculus

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.30

1 Introduction

The π -calculus is a mathematical model developed for studying and describing concurrent systems, particularly focusing on how these systems can communicate and change their structures dynamically. It was introduced by Milner, Parrow, and Walker in the late 1980s and early 1990s [18, 19]. Within this framework, equivalence relations, known as bisimilarities, were developed to compare the behaviors of processes. Milner, Parrow and Walker initially defined the late and early bisimilarity [19], which provide foundational approaches for reasoning about process equivalence. Later, Sangiorgi and Walker extended these notions by introducing open bisimilarity [22] and quasi-open bisimilarity [23], which allow for a more flexible treatment of free and bound names. These extensions address limitations in the original definitions, particularly with respect to handling name instantiation and substitution.

An important property for an equivalence relation in the π -calculus is that it is a congruence. A congruence relation ensures that if two processes are bisimilar, they remain bisimilar in any context. While late and early bisimilarities fail to satisfy congruence, as they are not preserved by the input prefix operator, both open and quasi-open bisimilarities are congruence relations. This property guarantees that if two processes are shown to be open or quasi-open bisimilar, they will exhibit identical behavior under any contextual composition.

However, Sangiorgi's definition does not account for the mismatch operator $[x \neq y]P$, which is crucial for modeling protocols that require explicit distinctions between names. The mismatch operator, denoted as $[x \neq y]P$, enables a process P to proceed only if the names x and y are distinct. If x and y are the same, the process cannot proceed and essentially behaves like the deadlock process 0.



© Tiange Liu, Alwen Tiu, and Ross Horne;

licensed under Creative Commons License CC-BY 4.0

36th International Conference on Concurrency Theory (CONCUR 2025).

Editors: Patricia Bouyer and Jaco van de Pol; Article No. 30; pp. 30:1–30:20

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We are particularly interested in congruence relations (open and quasi-open bisimilarity). With congruence, one can prove equivalence for individual components of a system and infer equivalence for the entire system without re-evaluating interactions in every context. This greatly simplifies reasoning about correctness and facilitates the modular design of mobile systems, where processes often operate in dynamic and evolving environments. While quasi-open bisimilarity with mismatch has been explored and formalised in previous work [13], open bisimilarity with mismatch remains an open topic. In this paper, we fill this gap by extending Sangiorgi's original definition of open bisimilarity to include mismatch.

Our interests in (quasi-)open bisimulation stem from its operational nature that brings it closer to implementation. In particular, it gets rid of one source of infinity – the quantification over all input when reasoning about processes that are parameteric on the input they receive. The lazy treatment of input values in open bisimulation is similar to that of symbolic bisimulation [7, 12], but with the additional advantage of being a congruence. The former allows for practical implementations of bisimulation checkers for the π -calculus and its extensions (see e.g., [5, 26, 27]), while the latter has been shown to be crucial in reasoning about privacy properties of security protocols formalised in the applied-pi calculus, a cryptographic extension of the π -calculus [15]. The addition of mismatch enables processes to make decisions based on name inequalities. This capability allows one to, for example, encode a typical if-then-else construct in programming languages and protocols, with the else branch corresponds to the mismatch prefix. This use of the mismatch operator is important in modeling real-world protocols, as such protocols typically incorporate various error conditions (that are typically captured in the else-branch of a decision point) that could potentially lead to privacy leakage, as shown in cryptographic extensions of the π -calculus, e.g., [4, 9]. By studying mismatch in a minimal setting in the π -calculus, we hope to gain insights into its uses in (cryptographic) extensions of the π -calculus.

In most presentations of the π -calculus including the original syntax, the mismatch operator is absent for several reasons. Firstly, introducing mismatch can violate the monotonicity property of substitutions in the π -calculus. Substitutions typically should not reduce the capability of a process for action [24, Chapter 1.1]. Formally, if $P \xrightarrow{\alpha} P'$, then we want $P\sigma \xrightarrow{\alpha\sigma} P'\sigma$ for all possible σ [20]. However, mismatch can lead to situations where two originally distinct names become identical after substitution, resulting in a process being less capable of performing actions due to name equality. For example, $[x \neq y]\tau$ can perform a τ -transition, but when we apply the substitution $\{y/x\}$, the process transforms into $[y \neq y]\tau$ which equals 0. Secondly, a naive extension of open bisimilarity with mismatch would result in a non-compositional open bisimilarity, breaking the congruence property that ensures the open bisimilarity is preserved under all contexts. We will use counterexamples to illustrate this point in the next section.

To maintain the monotonicity property, one crucial measure is to impose some restrictions on the substitution of names. We stipulate that, without any further assumptions, $[x \neq y]P$ can only execute if it is guaranteed that x and y can never become equal under an arbitrary context that can potentially modify x or y , e.g., if the process is nested inside an input prefix binding x or y . To account for the interactions between input names and restricted names (induced by scope extrusion in the π -calculus), a notion of *name distinctions* [22] is introduced in both the operational semantics of the π -calculus and our new definition of open bisimulation. Under this definition, these two processes $[x \neq y]\tau$ and τ can be distinguished using open bisimilarity because the former can perform a τ -transition only if the inequality between x and y is ensured, i.e., x and y cannot be equal under any possible substitutions, while the latter can perform the τ -transition unconditionally.

Horne et al. [13] introduced an extended definition of quasi-open bisimilarity that incorporates mismatch. In the same work, they also explored open bisimilarity with mismatch, proposing an extended definition based on a history-indexed syntax that allows histories to be prepended. An alternative extension of open bisimilarity was proposed in [17], also utilizing history-indexed syntax, but leveraging on a certain rigidisation relation. However, as we will demonstrate in Section 3, these definitions are fundamentally unsound in a strong sense, i.e., they are not preserved by parallel composition. We propose here a new definition of open bisimilarity that extends only slightly Sangiorgi’s original definition, by essentially requiring open bisimilarity to be preserved by additions of name distinctions. We then prove that this new open bisimilarity is both an equivalence relation and a congruence.

Outline. In Section 2 we review related work on process calculi featuring mismatch. In Section 3, we revisit the original syntax and operational semantics of the π -calculus (excluding mismatch) and review Sangiorgi’s foundational definitions of distinction and open strong bisimilarity. As all the bisimulation relations we discuss in this paper are strong bisimulation relations (where the silent transitions are taken into account in the bisimulation game), we shall simply refer to these relations as bisimulation rather than strong bisimulation. Our results should extend to weak bisimulation as well, as the issue of name distinctions is orthogonal to the strong vs weak semantics of bisimulation. In Section 4, we discuss prior attempts to incorporate mismatch into open bisimilarity and illustrate their issues through counterexamples. In Section 5, we add mismatch to the syntax and operational semantics, subsequently defining open bisimilarity with mismatch. We present a slightly revised operational semantics indexed by distinction D . Then this new definition of open bisimilarity is proven to be a equivalence relation and a congruence relation. In Section 6, we show that our open bisimilarity cannot be represented using the previously history-indexed semantics. Finally, Section 7 provides conclusions and outlines future work.

2 Related Work

We touch on some related work on process calculi featuring mismatch and work on open bisimilarity.

Parrow and Victor introduced the fusion calculus [21], which is a unified communication model where name equivalence and name fusion replace name-passing and scope extrusion, simplifying the core concepts of mobility. The Fu and Yang provided complete axiomatizations for fusion calculus both with and without the mismatch operator. The paper discusses how mismatch can complicate the semantics but is essential for certain types of behavioral equivalences and conditions. In [11], the authors investigated weak bisimulation congruences for finite π -calculus processes, focusing on challenges posed by the mismatch operator. The paper demonstrates that the standard definition of weak open congruence leads to problematic equivalence relations when mismatch is included, and proposes two alternatives: late open congruence and early open congruence.

Open bisimulation shares many similarities with symbolic bisimulation, in particular in its “lazy” treatment of name instantiations. Symbolic bisimulation has been extensively studied for different calculi: For value-passing CCS, Hennessy and Lin [12] generalised the standard notion of labelled transition graphs to symbolic transition graphs, allowing the operational semantics of value-passing processes to be represented in finite terms even when the underlying transition graph is infinite. This work introduces algorithms applicable to early and late bisimulation equivalences, extending the scope of standard bisimulation techniques.

Symbolic bisimulation for the π -calculus was addressed in [7], where symbolic transition systems are tailored to handle mobility and dynamic process communication, culminating in a sound and complete proof system for symbolic bisimulation. Delaune et al. proposed a symbolic semantics for the finite applied π -calculus [10], adapting symbolic bisimulation for modeling cryptographic protocols. While this approach is sound, it is not complete, primarily automating observational equivalence under constraints from external inputs. A more comprehensive framework was proposed in [16], which achieved both soundness and completeness for symbolic bisimulation in the full applied π -calculus. This framework also supports replication and other complex constructs, making it suitable for equivalence checking in dynamic and security-critical systems.

The original semantics provided for the applied π -calculus is sound and complete with respect to observational equivalence [1]. However, as we know, observational equivalence is not closed under input prefix, as per its definition in the context of protocol verification. This distinguishes it fundamentally from open bisimulation. Early work on symbolic bisimulation for CCS and its subsequent extension to the π -calculus can be characterised as “classical”. In these approaches, labelled transition semantics are indexed by constraints – arising from name equality or inequality in the π -calculus, and more complex terms in the applied π -calculus. Different solutions to these constraints correspond to different continuations of a given symbolic process. The constraint languages employed are based on classical logic, inherently incorporating the principle of the excluded middle for names ($x = y \vee x \neq y$). As demonstrated in [3], open bisimulation, in contrast, is inherently intuitionistic. Consequently, symbolic bisimulations are distinct from open bisimulation at that fundamental level.

Horne and Mauw have explored open bisimilarity in the applied π -calculus that features mismatch. In [14], they introduced a generalisation of the distinction relation to sets of inequalities between terms, where inequalities act as constraints that exclude unifying messages under an equational theory. Their framework was developed to support a chain of techniques for verifying privacy properties in a real-world ePassport case study, highlighting the practical relevance of open bisimilarity in applied security protocols. While their work demonstrates the applicability of the concept in a more expressive calculus, it does not establish a congruence result for the proposed bisimilarity. In contrast, this paper addresses this theoretical gap in the simpler setting of the π -calculus with mismatch, where many of the core challenges of compositional reasoning arise.

3 Preliminary definitions

In this section we review the syntax and operational semantics of the finite fragment (i.e., the fragment without replication or recursion) of the π -calculus and open bisimilarity.

3.1 π calculus

We assume a countably infinite set of *names*, whose members are ranged over by lower-case letters such as x, y and z . Figure 1 shows the original syntax and operational semantics of the π -calculus. We denote with \mathcal{P} the set of processes.

There are three kinds of prefixes for processes, i.e., the silent prefix, the output prefix, and the input prefix, indicating the capabilities of actions. The silent prefix represents an internal action that is not visible to the outside environment. It is used to model internal computation or synchronization that does not involve any communication with the external environment. The output prefixes represent an action where the process sends the name y along channel x . In free output $\bar{x}y$, the name y is not bound by the action, meaning it

$$\begin{array}{lcl}
\pi ::= & \tau & \text{(silent)} \\
& \bar{x}y & \text{(free output)} \\
& x(y) & \text{(input)} \\
P ::= & 0 & \text{(deadlock)} \\
& \nu x.P & \text{(new)} \\
& \pi.P & \text{(action)} \\
& [x = y]P & \text{(match)} \\
& P|P & \text{(par)} \\
& P + P & \text{(choice)}
\end{array}$$

$$\begin{array}{lcl}
\frac{}{\pi.P \xrightarrow{\pi} P} \text{(ACT)} & \frac{P \xrightarrow{\pi} Q \quad z \notin \text{bn}(\pi) \cup \text{fn}(\pi)}{\nu z.P \xrightarrow{\pi} \nu z.Q} \text{(RES)} & \frac{P \xrightarrow{\bar{x}z} Q \quad z \neq x}{\nu z.P \xrightarrow{\bar{x}(z)} Q} \text{(OPEN)} \\
\frac{P \xrightarrow{\bar{x}(z)} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} \nu z.(P'|Q')} \text{(CLOSE)} & \frac{P \xrightarrow{\bar{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}} \text{(COM)} & \frac{P \xrightarrow{\pi} R}{P + Q \xrightarrow{\pi} R} \text{(SUM)} \\
\frac{P \xrightarrow{\pi} Q \quad \text{bn}(\pi) \cap \text{fn}(R) = \emptyset}{P|R \xrightarrow{\pi} Q|R} \text{(PAR)} & \frac{P \xrightarrow{\pi} Q}{[x = x]P \xrightarrow{\pi} Q} \text{(MATCH)} &
\end{array}$$

■ **Figure 1** The original syntax and late transitional semantics of the π -calculus. Their symmetric variants are omitted.

is a freely chosen name. The input prefix represents an action where the process waits to receive a name on channel x , and the received name is bound within the scope of the input action. The processes can be of the form defined in Figure 1. Besides the action prefixes that we just introduced, $\nu x.P$ represents the creation of a new, private channel x within the process P . The scope of the name x is restricted to the process P , meaning x is not visible outside P . Match represents a conditional process that behaves like P can evolve if and only if x is equal to y . Parallel composition represents two processes running in parallel. Both processes can proceed independently or interact with each other through communication. Choice represents a process that can choose to behave either like process P or Q . The choice is nondeterministic, meaning the process can proceed with either option.

3.2 Distinction and open bisimilarity

As previously mentioned, names in the π -calculus are not inherently distinguished by how they can be instantiated. However, there are situations where we need to impose additional requirements on certain names, such as ensuring that some specific names remain distinct. Therefore, *distinctions* [22] are used to manage and track name inequalities explicitly. They help to maintain and enforce the conditions under which two processes can be considered behaviorally equivalent, taking into account the different ways names can be treated (bound or free) and ensuring that comparisons are meaningful even when names are abstracted or bound within processes.

A *substitution* is a mapping from names to names that is the identity everywhere except for a finite set of names, called its *domain*. Substitutions are ranged over by σ, θ, ρ , possibly with subscripts. We use the notation $\{y_1/x_1, \dots, y_n/x_n\}$ to enumerate a substitution mapping x_i to y_i , for $i \in \{1, \dots, n\}$. Given a substitution σ and names x and y , the substitution $\sigma[y/x]$ is defined as follows: $(\sigma[y/x])(z) = \sigma(z)$ if $z \neq x$, and $\sigma(x) = y$. Applications of a substitution to a syntactic expression (terms, processes, etc) are written in a postfix notation, e.g., $P\sigma$ denotes the result of applying σ to P . This notational convention is extended homomorphically to sets and relations. We require that the application of a substitution to an expression containing binders are capture-avoiding.

► **Definition 1** (Distinction). A distinction D is a finite symmetric and irreflexive relation on names. A substitution σ respects D if $(x, y) \in D$ implies $x\sigma \neq y\sigma$. We denote with \mathcal{D} the set of all distinctions. Given sets of names A and B , the distinction $A \otimes B$ is defined as:

$$A \otimes B = \{(a, b), (b, a) \mid a \in A, b \in B, a \neq b\}$$

This means that $A \otimes B$ contains all ordered pairs (a, b) and (b, a) such that a is an element of A , b is an element of B , and $a \neq b$.

Sangiorgi defined the open bisimilarity as a family of equivalence relations indexed by distinctions. Given two family of relations $\mathbb{R} = \{R^D\}_{D \in \mathcal{D}}$ and $\mathbb{S} = \{S^D\}_{D \in \mathcal{D}}$, we say \mathbb{R} is subsumed by \mathbb{S} , written $\mathbb{R} \preceq \mathbb{S}$, if for every $D \in \mathcal{D}$, we have $R^D \subseteq S^D$.

► **Definition 2** (Sangiorgi's open bisimilarity). A family of relations $\mathbb{R} = \{R^D\}_{D \in \mathcal{D}}$ is an open bisimulation if for every $D \in \mathcal{D}$, whenever PR^DQ :

- For all σ respecting D , $(P\sigma) R^{D\sigma} (Q\sigma)$;
- If $P \xrightarrow{\alpha} P'$ and α is not a bound output, then $\exists Q'$ s.t. $Q \xrightarrow{\alpha} Q'$ and $P' R^D Q'$;
- If $P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $Q \xrightarrow{\bar{x}(z)} Q'$ and $P' R^{D'} Q'$, where $D' = D \cup (\{z\} \otimes \text{fn}(P, Q))$.

We refer to \sim_o^D as open D -bisimilarity iff there exists an open bisimulation \mathbb{R} and an $R^D \in \mathbb{R}$ such that PR^DQ . Open bisimilarity refers to a special case of open D -bisimilarity where $D = \emptyset$.

However, as we mentioned in the introduction, directly adding mismatch to Sangiorgi's open bisimilarity would break the monotonicity property. The mismatch operator typically behaves as "if $x \neq y$ then P " [20], formally,

$$\frac{P \xrightarrow{\pi} Q}{[x \neq y]P \xrightarrow{\pi} Q} \text{ (MISMATCH)} \quad (1)$$

This definition is obviously not closed under substitution. As we illustrated in the introduction, $[x \neq y]P$ can execute as P , but when we apply $\{y/x\}$ to the process, $[y \neq y]P$ cannot perform any transitions.

4 History-indexed formulations of bisimilarity

As mentioned in the introduction, maintaining the monotonicity property requires imposing certain restrictions on name substitution. Several works have attempted to address this challenge. In this section, we review existing approaches to incorporating mismatch into open bisimilarity and discuss the shortcomings of each approach.

Horne et al. [13] presented an extended open bisimilarity using the notion of history h , which is a list of names annotated with either i (denoting an input name) or o (denoting an output name). A history h captures the sequence of names that a process sends and receives during its transitions. It is essentially a (partial) trace of a process. The use history-indexed open bisimulation was first proposed in [25] for the spi-calculus [2], a cryptographic extension of the π -calculus, although the idea dates back to the encoding of the π -calculus in a logical framework [27], where the quantifier alternation used in statements surrounding bisimilarity captures the notion of histories we use here. A history or a quantifier alternation gives rise to an implicit encoding of distinction, and is very natural from both the logical and an implementation perspective. The latter, for example, could build on existing logical frameworks (such as [5]) to provide a simple implementation of open bisimulation. It is thus

a natural question to ask whether one could reformulate open bisimulation using histories to encode distinctions. The answer for the case without mismatch is in the affirmative; in the presence of mismatch, this turns out to be not true. We shall come back to this in Section 6.

We simplify the notion of histories from [25] to include only names, rather than general terms. Names annotated with o (written as z^o) represent the output names that a process emits in its bound output transitions. Conversely, names annotated with i (written as z^i) signify the symbolic inputs (i.e., variables) received by the process. The difference between these two types of annotations is formalised in the following definition of respectful substitutions.

► **Definition 3** (Substitutions with respect to a history). *A substitution σ respects h if, for all h', h'' and x such that $h = h' \cdot x^o \cdot h''$, we have $x\sigma = x$ and $y\sigma \neq x$ for all $y \in h'$.*

In this context, the o -annotated names act like constants, while i -annotated names act like scoped variables, with their scoping determined by their relative positions in the history. Viewing a history as a trace of names that a process inputs and outputs, this scoping ensures that a name received earlier in the trace cannot be identified with a fresh name outputted later. Tiu and Miller have discussed in [27, Corollary 22] that open bisimulation indexed by histories does not affect the resulting notion of bisimilarity compared to Sangiorgi's original definition.

Given a history h , it is easy to generate a distinction that can functionally replace h . Let P and Q be two processes, h be a history. Define a distinction D_h from h such that

$$D_h = \{(x_i, x_j) \mid i \neq j \text{ and } x_i, x_j \in h^o, \text{ or } i < j \text{ and } x_i \in h^i, x_j \in h^o\} \quad (2)$$

The converse does not hold, as not all possible distinctions can be represented as histories. However, the open bisimilarity relation (i.e., the largest open bisimulation), defined via the history-indexed style, coincides with Sangiorgi's original definition (Definition 2), as one can show that open-bisimilarity between two processes (under the empty distinction) can be witnessed by a bisimulation relation indexed only by distinctions that are representable as histories [27].

With the history-indexed semantics, the operational semantics of mismatch can be adapted as follows.

$$\frac{h : P \xrightarrow{\pi} Q \quad h \models x \neq y}{h : [x \neq y]P \xrightarrow{\pi} Q} \text{ (MISMATCH)}$$

where $h \models x \neq y$ iff $x\sigma \neq y\sigma$ for all σ respecting h . Other operational semantics can be indexed by history similarly.

This semantics successfully preserves the monotonicity rule mentioned in the introduction. The history-indexed monotonicity rule can be stated as: Suppose $h : P \xrightarrow{\pi} Q$. Then $h\sigma : P\sigma \xrightarrow{\pi\sigma} Q\sigma$ for all σ that respect h and satisfy for all $x \in \text{bn}(\pi)$, $y\sigma = x$ iff $x = y$. The proof that monotonicity holds with the history-indexed mismatch rule is provided in [17].

While the monotonicity issue has been resolved, the congruence problem remains challenging. Let us look at a history-indexed open bisimilarity defined in [17].

► **Definition 4** (History-indexed open bisimilarity [17]). *An open bisimulation is a history-indexed collection $\{\mathcal{B}_o^h \mid h \in \mathcal{H}\}$ of symmetric relations on processes such that whenever $P\mathcal{B}_o^h Q$:*

- *For all substitutions σ respecting h , we have $P\sigma\mathcal{B}_o^{h\sigma}Q\sigma$.*
- *If $h : P \xrightarrow{\alpha} P'$ then $\exists Q'$ s.t. $h : Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{B}_o^{h\sigma}Q'$, where α is of the form τ or $\bar{x}y$;*

- If $h : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{\bar{x}(z)} Q'$ and $P' \mathcal{B}_o^{h \cdot z^o} Q'$;
- If $h : P \xrightarrow{x(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{x(z)} Q'$ and $P' \mathcal{B}_o^{h \cdot z^i} Q'$.

The pointwise union of all open bisimulations is denoted by $\{\sim_o^h \mid h \in \mathcal{H}\}$. We refer to \sim_o^h as open h -bisimilarity. We write $P \sim_o^{h'} Q$ if there exists an open bisimulation $\{\mathcal{B}_o^h \mid h \in \mathcal{H}\}$ and a history h' with only i -annotated names such that $P \mathcal{B}_o^{h'} Q$ and $\text{fn}(P, Q) \subseteq h'$. We call P and Q open bisimilar.

This definition aligns with Sangiorgi's open bisimilarity (Def. 2). In the third clause with bound output transitions, the history (environment) is updated with a fresh private name, which serves the same function as adding the fresh name to the distinction set.

This definition is not congruent in the presence of the mismatch operator. The process $[x \neq y]\tau$ under the history $h = x^i \cdot y^i$ is open bisimilar to 0, as there exists a respectful substitution $\{x/y\}$ that invalidates $x \neq y$, thereby preventing the τ -transition from $[x \neq y]\tau$. However, this open bisimilarity does not extend to $\bar{a}(x).[x \neq y]\tau$ and $\bar{a}(x).0$ under the history $h' = y^i \cdot x^o$, since $[x \neq y]$ always holds under the history h' .

To address this limitation, a few studies have been conducted. Horne et. al. [13] provided an extended version of the history-indexed open bisimilarity that allows to extend histories in the past, which gives rise to a strictly coarser semantics. Their definition can be stated as follows.

► **Definition 5** (Extended open bisimilarity [13]). *A symmetric relation R indexed by an environment is an open bisimulation whenever, if PR^hQ the following hold:*

- For all σ respecting h , $P\sigma R^{h\sigma}Q\sigma$;
- For any history h' , we have $PR^{h' \cdot h}Q$;
- If $h : P \xrightarrow{\alpha} P'$ and α is of the form τ or $\bar{x}y$, then $\exists Q'$ s.t. $h : Q \xrightarrow{\alpha} Q'$ and $P'R^hQ'$;
- If $h : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{\bar{x}(z)} Q'$ and $P'R^{h \cdot x^o}Q'$.
- If $h : P \xrightarrow{x(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{x(z)} Q'$ and $P'R^{h \cdot x^i}Q'$.

Open bisimilarity \sim_o is defined such that $P \sim_o Q$ holds whenever there exists an open bisimulation R such that $PR^{x_1^i \cdot x_2^i \cdots x_n^i}Q$ holds, where $\text{fn}(P) \cup \text{fn}(Q) \subseteq \{x_1, x_2, \dots, x_n\}$.

The above definition extends the original open bisimilarity by the second clause allowing for the prepending of history, enabling the incorporation of additional environmental information.

As it turns out, however, this semantics is not compositional.

► **Example 6.** We look at the following two processes:

$$P = \nu k, \ell. \bar{a}k. \bar{a}\ell. a(z). ([k = z]\tau + [k \neq z]\tau) \quad \text{v.s.} \quad \nu k, \ell. \bar{a}k. \bar{a}\ell. a(z). ([\ell = z]\tau + [\ell \neq z]\tau) = Q$$

Under Definition 5, P and Q are open bisimilar (so we are treating a as an i -annotated name). After three transitions, we compare

$$P' = [k = z]\tau + [k \neq z]\tau \quad \text{v.s.} \quad [\ell = z]\tau + [\ell \neq z]\tau = Q'$$

under the history $a^i \cdot k^o \cdot \ell^o \cdot z^i$. We cannot distinguish the two processes because the history does not indicate the equality either between k and z or between ℓ and z . However the above processes can be distinguished when placed in the following context.

$$a(w) \mid [\cdot]$$

A possible partial trace for $a(w) \mid P$ is as follows:

$$\begin{aligned} a(w) \mid P &\xrightarrow{\bar{a}(k)} a(w) \mid \nu \ell. \bar{a} \ell. a(z). ([k = z]\tau + [k \neq z]\tau) \\ &\xrightarrow{a(w)} \nu \ell. \bar{a} \ell. a(z). ([k = z]\tau + [k \neq z]\tau) \\ &\xrightarrow{\bar{a}(\ell)} a(z). ([k = z]\tau + [k \neq z]\tau) \\ &\xrightarrow{a(z)} ([k = z]\tau + [k \neq z]\tau) \end{aligned}$$

These transitions would have to be matched by $a(w) \mid Q$ for the two processes to be bisimilar, so proving their bisimilarity reduces to proving bisimilarity of P' and Q' under the history $h = a^i \cdot k^o \cdot w^i \cdot \ell^o \cdot z^i$. So let us suppose that P' and Q' are bisimilar under h . Since bisimilarity must be closed under respectful substitution, and since $\{w/z\}$ respects h , it follows that

$$P'\{w/z\} = [k = w]\tau + [k \neq w]\tau \quad \text{must be bisimilar to} \quad Q'\{w/z\} = [\ell = w]\tau + [\ell \neq w]\tau$$

under $h' = h\{w/z\} = a^i \cdot k^o \cdot w^i \cdot \ell^o$. According to the definition of respectful substitutions in Definition 3, $\ell \neq w$ holds, yet neither $k = w$ nor $k \neq w$ hold. Thus $Q'\{w/z\}$ can perform a τ transition while $P'\{w/z\}$ cannot, contradicting their supposed bisimilarity.

In another attempt to address the issue [17], the authors resolved the monotonicity problem leveraging history-indexed operational semantics. Then to address the congruence property, they introduced the concept of rigidisation to refine open bisimilarity. The rigidisation of names allows us transforming an i -annotated name into an o -annotated name.

► **Definition 7** (Rigidisation relation). *The relation \subseteq_{ro} is the smallest relation on histories such that:*

- $h \subseteq_{ro} h'$ iff $h = h_1 \cdot x^i \cdot h_2$ and $h' = h_1 \cdot x^o \cdot h_2$.
- $h \subseteq_{ro} h$.
- \subseteq_{ro} is transitively closed.

Then open bisimilarity is extended by adding the second clause using the rigidisation relation.

► **Definition 8** (Rigid open bisimilarity). *An open bisimulation is a history-indexed collection $\{\mathcal{B}_o^h \mid h \in \mathcal{H}\}$ of symmetric relations on processes such that whenever $P\mathcal{B}_o^h Q$:*

- For all substitutions σ respecting h , we have $P\sigma\mathcal{B}_o^{h\sigma}Q\sigma$.
- For any $h' \supseteq_{ro} h$, we have $P\mathcal{B}_o^{h'}Q$.
- If $h : P \xrightarrow{\alpha} P'$ then $\exists Q'$ s.t. $h : Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{B}_o^h Q'$, where α is of the form τ or $\bar{x}y$;
- If $h : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{\bar{x}(z)} Q'$ and $P'\mathcal{B}_o^{h \cdot z^o} Q'$;
- If $h : P \xrightarrow{x(z)} P'$ and z is fresh, then $\exists Q'$ s.t. $h : Q \xrightarrow{x(z)} Q'$ and $P'\mathcal{B}_o^{h \cdot z^i} Q'$.

The pointwise union of all open bisimulations is denoted by $\{\sim_o^h \mid h \in \mathcal{H}\}$. We refer to \sim_o^h as open h -bisimilarity. We write $P \sim_o^h Q$ if there exists an open bisimulation $\{\mathcal{B}_o^h \mid h \in \mathcal{H}\}$ and a history h with only i -annotated names such that $P\mathcal{B}_o^h Q$ and $\text{fn}(P, Q) \subseteq h$. We call P and Q open bisimilar.

This definition is also not compositional, as the following counterexample shows.

► **Example 9.** Consider the following processes:

$$P = a(z).a(y).a(x).([x \neq y]\tau.([x = z]\tau + [x \neq z]\tau)) \quad \text{v.s.} \quad a(z).a(y).a(x).([x \neq y]\tau.\tau) = Q$$

These two processes are rigid open bisimilar according to Definition 8. To see this, observe after three inputs we get to

$$([x \neq y]\tau.([x = z]\tau + [x \neq z]\tau) \sim_o^{z^i \cdot y^i \cdot x^i} [x \neq y]\tau.\tau$$

There are two minimal ways to pass that guard, using the rigidisation. One approach is to directly rigidise x^i to x^o , which enables the condition $[x \neq z]$. Alternatively, we can first set $x = z$, and then rigidise x , in which case $[x = z]$ is enabled. However, when these processes are placed in the following context, they are no longer rigid open bisimilar.

$$C[\cdot] = \nu y.b(x).\bar{b}(y).b(z).\nu a.(\bar{a}(z).\bar{a}(y).\bar{a}(x)|[\cdot])$$

Placing the above processes in the given context and executing for 6 steps we get:

$$[x \neq y]\tau.([x = z]\tau + [x \neq z]\tau) \sim_o^{b^i.x^i.y^o.z^i} [x \neq y]\tau.\tau$$

Now, since x appears before y is output in the history, we have $x \neq y$, so we reach the following after another τ -transition.

$$[x = z]\tau + [x \neq z]\tau \sim_o^{b^i.x^i.y^o.z^i} \tau$$

Trivially process τ on the right hand side can make a τ -transition, however, since neither $x = z$ nor $x \neq z$ are decided yet in this history, that τ -transition cannot be matched on the left hand side. This gives us a distinguishing strategy.

Therefore we violate the fundamental property of open bisimilarity – that it is a congruence. That is we have $P \sim_o^h Q$, but not $C[P] \sim_o^h C[Q]$.

5 A compositional open bisimilarity with mismatch

We now present our approach to incorporating mismatch in open bisimulation in a way that ensures the resulting bisimilarity is a congruence.

5.1 Syntax and operational semantics

This section introduces a variant of the π -calculus that incorporates mismatch. The labelled operational semantics are indexed by distinction D to address mismatch in open terms, where names are treated as variables instead of distinct constants. The syntax of processes is as in Figure 1, extended with the mismatch operator $[x \neq y]P$.

Intuitively, the mismatch operator $[x \neq y]P$ is interpreted as saying that that process behaves like P if “ x and y can never be equal”, that is, x and y can be proven distinct. This interpretation obviously depends on the context of name distinction under which this process is executed, so the transition judgment in our operational semantics includes an explicit reference to a distinction. The rest of the operators are interpreted in the same way as introduced in Section 3.1.

In the context of open bisimulation, names within a process can be substituted throughout execution. Consequently, the operational semantics for mismatch must account for all possible instantiations. To address this, we present the operational semantics of the π -calculus with mismatch in Figure 2. Our formulation introduces a slight variation from the standard late operational semantics by incorporating a distinction index for each transition. This index is crucial for including the mismatch operator. Another key point to observe is that the distinctions are extended with a fresh name in the OPEN and RES rules when the ν -binder appears in process expressions. This notation ensures that such names remain unchanged by respectful substitutions. This is crucial for dealing with mismatch when a variable is bound by a ν -binder, e.g., to conclude that $\nu x.[x \neq y]\tau$ can perform a τ -transition:

$$\frac{}{\emptyset : \nu x.[x \neq y]\tau \xrightarrow{\tau} 0} \text{ (ACT)}$$

$$\frac{\{(x, y), (y, x)\} : \tau \xrightarrow{\tau} 0}{\{(x, y), (y, x)\} : [x \neq y]\tau \xrightarrow{\tau} 0} \text{ (MISMATCH)}$$

$$\frac{}{\emptyset : \nu x.[x \neq y]\tau \xrightarrow{\tau} 0} \text{ (RES)}$$

$$\begin{array}{c}
\frac{}{D : \pi.P \xrightarrow{\pi} P} \text{ (ACT)} \quad \frac{D' : P \xrightarrow{\pi} Q \quad z \notin n(\pi) \quad D' = D \cup (\{z\} \otimes \text{fn}(\nu z.P))}{D : \nu z.P \xrightarrow{\pi} \nu z.Q} \text{ (RES)} \\
\frac{D' : P \xrightarrow{\bar{x}z} Q \quad z \neq x \quad D' = D \cup (\{z\} \otimes \text{fn}(\nu z.P))}{D : \nu z.P \xrightarrow{\bar{x}(z)} Q} \text{ (OPEN)} \quad \frac{D : P \xrightarrow{\bar{x}(z)} P' \quad D : Q \xrightarrow{x(z)} Q'}{D : P|Q \xrightarrow{\pi} \nu z.(P'|Q')} \text{ (CLOSE)} \\
\frac{D : P \xrightarrow{\bar{x}y} P' \quad D : Q \xrightarrow{x(z)} Q'}{D : P|Q \xrightarrow{\pi} P'|Q'\{y/z\}} \text{ (COM)} \quad \frac{D : P \xrightarrow{\pi} Q \quad \text{bn}(\pi) \cap \text{fn}(R) = \emptyset}{D : P|R \xrightarrow{\pi} Q|R} \text{ (PAR)} \\
\frac{D : P \xrightarrow{\pi} R}{D : P + Q \xrightarrow{\pi} R} \text{ (SUM)} \quad \frac{D : P \xrightarrow{\pi} Q}{D : [x = x]P \xrightarrow{\pi} Q} \text{ (MATCH)} \quad \frac{D : P \xrightarrow{\pi} Q \quad (x, y) \in D}{D : [x \neq y]P \xrightarrow{\pi} Q} \text{ (MISMATCH)}
\end{array}$$

■ **Figure 2** The late transition semantics with mismatch of the π -calculus indexed by distinction D . Their symmetric variants are omitted. For any transition $D : P \rightarrow Q$ that z is a bound name in P , we require z to be fresh from D .

► **Lemma 10.** *Let D be a distinction and suppose σ, θ are substitutions such that σ respects D . Then θ respects $D\sigma$ if and only if $\sigma \cdot \theta$ respects D .*

The following monotonicity property of processes ensures that any substitution does not diminish the process's capability for action. We show that our extended operational semantics with mismatch preserves the monotonicity property.

► **Lemma 11 (Monotonicity).** *Suppose $D : P \xrightarrow{\pi} Q$. Then $D\sigma : P\sigma \xrightarrow{\pi\sigma} Q\sigma$ for all σ that respect D and satisfy for all $x \in \text{bn}(\pi)$, $y\sigma = x$ iff $x = y$.*

Proof. We prove the lemma by induction on the transition rules. We show here an inductive case involving the MISMATCH rule.

$$\frac{D : P \xrightarrow{\pi} Q \quad (x, y) \in D}{D : [x \neq y]P \xrightarrow{\pi} Q}.$$

We want to prove that if $D : [x \neq y]P \xrightarrow{\pi} Q$ then $D\sigma : [x\sigma \neq y\sigma]P\sigma \xrightarrow{\pi\sigma} Q\sigma$ for all σ respecting D . From the MISMATCH rule we have $D : P \xrightarrow{\pi} Q$ and $(x, y) \in D$. By induction hypothesis, $D\sigma : P\sigma \xrightarrow{\pi\sigma} Q\sigma$ for all σ respecting D . Also from Lemma 10, $(x\sigma, y\sigma) \in D\sigma$. Then following the Mismatch rule, we have

$$\frac{D\sigma : P\sigma \xrightarrow{\pi\sigma} Q\sigma \quad (x\sigma, y\sigma) \in D\sigma}{D\sigma : [x\sigma \neq y\sigma]P\sigma \xrightarrow{\pi\sigma} Q\sigma} \quad \blacktriangleleft$$

5.2 Open bisimilarity with mismatch

Indexing the transition rule by the distinction D resolves the monotonicity problem. However, this does not address all the issues. The congruence property of open bisimilarity is still not guaranteed. We look at a simple example.

► **Example 12.** Consider the following two processes: $P = [x \neq y]\tau$ and $Q = 0$. They are bisimilar ($P \sim_o^D Q$) under Sangiorgi's definition (Def. 2), because x and y are free names so that $D = \emptyset$. Yet if we place the above processes in the context $\bar{a}(x).\{\cdot\}$,

$$\bar{a}(x).([x \neq y]\tau) \not\sim_o^D \bar{a}(x).0 \quad \text{where } D = \{(x, y), (y, x), (x, a), (a, x)\}.$$

30:12 Open Bisimilarity for the π -Calculus with Mismatch

Then after the first bound output transition, the resulting processes are no longer bisimilar according to the same definition. The former process can make a τ -transition because $(x, y) \in D$, which implies that x and y cannot be equal under all respectful substitutions. In contrast, the latter process cannot make such a transition.

We then need to revise the definition of open bisimilarity to address the congruence problem. The below definition slightly extends Definition 2 by adding the second clause, imposing the closure of open bisimulation under arbitrary extensions of distinctions. For clarity and convenience in subsequent proofs, for the rest of the paper, we denote open bisimilarity as a ternary relation, relating triplets of two processes and a distinction.

► **Definition 13** (Open bisimilarity). *An open bisimulation is a ternary relation $R \subseteq \mathcal{P} \times \mathcal{P} \times \mathcal{D}$ such that R is symmetric (i.e., $(P, Q, D) \in R$ iff $(Q, P, D) \in R$) and for all $(P, Q, D) \in R$:*

- *For all σ respecting D , $(P\sigma, Q\sigma, D\sigma) \in R$*
- *For all $D' \supseteq D$, $(P, Q, D') \in R$*
- *If $D : P \xrightarrow{\alpha} P'$ and α is not a bound output, then $\exists Q'$ such that $D : Q \xrightarrow{\alpha} Q'$ and $(P', Q', D) \in R$*
- *If $D : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ such that $D : Q \xrightarrow{\bar{x}(z)} Q'$ and $(P', Q', D') \in R$, where $D' = D \cup (\{z\} \otimes \text{fn}(P, Q))$*

Open D -bisimilarity \sim^D is such that $P \sim^D Q$ whenever there exists an open bisimulation R such that $(P, Q, D) \in R$. We write $P \sim Q$ to denote $P \sim^\emptyset Q$, i.e., they are open bisimilar under the empty distinction.

Consider again Example 12. We have shown that $[x \neq y]\tau \sim_o^D 0$, for $D = \emptyset$, under Definition 2. However, according to the second clause in our new definition, we can add $\{(x, y), (y, x)\}$ to D and then $D : [x \neq y]\tau \xrightarrow{\tau} 0$ while 0 cannot make such transition. Therefore $[x \neq y]\tau$ and 0 are not open bisimilar by our extended definition.

Similarly, we look at Example 6:

$$P = \nu k, \ell. \bar{a}k. \bar{a}\ell. a(z). ([k = z]\tau + [k \neq z]\tau) \quad \text{v.s.} \quad \nu k, \ell. \bar{a}k. \bar{a}\ell. a(z). ([\ell = z]\tau + [\ell \neq z]\tau) = Q$$

We have shown that P and Q are open bisimilar under the history-indexed open bisimilarity (Def. 5). The history is $k^o \cdot \ell^o \cdot z^i$, which corresponds to a distinction $D = \{(k, \ell), (\ell, k)\}$. By our new definition, however, we could define $D' \supseteq D$ such that $D' = \{(k, \ell), (\ell, k), (\ell, z), (z, \ell)\}$.

The names ℓ and z can never be equal under distinction D' while neither $k = z$ nor $k \neq z$ is decided yet. Thus there is a distinguishing strategy where the latter process plays a τ but the former cannot match the transition. Hence, under our new definition, P and Q are not open bisimilar, which avoids the non-congruence problem.

Problems with Example 9 can be resolved in a similar manner:

$$P = a(z). a(y). a(x). ([x \neq y]\tau. ([x = z]\tau + [x \neq z]\tau)) \quad \text{v.s.} \quad Q = a(z). a(y). a(x). ([x \neq y]\tau. \tau)$$

Initially, the distinction $D = \emptyset$, as there are no bound-output names in the processes. Under our new Definition 13, if we extend D to $D' = \{(x, y), (y, x)\}$, these processes are no longer open bisimilar. This is because $x \neq y$ is guaranteed, while the equality between x and z remains undecided. This extension avoids the non-congruence issue.

A well-formulated definition of open bisimilarity should be an equivalence relation, satisfying the properties of reflexivity, symmetry, and transitivity.

► **Theorem 14.** *The open D -bisimilarity is an equivalence relation.*

- *Reflexivity:* For all P and D , $P \sim^D P$.
- *Symmetry:* For all P, Q, D , if $P \sim^D Q$, then $Q \sim^D P$.
- *Transitivity:* For all P, Q, V and D , if $P \sim^D Q$ and $Q \sim^D V$, then $P \sim^D V$.

Proof. The proof is provided in Appendix 8.1. ◀

To prove that our open bisimilarity (Def. 13) is a congruence, we must demonstrate that if two processes are open bisimilar, they remain open bisimilar under all possible contexts.

First, since the behavior of distinctions within contexts depends on the structure of the context, we define a formal mechanism to update a distinction D when it is placed within a context $C[\cdot]$. Specifically, if the context includes a restriction operator νz , which binds the name z , the distinction is updated by removing all references to z from D (as z becomes local to the context). For all other contexts, the distinction D remains unchanged.

This mechanism ensures that distinctions accurately reflect the constraints on free names in the process, while adhering to the scoping rules imposed by restrictions.

Given a distinction D and a name z , we denote with $D \setminus z$ the distinction obtained from D by removing all pairs of names in D containing z , i.e., $D \setminus z = \{(x, y) \mid x \neq z \text{ and } y \neq z\}$.

► **Definition 15** (Distinction update under context). *Let $C[\cdot]$ be an arbitrary context, and $D \in \mathcal{D}$ be a distinction. We define the distinction $C[D]$ as follows:*

$$C[D] = \begin{cases} C'[D] \setminus z & \text{if } C[\cdot] = \nu z.C'[\cdot] \\ D & \text{otherwise} \end{cases}$$

Then the congruence of open bisimilarity can be formally stated as follows:

► **Theorem 16.** *Open bisimilarity is a D -congruence. Specifically:*

- *If $P \sim^D Q$, then $C[P] \sim^{C[D]} C[Q]$ for any context C .*

The core goal of the proof is to show that the relation $(C[P], C[Q], C[D])$ is included in bisimilarity. The most direct and intuitive approach would be to construct a bisimulation that relates them explicitly and verify that it satisfies the required properties of a bisimulation. However, finding such a bisimulation relation directly can be challenging and computationally expensive, especially when dealing with complex contexts, distinctions, and substitutions.

To address this, we adapt the *up-to technique*, a powerful method introduced by Milner and later developed extensively by Sangiorgi and others [24, Chapter 2.3], which simplifies the task of proving bisimilarity by reducing the complexity of direct comparisons. Instead of checking whether P and Q are directly bisimilar, this technique defines a function F , called a *safe function*, which transforms or approximates the bisimilarity relation. By verifying that F is *safe* – that is, F behaves consistently under various transformations such as substitutions, context embeddings, and name restrictions – the proof can be carried out more abstractly and modularly.

The key idea is that instead of requiring a bisimulation to explicitly relate every process pair at every step, we demonstrate that the function F preserves the necessary properties of bisimulation when applied to the transformed processes. In this way, the proof burden shifts from directly constructing and verifying a bisimulation to ensuring that F satisfies certain progression and closure properties. This abstraction makes the up-to technique more flexible and easier to apply in practice.

Before delving into the proof, we first define *strong open progression*, which is the foundation for defining the safe function F .

► **Definition 17** (Strong open progression). *Let R and S be ternary relations such that $R, S \subseteq \mathcal{P} \times \mathcal{P} \times \mathcal{D}$. We say R strongly open progresses to S , written $R \rightsquigarrow S$, if whenever $(P, Q, D) \in R$:*

- *For all σ respecting D , $(P\sigma, Q\sigma, D\sigma) \in S$*
- *For all $D' \supseteq D$, $(P, Q, D') \in S$*
- *If $D : P \xrightarrow{\alpha} P'$ and α is not a bound output, then $\exists Q'$ such that $D : Q \xrightarrow{\alpha} Q'$ and $(P', Q', D) \in S$*
- *If $D : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ such that $D : Q \xrightarrow{\bar{x}(z)} Q'$ and $(P', Q', D') \in S$, where $D' = D \cup (\{z\} \otimes \text{fn}(P, Q))$*

Now we can introduce the safe function. The safe function F operates on relations and preserves the progression structure, ensuring that the bisimilarity relation is not broken during transformations. This property is formalised by the notion of *strong safety*, which guarantees that F behaves consistently with respect to progression.

► **Definition 18** (Strongly safe function). *A function F is strongly safe if $R \subseteq S$ and $R \rightsquigarrow S$ implies $F(R) \subseteq F(S)$ and $F(R) \rightsquigarrow F(S)$.*

We now formally define a strongly safe function and present two key lemmas that describe its properties. These can be proved similarly to the analogous lemmas in [24, Chapter 2.3].

► **Lemma 19.** *If F is strongly safe and $R \rightsquigarrow F(R)$, then R and $F(R)$ are included in \sim .*

► **Lemma 20.** *If F is strongly safe and $\sim \subseteq F(\sim)$, then $F(\sim) = \sim$.*

To apply the up-to technique, we define the function F that maps a relation R to a new relation capturing all possible transformations of R under arbitrary contexts C . The following lemma establishes that the function F is *strongly safe*.

► **Lemma 21.** *Let F be the function*

$$F(R) = \{(C[P], C[Q], C[D]) \mid C \text{ is an arbitrary context and } (P, Q, D) \in R\}.$$

Then F is strongly safe.

Proof. Suppose $R \subseteq S$ and $R \rightsquigarrow S$. We need to prove that $F(R) \subseteq F(S)$ and $F(R) \rightsquigarrow F(S)$. The inclusion $F(R) \subseteq F(S)$ follows directly from $R \subseteq S$ and the definition of F .

To prove the progression, we need to show that if C is an arbitrary context and $(P, Q, D) \in R$, $(C[P], C[Q], C[D]) \in F(R)$ by definition of F , then the progression conditions in Def. 17 are satisfied. We need to prove each of the claims below.

▷ **Claim 1.** $(C[P]\sigma, C[Q]\sigma, C[D]\sigma) \in F(S)$ for each σ that respects $C[D]$.

▷ **Claim 2.** $(C[P], C[Q], D_S) \in F(S)$ for all $D_S \supseteq C[D]$.

▷ **Claim 3.** If $C[D] : C[P] \xrightarrow{\alpha} P'$ and α is not a bound output, then $C[D] : C[Q] \xrightarrow{\alpha} Q'$ with $(P', Q', C[D]) \in F(S)$.

▷ **Claim 4.** If $C[D] : C[P] \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $C[D] : C[Q] \xrightarrow{\bar{x}(z)} Q'$ with $(P', Q', C[D']) \in F(S)$ where $C[D'] = C[D] \cup (\{z\} \otimes \text{fn}(C[P], C[Q]))$.

We present the proof of Claim 2 here, as it involves the key clause introduced in our extended definition of open bisimilarity. The remaining three claims are proved by induction on C , which requires a thorough and exhaustive case analysis. Complete proofs are provided in Appendix 8.2.

There are two cases to consider for Claim 2. In the first case, we have that the context $C[\cdot]$ does not contain any ν -binders binding a name in D_S , so we have $C[D_S] = D_S$. Let $D_1 = D \cup D_S$. Then $D_1 \supseteq D_S$ and $C[D_1] = D_S$. From the assumption that $R \rightsquigarrow S$, we have $(P, Q, D_1) \in S$, therefore $(C[P], C[Q], C[D_1]) = (C[P], C[Q], D_S) \in F(S)$ by the definition of F . For the second case, suppose that the context $C[\cdot]$ contains a ν binder that binds a name in D_S . We consider the case where $C[\cdot]$ binds exactly one name z in D_S (but this argument can be generalised to any number of binders). Let D_S^w be obtained from D_S by replacing z with a fresh name w . Since z is a binder in $C[\cdot]$ it follows that $C[D]$ contains no occurrences of z . So we also have $D_S^w \supseteq C[D]$. Then by the construction in the first case, we have $(C[P], C[Q], D_S^w) \in F(S)$. Now by Claim 1, we can apply the substitution $\{z/w\}$, which respects D_S^w , to obtain $(C[P], C[Q], D_S) \in F(S)$ as required. \blacktriangleleft

Proof of Theorem 16. Let R be an open bisimulation and $F(R)$ defined as in Lemma 21. Clearly $R = F(R)$ when $C[\cdot]$ is empty. Therefore $R \subseteq F(R)$. Then $F(R)$ is also an open bisimulation by Lemma 20. \blacktriangleleft

6 Alternative definitions of open bisimilarity with mismatch

We reflect here on semantically different definitions of open bisimilarity with mismatch. A question is whether open bisimilarity, as in Def. 13, has an equivalent definition in terms of histories, by extending Def. 5 to allow more histories to be induced. In particular, an extended definition could permit more permutations such that inputs behave like universal quantifiers and outputs of fresh names like nominal quantifiers. This is a natural question, since several deep embeddings of the π -calculus work in this way.

It turns out that using (linear) histories would result in a distinct semantics. Consider the following example process.

$$P = [x \neq y][w \neq z]([x \neq z]\tau + [y \neq z]\tau + [x \neq w]\tau + [y \neq w]\tau) \quad \text{v.s.} \quad Q = [x \neq y][w \neq z]\tau$$

Clearly, according to Def. 13 these processes are not open bisimilar, since we can extend the set of distinction with the inequalities $x \neq y$ and $w \neq z$ without having to decide whether any of the other inequalities hold.

If however a ground output in a history were used to induce $x \neq y$ and $w \neq z$ then we require a history with x^i before y^o or x^o before y^i . In the same history we also require w^i before z^o or z^o before w^i . Without loss of generality suppose that x^i is before y^o and w^i is before z^o in a history. Now, since a history is an ordered sequence of inputs and outputs we have that w^i is before y^o or y^o is before w^i , as considered here.

- If w^i is before y^o , then we have that $w \neq y$ holds and hence a τ transition is enabled for both processes.
- If y^o is before w^i , then since we assumed also that x^i is before y^o and w^i is before z^o we have that x^i is before z^o . Hence $x \neq z$ holds and hence a τ transition is enabled for both processes.

Thus, regardless of how the definition is constructed, if a definition of open bisimilarity relies on a linear history to force distinctions, the processes P and Q above will be open bisimilar.

7 Conclusion and future work

This paper presents a novel definition of open bisimilarity for the π -calculus that integrates the mismatch operator. Mismatch, while crucial for modeling protocols requiring explicit name distinctions, introduces challenges such as violating the congruence property and

breaking monotonicity. Building on Sangiorgi’s original framework, we address these issues by introducing a distinction-indexed operational semantics that allows for arbitrary extensions of distinctions. Our approach ensures that open bisimilarity remains a congruence relation even in the presence of mismatch. While prior approaches that relied on history-indexed relations or rigidisation strategies did not achieve compositionality, our method successfully addresses this limitation while also providing greater simplicity and conceptual clarity. Through formal proofs, we demonstrated that the extended open bisimilarity is both equivalent and congruent. This makes it a robust and expressive framework for reasoning about mobile processes and communication protocols, enabling better modeling of real-world systems.

As immediate future work, we plan to formalise the results of this paper in the theorem prover Isabelle/Nominal [28], building on an existing formalisation of the π -calculus [6].

Another avenue for possible future work is to generalise our transition semantics to be parameterised by *logical constraints*. Logically speaking, both distinctions and substitutions can be seen as expressing a conjunction of (in)equalities between names. This can be generalised in at least two possible directions: one in which we allow expressions of (in)equalities between (algebraic) terms (like those arising from the transition semantics for the applied π -calculus [1, 14]) and another where we allow other logical connectives (e.g., disjunction, implication, quantifiers). In such an extension, it is then natural to ask how such a logical constraint should be interpreted semantically, e.g., whether it should be viewed as a formula in classical (first-order) logic, or intuitionistic logic. We conjecture that the first interpretation (in classical logic) would give rise to the notion of symbolic bisimulation [12], and the second interpretation would give rise to open bisimulation. Work on symbolic bisimilarity for the ψ -calculus [8] takes a step in this direction. The ψ -calculus is parametrised on a constraint system that can be very general. Notably in Sec. 7 of the aforementioned paper a constraint logic is defined that must be classical for the completeness results in that paper to hold. By simply working without the law of excluded middle, we hypothesise that a notion of open bisimulation for ψ -calculi is obtained where the constraint logic can otherwise be general.

References

- 1 Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *J. ACM*, 65(1):1:1–1:41, 2018. doi:10.1145/3127586.
- 2 Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In Richard Graveman, Philippe A. Janson, Clifford Neuman, and Li Gong, editors, *CCS ’97, Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1-4, 1997*, pages 36–47. ACM, 1997. doi:10.1145/266420.266432.
- 3 Ki Yung Ahn, Ross Horne, and Alwen Tiu. A characterisation of open bisimilarity using an intuitionistic modal logic. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPICs*, pages 7:1–7:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPICs.CONCUR.2017.7.
- 4 Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 107–121. IEEE Computer Society, 2010. doi:10.1109/CSF.2010.15.
- 5 David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur, and Alwen Tiu. The bedwyr system for model checking over syntactic expressions. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, pages 391–397. Springer, 2007. doi:10.1007/978-3-540-73595-3_28.

- 6 Jesper Bengtson. The pi-calculus in nominal logic. *Arch. Formal Proofs*, 2012, 2012. URL: https://www.isa-afp.org/entries/Pi_Calculus.shtml.
- 7 Michele Boreale and Rocco De Nicola. A symbolic semantics for the pi-calculus. *Inf. Comput.*, 126(1):34–52, 1996. doi:10.1006/INCO.1996.0032.
- 8 Johannes Borgström, Ramunas Gutkovas, Ioana Rodhe, and Björn Victor. The psi-calculi workbench: A generic tool for applied process calculi. *ACM Trans. Embed. Comput. Syst.*, 14(1):9:1–9:25, 2015. doi:10.1145/2682570.
- 9 Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Trace equivalence decision: negative tests and non-determinism. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 321–330. ACM, 2011. doi:10.1145/2046707.2046744.
- 10 Stéphanie Delaune, Steve Kremer, and Mark Ryan. Symbolic bisimulation for the applied pi calculus. In Vikraman Arvind and Sanjiva Prasad, editors, *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, volume 4855 of *Lecture Notes in Computer Science*, pages 133–145. Springer, 2007. doi:10.1007/978-3-540-77050-3_11.
- 11 Yuxi Fu and Zhenrong Yang. Tau laws for pi calculus. *Theoretical Computer Science*, 308(1-3):55–130, 2003. doi:10.1016/S0304-3975(03)00202-0.
- 12 Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theor. Comput. Sci.*, 138(2):353–389, 1995. doi:10.1016/0304-3975(94)00172-F.
- 13 Ross Horne, Ki Yung Ahn, Shang-Wei Lin, and Alwen Tiu. Quasi-open bisimilarity with mismatch is intuitionistic. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 26–35. ACM, 2018. doi:10.1145/3209108.3209125.
- 14 Ross Horne and Sjouke Mauw. Discovering epassport vulnerabilities using bisimilarity. *Log. Methods Comput. Sci.*, 17(2):24, 2021. doi:10.23638/LMCS-17(2:24)2021.
- 15 Ross Horne, Sjouke Mauw, and Semen Yurkov. When privacy fails, a formula describes an attack: A complete and compositional verification method for the applied π -calculus. *Theor. Comput. Sci.*, 959:113842, 2023. doi:10.1016/J.TCS.2023.113842.
- 16 Jia Liu and Huimin Lin. A complete symbolic bisimulation for full applied pi calculus. *Theor. Comput. Sci.*, 458:76–112, 2012. doi:10.1016/J.TCS.2012.07.034.
- 17 Tiange Liu, Alwen Tiu, and Jim de Groot. Modal logics for mobile processes revisited. In *34th International Conference on Concurrency Theory (CONCUR 2023)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2023.
- 18 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Information and computation*, 100(1):1–40, 1992. doi:10.1016/0890-5401(92)90008-4.
- 19 Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Information & Computation*, 100:41–77, 1992. doi:10.1016/0890-5401(92)90009-5.
- 20 Joachim Parrow and Davide Sangiorgi. Algebraic theories for name-passing calculi. *Information and Computation*, 120(2):174–197, 1995. doi:10.1006/INCO.1995.1108.
- 21 Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 98CB36226)*, pages 176–185. IEEE, 1998. doi:10.1109/LICS.1998.705654.
- 22 Davide Sangiorgi. A theory of bisimulation for the pi-calculus. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 127–142. Springer, 1993. doi:10.1007/3-540-57208-2_10.
- 23 Davide Sangiorgi and David Walker. On barbed equivalences in pi-calculus. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR 2001 - Concurrency Theory, 12th International Conference, Aalborg, Denmark, August 20-25, 2001, Proceedings*, volume 2154 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2001. doi:10.1007/3-540-44685-0_20.

- 24 Davide Sangiorgi and David Walker. *The pi-calculus: a Theory of Mobile Processes*. Cambridge university press, 2003.
- 25 Alwen Tiu. A trace based bisimulation for the spi calculus: An extended abstract. In Zhong Shao, editor, *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007, Singapore, November 29-December 1, 2007, Proceedings*, volume 4807 of *Lecture Notes in Computer Science*, pages 367–382. Springer, 2007. doi:10.1007/978-3-540-76637-7_25.
- 26 Alwen Tiu and Jeremy E. Dawson. Automating open bisimulation checking for the spi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 307–321. IEEE Computer Society, 2010. doi:10.1109/CSF.2010.28.
- 27 Alwen Tiu and Dale Miller. Proof search specifications of bisimulation and modal logics for the pi-calculus. *ACM Trans. Comput. Log.*, 11(2):13:1–13:35, 2010. doi:10.1145/1656242.1656248.
- 28 Christian Urban. Nominal techniques in isabelle/hol. *J. Autom. Reason.*, 40(4):327–356, 2008. doi:10.1007/S10817-008-9097-2.

8 Appendix

8.1 Proof for Lemma 14

Proof. We illustrate the proof for transitivity here (the other cases are straightforward). By definition, \sim since $P \sim^D Q$, there exists an open bisimulation relation R_1 such that $(P, Q, D) \in R_1$. Similarly, there exists a R_2 such that $(Q, V, D) \in R_2$.

Let $R_3 = \{(P, V, D) \mid (P, Q, D) \in R_1, (Q, V, D) \in R_2\}$. Obviously, $(P, V, D) \in R_3$, so it suffices to show that R_3 is an open bisimulation.

- Suppose $(P, V, D) \in R_3$, then $(P, Q, D) \in R_1$ and $(Q, V, D) \in R_2$. For all σ respecting D , we have $(P\sigma, Q\sigma, D\sigma) \in R_1$, and $(Q\sigma, V\sigma, D\sigma) \in R_2$. Then by definition of R_3 , $(P\sigma, V\sigma, D\sigma) \in R_3$.
- Let D' be a distinction that $D' \supseteq D$. By definition of open bisimilarity, $(P, Q, D') \in R_1$ and $(Q, V, D') \in R_2$. Hence by definition of R_3 , $(P, V, D') \in R_3$.
- If $D : P \xrightarrow{\alpha} P'$ and α is not a bound output, then $\exists Q'$ such that $D : Q \xrightarrow{\alpha} Q'$ and $(P', Q', D) \in R_1$. Then $\exists V'$ such that $D : V \xrightarrow{\alpha} V'$ and $(Q', V', D) \in R_2$. By definition of R_3 , $(P', V', D) \in R_3$.
- If $D : P \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $\exists Q'$ such that $D : Q \xrightarrow{\bar{x}(z)} Q'$ and $(P', Q', D') \in R_1$ where $D' = D \cup (\{z\} \otimes \text{fn}(P, Q, V))$. Then $\exists V'$ such that $D : V \xrightarrow{\bar{x}(z)} V'$ and $(Q', V', D') \in R_2$ where $D' = D \cup (\{z\} \otimes \text{fn}(P, Q, V))$. By definition of R_3 , $(P', V', D') \in R_3$. ◀

8.2 Proof for Lemma 21

Proof. We prove each of the conditions below (each listed as a claim followed by a proof).

▷ **Claim 5.** $(C[P]\sigma, C[Q]\sigma, C[D]\sigma) \in F(S)$ for each σ that respects $C[D]$.

Proof. Because $R \rightsquigarrow S$, $(P\theta, Q\theta, D\theta) \in S$ for all θ respecting D and $(C[P\theta], C[Q\theta], C[D\theta]) \in F(S)$ by definition of F .

We prove the statement by induction on C . When $C = [\cdot]$, $(C[P], C[Q], C[D]) = (P, Q, D)$. Then $C[P]\sigma = P\sigma, C[Q]\sigma = Q\sigma, C[D]\sigma = D\sigma$. Since $R \rightsquigarrow S$ and $(P, Q, D) \in R$, we have $(P\sigma, Q\sigma, D\sigma) \in S$. Thus $(C[P]\sigma, C[Q]\sigma, C[D]\sigma) \in S \subseteq F(S)$.

When $C = \nu z.C'[\cdot]$, $C[D] = C'[D] \setminus z$ by Definition 15. Since substitutions must be capture-avoiding, without loss of generality, we can assume that z is not in the range of σ . Let $\sigma' = \sigma[z/z]$, i.e., σ' is obtained from σ by removing the mapping for z . We have that σ'

respects $C'[D] \setminus z$, and since z is a bound name $C'[\cdot]$, we also have $C[P]\sigma = (\nu z.C'[P])\sigma = \nu z.(C'[P]\sigma')$ and $C[Q]\sigma = (\nu z.C'[Q])\sigma = \nu z.(C'[Q]\sigma')$ and $(\nu z.C'[D])\sigma = (C'[D] \setminus z)\sigma = C'[D]\sigma'$. By induction hypothesis, $(C'[P]\sigma', C'[Q]\sigma', C'[D]\sigma') \in F(S)$. By definition of F , $(C[P]\sigma, C[Q]\sigma, C[D]\sigma) \in F(S)$.

When $C = M \mid C'[\cdot]$, by Definition 15, $C[D] = C'[D]$. By induction hypothesis, $(C'[P]\sigma, C'[Q]\sigma, C'[D]\sigma) \in F(S)$ for all σ respects $C'[D]$. By definition of F , $(M \mid C'[P]\sigma, M \mid C'[Q]\sigma, M \mid C'[D]\sigma) \in F(S)$. Because σ also respects $C[D]$, $((M \mid C'[P])\sigma, (M \mid C'[Q])\sigma, (M \mid C'[D])\sigma) \in F(S)$.

Other cases can be derived similarly. \triangleleft

\triangleright **Claim 6.** $(C[P], C[Q], D_S) \in F(S)$ for all $D_S \supseteq C[D]$.

Proof. There are two cases to consider. In the first case, we have that the context $C[\cdot]$ does not contain any ν -binders binding a name in D_S , so we have $C[D_S] = D_S$. Let $D_1 = D \cup D_S$. Then $D_1 \supseteq D_S$ and $C[D_1] = D_S$. From the assumption that $R \rightsquigarrow S$, we have $(P, Q, D_1) \in S$, therefore $(C[P], C[Q], C[D_1]) = (C[P], C[Q], D_S) \in F(S)$ by the definition of F . For the second case, suppose that the context $C[\cdot]$ contains a ν binder that binds a name in D_S . We consider the case where $C[\cdot]$ binds exactly one name z in D_S (but this argument can be generalised to any number of binders). Let D_S^w be obtained from D_S by replacing z with a fresh name w . Since z is a binder in $C[\cdot]$ it follows that $C[D]$ contains no occurrences of z . So we also have $D_S^w \supseteq C[D]$. Then by the construction in the first case, we have $(C[P], C[Q], D_S^w) \in F(S)$. Now by Claim 1, we can apply the substitution $\{z/w\}$, which respects D_S^w , to obtain $(C[P], C[Q], D_S) \in F(S)$ as required. \triangleleft

\triangleright **Claim 7.** If $C[D] : C[P] \xrightarrow{\alpha} P'$ and α is not a bound output, then $C[D] : C[Q] \xrightarrow{\alpha} Q'$ with $(P', Q', C[D]) \in F(S)$.

Proof. We prove by induction on C .

Case 1. Suppose $C = [\cdot]$, so $D : P \xrightarrow{\alpha} P'$. Since $(P, Q, D) \in R$ and $R \rightsquigarrow S$, we have $D : Q \xrightarrow{\alpha} Q'$ with $(P', Q', D) \in S$. Then $(P', Q', D) \in F(S)$ since $S \subseteq F(S)$.

Case 2. Suppose $C = \alpha.C'$ (α is not a bound output), then $P' = C'[P]$ and $C[D] = C'[D]$. $C[D] : C[Q] \xrightarrow{\alpha} C'[Q]$ by the ACT rule. Let $Q' = C'[Q]$. Because $(P, Q, D) \in R$, $(C'[P], C'[Q], C'[D]) \in F(R)$ by definition of F . Since $F(R) \subseteq F(S)$, we have $(C'[P], C'[Q], C'[D]) \in F(S)$, i.e., $(P', Q', C[D]) \in F(S)$.

Case 3. Suppose $C = \nu z.C'$, then $C[D] = C'[D] \setminus z$. By the RES rule, $C'[D] \setminus z : \nu z.C'[P] \xrightarrow{\alpha} \nu z.P'$ if $C'[D] : C'[P] \xrightarrow{\alpha} P'$. By induction hypothesis, we get $C'[D] : C'[Q] \xrightarrow{\alpha} Q'$ and $(P', Q', C'[D]) \in F(S)$. Then we have $C'[D] \setminus z : \nu z.C'[Q] \xrightarrow{\alpha} \nu z.Q'$ by the RES rule. Also by definition of F , we get $(\nu z.P', \nu z.Q', C[D]) \in F(S)$ from $(P', Q', C'[D]) \in F(S)$.

Case 4. Suppose $C = C' \mid A$, A is any process. In this case $C[D] = C'[D]$. If $C[D] : C'[P] \mid A \xrightarrow{\alpha} P' \mid A$, then $C[D] (= C'[D]) : C'[P] \xrightarrow{\alpha} P'$ according to PAR rule. By induction hypothesis, $C'[D] : C'[Q] \xrightarrow{\alpha} Q'$ and $(P', Q', C'[D]) \in F(S)$. Then we have $C[D] : C'[Q] \mid A \xrightarrow{\alpha} Q' \mid A$ by the PAR rule. From $(P', Q', C'[D]) \in F(S)$ and definition of F , we have $(P' \mid A, Q' \mid A, C[D]) \in F(S)$ as required.

Consider the symmetric case, when A is the process that is acting $C[D] : A \xrightarrow{\alpha} A'$, then $C[D] : C'[P] \mid A \xrightarrow{\alpha} C'[P] \mid A'$ and $C[D] : C'[Q] \mid A \xrightarrow{\alpha} C'[Q] \mid A'$ according to PAR rule. By definition of F , $(C'[P] \mid A', C'[Q] \mid A', C'[D] \mid A') \in F(R)$ where $C'[D] \mid A' = C[D]$. Then $(C'[P] \mid A', C'[Q] \mid A', C[D]) \in F(S)$ because $F(R) \subseteq F(S)$.

If we have $C[D] : C'[P] \xrightarrow{\bar{x}(z)} P'$; $C[D] : A \xrightarrow{x(z)} A'$, then $C[D] : C'[P] \mid A \xrightarrow{\tau} \nu z.(P' \mid A')$ according to CLOSE rule. From $C[D](= C'[D]) : C'[P] \xrightarrow{\bar{x}(z)} P'$, we have $C'[D] : C'[Q] \xrightarrow{\bar{x}(z)} Q'$ and $(P', Q', C[D']) \in F(S)$ where $C[D]' = C'[D] \cup (\{z\} \otimes \text{fn}(C'[P], C'[Q]))$ by induction hypothesis. Then $C[D] : C'[Q] \mid A \xrightarrow{\tau} \nu z.(Q' \mid A')$ by the CLOSE rule. From $(P', Q', C[D']) \in F(S)$ and the definition of F , we have $(P' \mid A', Q' \mid A', C[D']) \in F(S)$ where $C[D]' = C'[D] \cup (\{z\} \otimes \text{fn}(C'[P], C'[Q])) = C[D] \cup (\{z\} \otimes \text{fn}(C'[P], C'[Q]))$. Then by definition of F , $(\nu z.(P' \mid A'), \nu z.(Q' \mid A'), C[Q]) \in F(S)$.

When $C[D] : C'[P] \xrightarrow{x(z)} P'$ and $C[D] : A \xrightarrow{\bar{x}y} A'$, we have $C[D] : C'[P] \mid A \xrightarrow{\tau} P'\{y/z\} \mid A'$ according to the COM rule. By induction hypothesis, $C[D](= C'[D]) : C'[Q] \xrightarrow{x(z)} Q'$ so that $C[D] : C'[Q] \mid A \xrightarrow{\tau} Q'\{y/z\} \mid A'$, and $(P', Q', C[D]) \in F(S)$. Because z is a placeholder that $z \notin C[D]$, $C[D]\{y/z\} = C[D]$. Then $(P'\{y/z\}, Q'\{y/z\}, C[D]) \in F(S)$, and $(P'\{y/z\} \mid A', Q'\{y/z\} \mid A', C[D]) \in F(S)$ by definition of F .

The symmetric case is when $C[D] : C'[P] \xrightarrow{\bar{x}y} P'$ and $C[D] : A \xrightarrow{x(z)} A'$, we have $C[D] : C'[P] \mid A \xrightarrow{\tau} P' \mid A'\{y/z\}$. By induction hypothesis, $C[D] : C'[Q] \xrightarrow{\bar{x}y} Q'$ so that $C[D] : C'[Q] \mid A \xrightarrow{\tau} Q' \mid A'\{y/z\}$ and $(P', Q', C[D]) \in F(S)$. Then by definition of F , $(P' \mid A'\{y/z\}, Q' \mid A'\{y/z\}, C[D]) \in F(S)$.

Case 5. Suppose $C = [x \neq y]C'$, then $C[D] = C'[D]$. According to the MISMATCH rule, $C[D] : [x \neq y]C'[P] \xrightarrow{\pi} P'$ when $C[D] : C'[P] \xrightarrow{\pi} P'$ and $(x, y) \in D$. By induction hypothesis, we have $C[Q] : C'[Q] \xrightarrow{\pi} Q'$ and $(P', Q', C[D]) \in F(S)$. And by the MISMATCH rule, $C[D] : [x \neq y]C'[Q] \xrightarrow{\pi} Q'$.

Other cases can be proved similarly using the same technique. \triangleleft

\triangleright **Claim 8.** If $C[D] : C[P] \xrightarrow{\bar{x}(z)} P'$ and z is fresh, then $C[D] : C[Q] \xrightarrow{\alpha} Q'$ with $(P', Q', C[D']) \in F(S)$ where $C[D]' = C[D] \cup (\{z\} \otimes \text{fn}(C[P], C[Q]))$.

Consider the bound output case in **Case 2.**, when $C = \bar{x}(z).C'$, $P' = C'[P]$. In this case $C[D] = C'[D] \setminus z$ because $\bar{x}(z)$ is just the abbreviation of $\nu z.\bar{x}z$. By the ACT rule, $C[D] : C[Q] \xrightarrow{\bar{x}(z)} C'[Q]$. Let $Q' = C'[Q]$. By definition of F , $(C'[P], C'[Q], C[D]) \in F(R)$, i.e., $(P', Q', C[D] \cup (\{z\} \otimes \text{fn}(C[P], C[Q]))) \in F(R) \subseteq F(S)$.

Consider the bound output in **Case 3.** Consider the RES rule. When $C = \nu y.C'$ and $C[D] = C'[D] \setminus y$. We have $C[D] : \nu y.C'[P] \xrightarrow{\bar{x}(z)} \nu y.P'$ if $C[D]_y : C'[P] \xrightarrow{\bar{x}(z)} P'$, where $C[D]_y = C[D] \cup (\{y\} \otimes \text{fn}(C[P]))$. By induction hypothesis, $C[D]_y : C'[Q] \xrightarrow{\bar{x}(z)} Q'$ and $(P', Q', C[D]_{yz}) \in F(S)$, where $C[D]_{yz} = C[D]_y \cup (\{z\} \otimes \text{fn}(C'[P], C'[Q])) = C[D]_y \cup (\{z\} \otimes \text{fn}(C[P], C[Q]))$. By definition of F , we have $(\nu y.P', \nu y.Q', C[D']) \in F(S)$ where $C[D]' = C[D] \cup (\{z\} \otimes \text{fn}(C[P], C[Q]))$ as required. Also by the RES rule we have $C[D] : \nu y.C'[Q] \xrightarrow{\bar{x}(z)} \nu y.Q'$.

Now we look at the OPEN rule. $C[D] : \nu z.C'[P] \xrightarrow{\bar{x}(z)} P'$ when $C[D]' : C'[P] \xrightarrow{\bar{x}z} P'$ with $z \notin \{x\} \cup D$ and $C[D]' = C[D] \cup (\{z\} \otimes \text{fn}(\nu z.C'[P]))$. By induction hypothesis, $C[D]' : C'[Q] \xrightarrow{\bar{x}z} Q'$ and $(P', Q', C[D']) \in F(S)$. Also we have $C[D] : \nu z.C'[Q] \xrightarrow{\bar{x}(z)} Q'$ by the OPEN rule. \blacktriangleleft