A Theory of May Testing for Actors

Prasannaa Thati, Reza Ziaei, Gul Agha
University of Illinois at Urbana-Champaign
{thati,ziaei,agha}@cs.uiuc.edu

Abstract

The Actor model and π-calculus have served as the basis of a large body of research on concurrency. We represent the Actor model as a typed asynchronous π-calculus, called Aπ. The type system imposes a certain discipline on the use of names to capture actor properties. Since the reduction semantics of Aπ is the same as that of π-calculus, we have a basis for direct comparison between actors and π-calculus. We investigate the notion of *may* testing in Aπ and give a trace based characterization of it. We also present two variants of Aπ that differ in name matching capabilities, and discuss possible approaches to alternate characterizations of may testing for them. Our characterizations are closely related to those for variants of asynchronous π-calculus. We present a detailed comparison of our characterizations with them.

Key Words: Actors, interaction paths, may testing, π-calculus

1 Introduction

We introduce a basic calculus for the Actor model [1], called Aπ, and present a trace based characterization of may testing [7] in it. It is well-known that a trace based semantic characterization simplifies reasoning about equivalences, as it does not involve quantification over observing contexts. Both testing theories [2] and trace based models [17] have been studied for actors but the relation between them has not been investigated.

Aπ is a typed asynchronous π-calculus [3, 8, 14], where the type system enforces properties specific to the Actor model. Since the operational semantics of π-calculus is unchanged, Aπ can be seen as an embedding of the Actor model in π-calculus. This embedding not only provides a direct basis for comparison between the two models, but also enables us to apply concepts and techniques developed for π-calculus to Actors. Many formalisms for the Actor model have been proposed in the past [2, 6, 9, 16, 17] and various notions of equivalence have been considered for them [2, 6, 17]. However, none of these formalisms is directly comparable to π-calculus. On the other hand, we believe reusing a well-known formalism provides some advantages over adopting a fresh approach.

We consider three variants of Aπ which differ in their ability to compare names: one with the ability to both match and mismatch names, one with only the ability to match names, and one with a restricted form of matching that allows only comparison with names local to the process. Each of these variations leads to a different may testing theory, as the discrimination power of observers decreases with reduced ability. Variants of π-calculus that differ in name matching capabilities have been studied [11, 12], but to the best of our knowledge the notion of restricted matching is unique to ours.

We discuss the alternate characterization of may testing for all the variants of Aπ. The approach we adopt is similar to that used for characterization of may testing for asynchronous π-calculus[11]. The characterization for the variant with both match and mismatch differs from the one presented in [11] in several respects. This is because of differences between the notions of observability in actors and π-calculusthat are reflected as differences between the labeled transition system for the two models. However, despite this difference, the characterization for the variant without mismatch is very similar to that in [11]. But, the characterization for the variant with restricted match needs some radical changes to the previous techniques. Due to space limitation, we present the results for only Aπ with match and mismatch (simply Aπ from now on), and sketch the key ideas behind characterizations for the other two variants.

Following is the layout of the rest of the paper. In Section 2, we present the syntax, type system, and reduction semantics of Aπ. In Sections 3 and 4, we define a labeled transition system and use the resulting traces to give an alternate characterization of may testing for Aπ. In Section 5, we present the syntax and semantics of the other two variants of Aπ, and sketch the alternate characterization of may testing in them. In Section 6, we discuss related work and future directions of research.

2 The Calculus

2.1 The Actor Model

A computational system in the Actor Model, called a *configuration*, consists of a collection of concurrently executing actors and a collection of messages in transit [1]. Each actor has a unique name (the *uniqueness* property) and a behavior, and communicates with other actors via asynchronous messages. Actors are reactive in nature, i.e. they execute only in response to messages received. An actor's behavior is deterministic in that its response to a message is uniquely determined by the message contents. Message delivery in the Actor model is fair [4]. The delivery of a message can only be delayed for a finite but unbounded amount of time.

An actor can perform three basic actions on receiving a message: (a) create a finite number of actors with universally fresh names, (b) send a finite number of messages, and (c) assume a new behavior. Furthermore, all actions performed on receiving a message are concurrent; there is no ordering between any two of them. The following observations are in order here. First, actors are persistent in that they do not disappear after processing a message (the *persistence* property). Second, actors cannot be created with well known names or names received in a message (the *freshness* property).

2.2 Syntax

We assume an infinite set of names $\mathcal{N}$, and a set $\mathcal{B}$ of behavior identifiers. We let $u, v, w, x, y, z, \ldots$ range over $\mathcal{N}$, and B range over $\mathcal{B}$. We write $\tilde{x}$ for a tuple of names, and $len(\tilde{x})$ for the length of the tuple. For $\tilde{x}$ of length n, x_i for $i \leq n$ denotes the i^{th} component of the tuple. We let C range over the set of preterms $\mathcal{C}$, which is defined by the following context-free grammar.

$$C := \; 0 \;\mid\; x(y).C \;\mid\; \overline{x}y \;\mid\; [x=y](C_1,C_2) \;\mid\; (\nu x)C \;\mid\; C_1|C_2 \;\mid\; B\langle\tilde{x};\tilde{y}\rangle$$

The order of precedence of combinators is the order in which they appear. The nil term 0, represents an empty configuration. The output term $\overline{x}y$, represents a configuration with a single message targeted to x and with contents y. We call x the subject of the output term. The input term $x(y).C$ represents a configuration with an actor x whose behavior is $(\tilde{y})C$. We call x the subject of the input term. The composition $C_1|C_2$ is a configuration containing all the actors and messages in C_1 and C_2. The conditional $[x=y](C_1,C_2)$ is C_1 if x and y are the same names, and C_2 otherwise. The restriction $(\nu x)C$ is the same as C, except that x is now private to C. The term $B\langle\tilde{u};\tilde{v}\rangle$ is a behavior instantiation. The identifier B has a single defining equation of the form $B \overset{def}{=} (\tilde{x};\tilde{y})x_1(z).C$, where $\tilde{x}$ is a tuple of distinct names of length 1 or 2, and $\tilde{x}$, $\tilde{y}$ together contain exactly the free names in $x_1(z).C$. The definition provides a template for an actor behavior. For an instantiation $B\langle\tilde{u};\tilde{v}\rangle$ we assume $len(\tilde{u}) = len(\tilde{x})$, and $len(\tilde{v}) = len(\tilde{y})$.

2.3 Notational Conventions and Definitions

For a tuple $\tilde{x}$, we denote the set of names occurring in $\tilde{x}$ by $\{\tilde{x}\}$. We write $\tilde{x},\tilde{y}$ for the result of appending $\tilde{y}$ to $\tilde{x}$. We let $\hat{z}$ range over $\{\emptyset, \{z\}\}$. By $\tilde{x},\hat{z}$ we mean $\tilde{x},z$ if $\hat{z} = \{z\}$, and $\tilde{x}$ otherwise. The term $(\hat{z})C$ is $(\nu z)C$ if $\hat{z} = \{z\}$, and C otherwise. We write $(\nu x_1,\ldots,x_n)C$ instead of $(\nu x_1)...(\nu x_n)C$.

The functions $fn(.)$, $bn(.)$, $n(.)$ are defined on preterms the obvious way. Alpha equivalence on preterms, $\equiv_\alpha$, is defined as usual. We also use the usual definition and notational convention for name substitution, and let σ range over substitutions. For a name x we write $\sigma(x)$ for the name to which x is mapped to by σ, and for a set of names S, we write $\sigma(S)$ to denote the set obtained by applying σ to each element of S. Name substitutions on configurations are defined modulo alpha equivalence, with the usual renaming convention to avoid captures. We write $C\sigma$ to denote the result of applying the simultaneous substitution σ to C.

Let $X \subset \mathcal{N}$. We assume $\bot, * \notin \mathcal{N}$, and define $X^* = X \cup \{\bot, *\}$. For $f : X \to X^*$, we define $f^* : X^* \to X^*$ as $f^*(x) = f(x)$ for $x \in X$ and $f(\bot) = f(*) = \bot$. Further, if σ is a substitution which is one-to-one on X, we define $f\sigma : \sigma(X) \to \sigma(X)^*$ as $f\sigma(\sigma(x)) = \sigma(f(x))$, where we let $\sigma(\bot) = \bot$ and $\sigma(*) = *$.

2.4 Type System

Not all preterms represent actor configurations. Unlike π-calculus where names denote communication channels, a name in the Actor model uniquely denotes a persistent agent. To capture this object paradigm we need to impose a certain discipline on the use of names, which we do using a type system. Well-typed preterms, called terms, will represent actor configurations.

Strictly enforcing all actor properties would make Aπ too weak to express certain communication patterns. One such scenario is where, instead of assuming a new behavior immediately after receiving a message (as required by persistence property), an actor has to wait until certain synchronization conditions are met before processing the next message. For example, such a delaying mechanism is required to express polyadic communication, where an actor has to delay the assumption of a behavior and processing of other messages until all the arguments are transfered. We therefore relax the persistence requirement, and allow actors to temporarily assume a series of fresh names, one at a time, and resume the old name at a later point. Basically, the synchronization task is delegated from one new name to another until the last one releases the actor after certain synchronization conditions are met.

A typing judgment is of the form $\rho; f \vdash C$, where ρ is the set of free names in C that denote actors in C, and $f : \rho \to \rho^*$ is a function that relates actors in C to the temporary names they have assumed currently. Specifically, $f(x) = \perp$ means that x is a regular actor name and not a temporary one, $f(x) = *$ means x is the temporary name of an actor with a private name (bound by a restriction), and $f(x) = y \notin \{\perp, *\}$ means that actor y has assumed the temporary name x. The function f has the following properties: for all $x, y \in \rho$, $f(x) \neq x$, $f(x) = f(y) \notin \{\perp, *\}$ implies $x = y$, and $f^*(f(x)) = \perp$. While the first property is obvious, the second states that an actor cannot assume more than one temporary name at the same time, and the third states that temporary names are not like regular actor names in that they themselves cannot temporarily assume new names but can only delegate their capability of releasing the original actor to new names.

We define the following functions and relations that will be used in defining the type rules.

Definition 1 *Let $f_1 : \rho_1 \to \rho_1^*$ and $f_2 : \rho_2 \to \rho_2^*$.*

1. *We define $f_1 \oplus f_2 : \rho_1 \cup \rho_2 \to (\rho_1 \cup \rho_2)^*$ as*

$$(f_1 \oplus f_2)(x) = \begin{cases} f_1(x) & \text{if } x \in \rho_1, \text{ and } f_1(x) \neq \perp \text{ or } x \notin \rho_2 \\ f_2(x) & \text{otherwise} \end{cases}$$

   Note that $\oplus$ is associative.

2. *If $\rho \subset \rho_1$ we define $f|\rho : \rho \to \rho^*$ as*

$$(f|\rho)(x) = \begin{cases} * & \text{if } f(x) \in \rho_1 - \rho \\ f(x) & \text{otherwise} \end{cases}$$

3. *We say f_1 and f_2 are compatible if $f = f_1 \oplus f_2$ has following properties: $f = f_2 \oplus f_1$, and for all $x, y \in \rho_1 \cup \rho_2$, $f(x) \neq x$, $f^*(f(x)) = \perp$, and $f(x) = f(y) \notin \{\perp, *\}$ implies $x = y$.* □

Definition 2 *For a tuple $\tilde{x}$, we define $ch(\tilde{x}) : \{\tilde{x}\} \to \{\tilde{x}\}^*$ as $ch(\epsilon) = \{\}$, and if $len(x) = n$, $ch(\tilde{x})(x_i) = x_{i+1}$ for $1 \leq i < n$ and $ch(\tilde{x})(x_n) = \perp$.* □

The type rules are shown in Table 1. Rules *NIL* and *MSG* are obvious. In the *ACT* rule, if $\hat{z} = \{z\}$ then actor z has assumed temporary name x. The condition $y \notin \rho$ ensures that actors are not created with names received in a message. In the terminology of [15], only output capability of names can be passed in messages. The conditions $y \notin \rho$ and $\rho - \{x\} = \hat{z}$ together guarantee the freshness property by ensuring that new actors are created with fresh names. Note that it is possible for x to be a regular name, i.e. $\rho - \{x\} = \emptyset$, and disappear after receiving a message, i.e. $x \notin \rho$. We interpret this as the actor x assuming a *Sink* behavior that simply consumes all messages it receives. With this interpretation the persistence property is not violated.

The compatibility check in *COND* rule prevents errors such as two actors, each in a different branch, assuming the same temporary name, or the same actor assuming different temporary

$$\textit{NIL: } \emptyset; \{\} \vdash 0 \qquad\qquad \textit{MSG: } \emptyset; \{\} \vdash \overline{x}y$$

$$\textit{ACT: } \frac{\rho; f \vdash C}{\{x\} \cup \hat{z}; ch(x, \hat{z}) \vdash x(y).C} \;\; \texttt{if} \;\; \begin{array}{l} \rho - \{x\} = \hat{z},\; y \notin \rho, \; \texttt{and} \\ f = \begin{cases} ch(x, \hat{z}) & \texttt{if } x \in \rho \\ ch(\epsilon, \hat{z}) & \text{otherwise} \end{cases} \end{array}$$

$$\textit{COND: } \frac{\rho_1; f_1 \vdash C_1 \quad \rho_2; f_2 \vdash C_2}{\rho_1 \cup \rho_2; f_1 \oplus f_2 \vdash [x = y](C_1, C_2)}$$

$$\texttt{if } f_1 \texttt{ and } f_2 \texttt{ are compatible}$$

$$\textit{COMP: } \frac{\rho_1; f_1 \vdash C_1 \quad \rho_2; f_2 \vdash C_2}{\rho_1 \cup \rho_2; f_1 \oplus f_2 \vdash C_1 | C_2} \;\; \texttt{if } \;\rho_1 \cap \rho_2 = \phi$$

$$\textit{RES: } \frac{\rho; f \vdash C}{\rho - \{x\}; f | (\rho - \{x\}) \vdash (\nu x)C}$$

$$\textit{INST: } \{\tilde{x}\}; ch(\tilde{x}) \vdash B\langle \tilde{x}; \tilde{y} \rangle \quad \texttt{if } len(\tilde{x}) = 2 \texttt{ implies } x_1 \neq x_2$$

Table 1: Type rules for Aπ.

names in different branches. The *COMP* rule guarantees the uniqueness property by ensuring that the two composed configurations do not contain actors with the same name. In the *RES* rule, f is updated so that if x has assumed a temporary name y in C, then y's role as a temporary name is remembered but x is forgotten. The *INST* rule assumes that if $len(\tilde{x}) = 2$ then $B\langle \tilde{x}; \tilde{y} \rangle$ denotes an actor x_2 that has assumed temporary name x_1.

Type checking a preterm involves checking the accompanying behavior definitions. For *INST* rule to be sound, for every definition $B \overset{def}{=} (\tilde{x}; \tilde{y})x_1(z).C$ and substitution $\sigma = \{\tilde{u}, \tilde{v}/\tilde{x}, \tilde{y}\}$ that is one-to-one on $\{\tilde{x}\}$, the judgment $\{\tilde{u}\}; ch(\tilde{u}) \vdash (x_1(z).C)\sigma$ should be derivable. From Lemma 2, it follows that this constraint is satisfied if $\{\tilde{x}\}; ch(\tilde{x}) \vdash x_1(z).C$ is derivable. Thus, a preterm is well-typed only if for each accompanying behavior definition $B \overset{def}{=} (\tilde{x}; \tilde{y})x_1(z).C$, the judgment $\{\tilde{x}\}; ch(\tilde{x}) \vdash x_1(z).C$ is derivable.

The following lemma states a soundness property of the type system.

Lemma 1 *If $\rho; f \vdash C$ then $\rho \subset fn(C)$, and for all $x, y \in \rho$, $f(x) \neq x$, $f^*(f(x)) = \perp$, and $f(x) = f(y) \notin \{\perp, *\}$ implies $x = y$. Further, if $\rho'; f' \vdash C$ then $\rho = \rho'$ and $f = f'$.* $\square$

Not all substitutions on a term C yield terms. A substitution σ may identify distinct actor names in C and therefore violate the uniqueness property. But, if σ renames different actors in C to different names then $C\sigma$ will be well typed.

Lemma 2 *If $\rho; f \vdash C$ and σ is one-to-one on ρ then $\sigma(\rho); f\sigma \vdash C\sigma$.* $\square$

2.5 Reduction Semantics

Reduction semantics of Aπ is the same as that of π-calculus with mismatch. It is defined in terms of the usual structural congruence over preterms and reduction rules shown in Definition 3 and Table 2. We use $\Longrightarrow$ to denote the reflexive transitive closure of $\longrightarrow$.

$$\textit{RECV:}\quad x(y).C \mid \overline{x}z \longrightarrow C\{z/y\}$$

$$\textit{IF:}\quad [x=x](C_1,C_2) \longrightarrow C_1 \qquad \textit{ELSE:}\quad [x=y](C_1,C_2) \longrightarrow C_2 \quad \texttt{if } x \neq y$$

$$\textit{HIDE:}\quad \frac{C \longrightarrow C'}{(\nu x)C \longrightarrow (\nu x)C'} \qquad \textit{PAR:}\quad \frac{C_1 \longrightarrow C_1'}{C_1|C_2 \longrightarrow C_1'|C_2}$$

$$\textit{REQV:}\quad \frac{C_1' \longrightarrow C_2'}{C_1 \longrightarrow C_2} \quad \texttt{if} \quad \begin{array}{l} C_1 \equiv C_1' \\ C_2 \equiv C_2' \end{array}$$

Table 2: Reduction rules for Aπ.

Definition 3 (structural congruence) *The relation $\equiv$ is the smallest congruence relation on preterms closed under the following laws:*

1. *If $C_1 \equiv_\alpha C_2$ then $C_1 \equiv C_2$.*
2. *The combinator '$|$' is commutative and associative with 0 as identity.*
3. *$(\nu x,y)C \equiv (\nu y,x)C$, $(\nu x)0 \equiv 0$.*
4. *If $x \notin fn(C_2)$ then $(\nu x)C_1|C_2 \equiv (\nu x)(C_1|C_2)$.*
5. *If $B \overset{def}{=} (\tilde{x};\tilde{y})x_1(z).C$, $len(\tilde{u}) = len(\tilde{x})$, and $len(\tilde{v}) = len(\tilde{y})$ then $B\langle\tilde{u};\tilde{v}\rangle \equiv (x_1(z).C)\{(\tilde{u},\tilde{v})/(\tilde{x},\tilde{y})\}$.* □

Lemma 3 and Theorem 1 state that type system respects both the structural congruence and reduction rules.

Lemma 3 *Let $C_1 \equiv C_2$. Then $fn(C_1) = fn(C_2)$, and $\rho; f \vdash C_1$ if and only if $\rho; f \vdash C_2$.* □

Theorem 1 (subject reduction 1) *Let $\rho; f \vdash C$ and $C \longrightarrow C'$. Then $\rho'; f' \vdash C'$, for some $\rho' \subset \rho$, and $f' : \rho' \to \rho'^*$ satisfying the following conditions: $f'(x) = \bot$ if $f(x) = \bot$, $f'(x) \in \{f(x), \bot\}$ otherwise.* □

Since well-typed terms are closed under reduction, it follows that actor properties are preserved during a computation. However, note that the source and the target of a transition need not have the same typing judgment. This is because of two reasons. First, actors may disappear. As the reader may recall, this is interpreted as the actor assuming a sink behavior. Second, an actor with a temporary name may re-assume its original name, or decide to never assume it.

We show how the ability to temporarily assume a fresh name can be used to encode polyadic communication in Aπ. We assume that the subject of a polyadic receive is not a temporary name. In particular, in the encoding below, x cannot be a temporary name. The idea behind translation is to let x temporarily assume a fresh name z which is used to receive all the arguments without any interference from other messages, and re-assume x after the receipt. For fresh u, z we have

$$[\![\overline{x}\langle y_1,\ldots,y_n\rangle]\!] \quad = \quad (\nu u)(\overline{x}u \mid S_1\langle u; y_1,\ldots,y_n\rangle)$$

$$\begin{array}{rcll}
S_i & \overset{def}{=} & (u; y_i, \ldots, y_n)u(z).(\overline{z}y_i \mid S_{i+1}\langle u; y_{i+1}, \ldots, y_n\rangle) & 1 \le i < n \\
S_n & \overset{def}{=} & (u; y_n)u(z).\overline{z}y_n & \\
[\![x(y_1, \ldots, y_n).C]\!] & = & x(u).(\nu z)(\overline{u}z \mid R_1\langle z, \hat{x}; u, \tilde{a}\rangle) & \\
R_i & \overset{def}{=} & (z, \hat{x}; u, \tilde{a})z(y_i).(\overline{u}z \mid R_{i+1}\langle z, \hat{x}; u, \tilde{a}\rangle) & 1 \le i < n \\
R_n & \overset{def}{=} & (z, \hat{x}; u, \tilde{a})z(y_n).(\overline{u}z \mid [\![C]\!]) &
\end{array}$$

where $\tilde{a} = fn(x(y_1, \ldots, y_n).C) - \{x\}$, and $\hat{x} = \{x\}$ if for some ρ, f, we have $\rho \cup \{x\}; f \vdash [\![C]\!]$, and $\hat{x} = \emptyset$ otherwise.

The formalism thus far does not account for *fairness* in message deliveries that is required by the Actor model. We do not consider fairness, as it does not make a difference to the may testing theory we are concerned with. The reader is referred to Section 6 for further discussion about this.

3 May Testing Equivalence

We now instantiate the general notion of may testing [7] on Aπ. As in any typed calculus, testing in Aπ takes typing into account; an observer O can be used to test C only if $C|O$ is well typed. Since the set of valid tests varies between configurations, we parameterize the may preorder with the set of observers that is used to decide the order.

Definition 4 (may testing) *Observers are actor configurations that can emit a special message $\overline{\mu}\mu$. We let O range over the set of observers. For C, O such that $C|O$ is well-typed, we say C $\underline{may}$ O if $C|O \Longrightarrow C'|\overline{\mu}\mu$ for some C'. Let $\rho_1; f_1 \vdash C_1$ and $\rho_2; f_2 \vdash C_2$. Then for ρ such that $\rho_1, \rho_2 \subset \rho$ we say $C_1 \underset{\sim}{\sqsubset}_\rho C_2$ if for every O such that $\rho'; f' \vdash O$ and $\rho' \cap \rho = \emptyset$, C_1 $\underline{may}$ O implies C_2 $\underline{may}$ O. We say $C_1 \simeq_\rho C_2$ if $C_1 \underset{\sim}{\sqsubset}_\rho C_2$ and $C_2 \underset{\sim}{\sqsubset}_\rho C_1$. Note that $\underset{\sim}{\sqsubset}_\rho$ is reflexive and transitive, and $\simeq_\rho$ is an equivalence relation.* □

The parameter of a preorder indicates the size of the observer set that is used to decide the order; the larger the parameter, the smaller the observer set. From this observation, it is easy to see that when $\rho_1 \subset \rho_2$, we have $C_1 \underset{\sim}{\sqsubset}_{\rho_1} C_2$ implies $C_1 \underset{\sim}{\sqsubset}_{\rho_2} C_2$, but not the converse. To see why the converse doesn't hold, we have $0 \simeq_{\{x\}} \overline{x}x$, but only $0 \underset{\sim}{\sqsubset}_\emptyset \overline{x}x$ and $\overline{x}x \not\underset{\sim}{\sqsubset}_\emptyset 0$. Similarly, $\overline{x}x \simeq_{\{x,y\}} \overline{y}y$, but $\overline{x}x \not\underset{\sim}{\sqsubset}_\emptyset \overline{y}y$ and $\overline{y}y \not\underset{\sim}{\sqsubset}_\emptyset \overline{x}x$. However, the converse holds if $fn(C_1) \cup fn(C_2) \subset \rho_1$.

Theorem 2 *Let $\rho_1 \subset \rho_2$. Then $C_1 \underset{\sim}{\sqsubset}_{\rho_1} C_2$ implies $C_1 \underset{\sim}{\sqsubset}_{\rho_2} C_2$. Further, if $fn(C_1) \cup fn(C_2) \subset \rho_1$ then $C_1 \underset{\sim}{\sqsubset}_{\rho_2} C_2$ implies $C_1 \underset{\sim}{\sqsubset}_{\rho_1} C_2$.* □

4 An Alternate Characterization of May Testing

We give an alternate characterization of may testing which does not involve quantification over observing contexts. The characterization is trace-based, i.e. it is in terms of sequences of observable actions, namely the message exchanges, that a configuration may perform while interacting with its environment.

The set of possible message exchanges at any time is determined by the current ownership of names, i.e. which names denote actors in the configuration and which those in the environment.

The configuration can input only messages targeted to one of its actors that is not hidden from the environment (a receptionist), and can emit only messages targeted to an actor in the environment (an external actor). We call this the *encapsulation* property. Note that the information about ownership of names is in general not contained in the syntax of a configuration as internal actors may disappear (assume sink behavior) and external names may be forgotten as the configuration evolves. We therefore define the notion of a configuration interface that records the history of ownership of names, and use it to define a labeled transition system that characterizes observable actions.

4.1 Labeled Transition System and Interaction Paths

Definition 5 (interfaces) *An interface is a pair of sets of names written as* $[\rho, \chi]$, *where* $\rho \cap \chi = \emptyset$. *We let* I *range over interfaces. We define an ordering on interfaces as* $[\rho_1, \chi_1] \leq [\rho_2, \chi_2]$ *if* $\rho_1 \subset \rho_2$ *and* $\chi_1 \subset \chi_2 \cup \rho_2$. □

Lemma 4 *The relation* $\leq$ *on interfaces is a partial order.* □

We associate a configuration with interface $[\rho, \chi]$ to mean that names in ρ denote receptionists of the configuration and those in χ denote its external actors. Thus, the configuration can input only messages with target in ρ and emit messages with target in χ. Note that since the computational history of a configuration is not contained in its syntax, a configuration can have several possible interfaces. The idea behind partial order on interfaces is that if $I_1 \leq I_2$ and I_1 is a possible interface of a configuration then so is I_2.

Definition 6 *Let* $\rho; f \vdash C$, *and* $\chi = fn(C) - \rho$. *Then we say* $[\rho', \chi']$ *is a possible interface of* C *and write* $C : [\rho', \chi']$ *if* $[\rho, \chi] \leq [\rho', \chi']$. *We call* $[\rho, \chi]$ *the minimal interface of* C.

Remark*: Note that as a direct consequence of Lemma 3, if* $C_1 \equiv C_2$ *then* $C_1 : [\rho, \chi]$ *if and only if* $C_2 : [\rho, \chi]$. □

We define labeled transitions over configurations with interfaces, which are written as $\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi}$. We say $\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi}$ is well-formed if and only if $C : [\rho, \chi]$. The transition rules are given in Table 3. Transition labels can be of five forms: τ (a silent action), $\overline{x}y$ (free output of a message with target x and content y), $\overline{x}(y)$ (bound output), xy (free input of a message) and $x(y)$ (bound input). We denote the set of all visible actions (non-τ) actions by $\mathcal{L}$, and let α range over $\mathcal{L}$. The functions $fn(.)$, $bn(.)$ and $n(.)$ are defined on $\mathcal{L}$ the usual way. To have a convenient uniform notation for free and bound actions we use the following convention: $(\emptyset)\overline{x}y = \overline{x}y$, $(\{y\})\overline{x}y = \overline{x}(y)$, and similarly for input actions. We define a complementation function on $\mathcal{L}$ as $\overline{(\hat{y})xy} = (\hat{y})\overline{x}y$, $\overline{(\hat{y})\overline{x}y} = (\hat{y})xy$.

The labeled transition system is essentially a simple extension of the reduction system to include observable actions. The *IN* and *OUT* rules together capture the encapsulation property we described earlier. The *IN* rule states that a configuration can receive only a message targeted to one of its receptionists. The message is asynchronous and is added to the pool of messages in the configuration. The external actor set of the interface may expand as the received message may contain new external actor names. The *OUT* rule states that only messages targeted to an external actor can leave the configuration. The receptionist set may expand because names of hidden actors can be exported in the message. Note that fresh names are chosen for these new receptionists so that uniqueness property is preserved.

$$\begin{array}{ll}
\textit{IN:} & \langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{(\hat{y})xy} \langle\!\langle C \mid \overline{x}y\rangle\!\rangle^{\rho}_{(\chi\cup\{y\})-\rho} \quad \texttt{if } \hat{y}\cap(\rho\cup\chi)=\emptyset,\ x\in\rho \\
\textit{OUT:} & \langle\!\langle (\nu\hat{y})(C|\overline{x}y)\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{(\hat{y})\overline{x}y} \langle\!\langle C\rangle\!\rangle^{\rho\cup\hat{y}}_{\chi} \quad \texttt{if } \hat{y}\cap(\rho\cup\chi)=\emptyset,\ x\in\chi \\
\textit{TAU:} \ \dfrac{C \longrightarrow C'}{\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{\tau} \langle\!\langle C'\rangle\!\rangle^{\rho}_{\chi}} & \textit{LEQV:} \ \dfrac{\langle\!\langle C_1'\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{\alpha} \langle\!\langle C_2'\rangle\!\rangle^{\rho}_{\chi}}{\langle\!\langle C_1\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{\alpha} \langle\!\langle C_2\rangle\!\rangle^{\rho}_{\chi}} \ \texttt{if} \ \begin{array}{l} C_1\equiv C_1' \\ C_2 \equiv C_2'\end{array}
\end{array}$$

Table 3: Labelled transition system for Aπ

Lemma 5 states that the transition system is consistent with the reduction system. Theorem 3 states the soundness of the transition system and also characterizes the evolution of configuration interfaces.

Lemma 5 *If* $C : [\rho, \chi]$ *then* $C \longrightarrow C'$ *if and only if* $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{\tau} \langle\!\langle C'\rangle\!\rangle^{\rho}_{\chi}$. □

Theorem 3 (subject reduction 2) *If* $C_1 : [\rho_1, \chi_1]$ *and* $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{\alpha} \langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2}$ *then* $C_2 : [\rho_2, \chi_2]$, $\rho_1 \subset \rho_2$ *and* $\chi_1 \subset \chi_2$.

Remark*: Note that* $[\rho_1, \chi_1] \leq [\rho_2, \chi_2]$. □

We let s, r, t range over $\mathcal{L}^*$. For $s = \alpha_1 \ldots \alpha_i \ldots \alpha_n$, we define $len(s) = n$, and $s(i) = \alpha_i$, for $1 \leq i \leq len(s)$. The functions $fn(.), bn(.)$ and $n(.)$ are defined on $\mathcal{L}^*$ the obvious way. The complementation function on $\mathcal{L}$ is extended to $\mathcal{L}^*$ the obvious way. We use $\Longrightarrow$ to denote the reflexive transitive closure of $\xrightarrow{\tau}$, and $\xRightarrow{\alpha}$ to denote $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$. Note that $\Longrightarrow$ is overloaded to denote both sequences of reductions and $\xrightarrow{\tau}$ transitions, but its context of use will always clarify which one is being used. For $s = l.s'$ we use $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{s} \langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2}$ to denote $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{l}\xrightarrow{s'} \langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2}$, and similarly $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xRightarrow{s} \langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2}$ to denote $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xRightarrow{l}\xRightarrow{s'} \langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2}$. We write $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xRightarrow{s}$ if $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xRightarrow{s} \langle\!\langle C'\rangle\!\rangle^{\rho'}_{\chi'}$ for some C', ρ', χ', and similarly for $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{s}$ and $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xrightarrow{\tau}$.

The sequences of observable actions, called *interaction paths*, that a configuration C with interface $[\rho, \chi]$ may perform are precisely $s \in \mathcal{L}^*$ such that $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi} \xRightarrow{s}$. The following lemma, which is true in Aπ, relates a computation involving two composed configurations to the interaction paths that each exhibits during the computation.

Lemma 6 (zip-unzip) *Let* $C_1 : [\rho_1, \chi_1], C_2 : [\rho_2, \chi_2]$, *and* $\rho_1 \cap \rho_2 = \emptyset$. *Then* $C_1|C_2 \Longrightarrow C$ *if and only if for some* s, $\langle\!\langle C_1\rangle\!\rangle^{\rho_1}_{\chi_1} \xRightarrow{s} \langle\!\langle C_1'\rangle\!\rangle^{\rho_1'}_{\chi_1'}$, $\langle\!\langle C_2\rangle\!\rangle^{\rho_2}_{\chi_2} \xRightarrow{\overline{s}} \langle\!\langle C_2'\rangle\!\rangle^{\rho_2'}_{\chi_2'}$ *and* $C \equiv (\nu\tilde{z})(C_1'|C_2')$, *where* $\{\tilde{z}\} = bn(s)$. □

4.2 The Alternate Characterization

We present an alternate characterization of may testing in Aπ that is based on interaction paths. We follow the same approach used for asynchronous π-calculus by Boreale [11]. However, differences between the two calculi, such as in name matching capabilities and notions of observability, lead to changes in the characterization and require different proofs to establish it.

To demonstrate these differences, we follow Boreale's proof layout and highlight the differences as they arise.

Definition 7 and Lemma 7 demonstrate the relation between interaction paths exhibited by a configuration and the evolution of its interface.

Definition 7 *We define the following functions on interaction paths*

$$\begin{array}{ll}
rcp([\rho,\chi],\epsilon)=\rho & ext([\rho,\chi],\epsilon)=\chi \\
rcp([\rho,\chi],s.(\hat{y})\overline{x}y)=\hat{y}\cup rcp([\rho,\chi],s) & ext([\rho,\chi],s.(\hat{y})\overline{x}y)=ext([\rho,\chi],s) \\
rcp([\rho,\chi],s.(\hat{y})xy)=rcp([\rho,\chi],s) & ext([\rho,\chi],s.(\hat{y})xy)=(\{y\}\cup ext([\rho,\chi],s)) \\
 & \qquad -rcp([\rho,\chi],s)
\end{array}$$

□

Lemma 7 *If $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi}\overset{s}{\Longrightarrow}\langle\!\langle C'\rangle\!\rangle^{\rho'}_{\chi'}$ then*

1. *$\rho'=rcp([\rho,\chi],s)$ and $\chi'=ext([\rho,\chi],s)$,*
2. *$s=s_1.(\hat{y})xy.s_2$ implies $x\in rcp([\rho,\chi],s_1)$, and $\hat{y}\cap(rcp([\rho,\chi],s_1)\cup ext([\rho,\chi],s_1))=\emptyset$.*
3. *$s=s_1.(\hat{y})\overline{x}y.s_2$ implies $x\in ext([\rho,\chi],s_1)$, and $y\in rcp([\rho,\chi],s_1)\cup ext([\rho,\chi],s_1)$ if and only if $\hat{y}=\emptyset$.*
4. *$rcp([\rho,\chi],s)\cup ext([\rho,\chi],s)=n(s)\cup\rho\cup\chi$, and*
5. *$s=s_1.\alpha.s_2$ implies $bn(\alpha)\cap(n(s_1)\cup\rho\cup\chi)=\emptyset$.* □

For $\rho\cap\chi=\emptyset$, we define $\mathcal{L}^*[\rho,\chi]$ as the set of all $s\in\mathcal{L}^*$ that satisfy conditions 2 and 3 of Lemma 7. We define alpha equivalence on paths the obvious way, and work modulo alpha equivalence. Note that $\mathcal{L}^*[\rho,\chi]$ is not closed under alpha renaming, that is for $s\in\mathcal{L}^*[\rho,\chi]$ there may be $r\equiv_\alpha s$ but $r\notin\mathcal{L}^*[\rho,\chi]$. Therefore, we will only consider alpha renaming that does not result in such ill-formed paths

Definition 8 (path transformation) *We define a relation $\preceq$ on $\mathcal{L}^*$ as the reflexive transitive closure of the relation defined in Table 4.* □

The intuition behind laws in Table 4, which is stated formally in Lemma 8, is that, for $r,s\in\mathcal{L}^*[\rho,\chi]$ and $r\prec s$, if a configuration exhibits $\overline{s}$ then it can also exhibit $\overline{r}$. *L3* states that two consecutive outputs can be commuted, while *L4* states that two consecutive inputs can be commuted. *L5* states that an output can be postponed to after an input, provided the input doesn't use the bound name exported by the output. *L3* and *L5* can be used to postpone outputs, and *L4* and *L5* to prepone inputs. These two rules capture the essence of asynchrony. *L1* states that additional inputs may be appended, and *L2* states that a tailing output can be removed as there is no interaction after it that depends on it.

Lemma 8 *If $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi}\overset{\overline{s}}{\Longrightarrow}$, $r\preceq s$ and $\overline{r}\in\mathcal{L}^*[\rho,\chi]$, then $\langle\!\langle C\rangle\!\rangle^{\rho}_{\chi}\overset{\overline{r}}{\Longrightarrow}$.* □

(L1)	$s.(\hat{y})\overline{x}y$	$\prec$	s			
(L2)	s	$\prec$	$s.(\hat{y})xy$			
(L3)	$s_1.(M)uv.(N)xy.s_2$	$\prec$	$s_1.(\hat{y})xy.(\hat{v})uv.s_2$	`where`	$M = \hat{v},\ N = \hat{y}$	`if` $v \notin \hat{y}$
					$M = \hat{y},\ N = \emptyset$	`otherwise`
(L4)	$s_1.(M)\overline{u}v.(N)\overline{x}y.s_2$	$\prec$	$s_1.(\hat{y})\overline{x}y.(\hat{v})\overline{u}v.s_2$	`where`	$M = \hat{v},\ N = \hat{y}$	`if` $v \notin \hat{y}$
					$M = \hat{y},\ N = \emptyset$	`otherwise`
(L5)	$s_1.(\hat{v})\overline{u}v.(\hat{y})xy.s_2$	$\prec$	$s_1.(\hat{y})xy.(\hat{v})\overline{u}v.s_2$	`if` $u, v \notin \hat{y}$		

Table 4: A relation on interaction paths.

We now compare our laws with those of Boreale. The mismatch capability in Aπ enables distinguishing bound names from free names. Thus, Boreale's law that allows replacing bound names in an input action with free names is not applicable in Aπ. Furthermore, Boreale's *annihilation* law, which states that a configuration can consume a pair of complementary interactions, is not needed in Aπ for two reasons. First, due to the encapsulation property a configuration can never exhibit complementary actions. Second, because we do not have a law that substitutes bound names with fresh names, no path with complementary actions is related to a path in $\mathcal{L}^*[\rho, \chi]$. For the variants without mismatch capability, the two laws above will be needed (see Section 5). Finally, as opposed to asynchronous inputs allowed by the *IN* rule, Boreale's LTS uses synchronous inputs. As a consequence, *L4* is not applicable there. These differences lead to a stronger characterization of may preorder for Aπ (see below).

Using $\preceq$ we define the following preorder on configurations, which we will prove to be an alternate characterization of may preorder.

Definition 9 *Let $[\rho_1, \chi_1]$ be the minimal interface of C_1 and $[\rho_2, \chi_2]$ that of C_2. For ρ such that $\rho_1, \rho_2 \subset \rho$, we say $C_1 \ll_\rho C_2$ if for $\chi = (\chi_1 \cup \chi_2) - \rho$, $\langle\!\langle C_1 \rangle\!\rangle^\rho_\chi \overset{s}{\Longrightarrow}$ implies $\langle\!\langle C_2 \rangle\!\rangle^\rho_\chi \overset{r}{\Longrightarrow}$ for some $r \preceq s$.* □

Although it is easy to see that $C_1 \ll_\rho C_2$ implies $C_1 \mathrel{\overset{\sqsubset}{\sim}}_\rho C_2$, the reverse direction is more involved. To prove the reverse direction, we construct for a given $s \in \mathcal{L}^*[\rho, \chi]$, an observer O such that for $C : [\rho, \chi]$, if C $\underline{may}$ O then $\langle\!\langle C \rangle\!\rangle^\rho_\chi \overset{r}{\Longrightarrow}$ for some $r \preceq s$.

Definition 10 (canonical observer) *For $s \in \mathcal{L}^*[\rho, \chi]$, we define an observer*

$$O([\rho, \chi], s) = (\nu \tilde{x}, z)(|_{x_i \in ext([\rho,\chi],s)} Proxy(s, x_i, z) \mid O'([\rho, \chi], s, z)), \text{ where}$$

$$\{\tilde{x}\} = bn(s) - rcp([\rho, \chi], s)$$

$$O'([\rho, \chi], \epsilon, z) \overset{\triangle}{=} \overline{\mu}$$

$$O'([\rho, \chi], (\hat{y})xy.s, z) \overset{\triangle}{=} \overline{x}y | O'([\rho, \chi \cup \{y\} - \rho], s, z)$$

$$O'([\rho, \chi], \overline{x}y.s, z) \overset{\triangle}{=} z(u, v).[u = x \wedge v = y](O'([\rho, \chi], s, z), 0) \qquad v, u \text{ fresh}$$

$$O'([\rho, \chi], \overline{x}(y).s, z) \overset{\triangle}{=} z(u, y).[u = x \wedge y \notin (\rho \cup \chi)](O'([\rho \cup \{y\}, \chi], s, z), 0) \qquad u \text{ fresh}$$

$$Proxy(\epsilon, x, z) = 0$$

$$Proxy((\hat{y})xy.s, x, z) \overset{\triangle}{=} Proxy(s, x, z)$$

$$Proxy((\hat{y})\overline{x}y.s, x, z) \overset{\triangle}{=} x(v).(\overline{z}\langle x, v\rangle \mid Proxy(s, x, z)) \qquad v \text{ fresh}$$

In the above, $\stackrel{\triangle}{=}$ is used for macro definitions. $\square$

The observer $O([\rho,\chi],s)$ consists of a collection of proxies and a central matcher. There is one forwarding proxy for each external name a configuration C knows while doing s. This forwarding mechanism, which is absent in Boreale's construction, is essential in our case because of uniqueness of actor names. The matcher which analyzes the forwarded messages, keeps track of names in the "current" interface of C and uses them to distinguish bound names from free names in outputs. This technique works because if $(\hat{y})\overline{x}y.s \in \mathcal{L}^*[\rho,\chi]$ and $y \notin \rho \cup \chi$ then $\hat{y} = \{y\}$. The abbreviations $\notin$ and $\wedge$ used in the definition can be encoded using the conditional construct. The encoding of $\notin$ requires the ability to mismatch names. Note that the definition also uses polyadic communication between proxies and matcher, whose encoding was shown in Section 2.

We not only require a different construction for the canonical observer than Boreale's, but also an essentially different argument for establishing Lemma 9 (see Appendix for proof). For $s \in \mathcal{L}^*[\rho,\chi]$, let $\mathit{ffi}([\rho,\chi],s)$ be the set of all names y such that s can be written as $s_1.xy.s_2$ and $y \notin \mathit{rcp}([\rho,\chi],s_1) \cup \mathit{ext}([\rho,\chi],s_1)$. It is easy to show that if $s \in \mathcal{L}^*[\rho,\chi]$ and $\chi' = \chi \cup \mathit{ffi}([\rho,\chi],s)$, then $O([\rho,\chi],s) : [\chi',\rho]$.

Lemma 9 *Let $r,s \in \mathcal{L}^*[\rho,\chi]$ and $\chi' = \chi \cup \mathit{ffi}([\rho,\chi],s)$. Then $\langle\!\langle O([\rho,\chi],s)\rangle\!\rangle_{\rho}^{\chi'} \stackrel{\overline{r}.\overline{\mu}\mu}{\Longrightarrow}$ implies $r \preceq s$.*
$\square$

Following is the alternate characterization of may preorder.

Theorem 4 *$C_1 \underset{\sim}{\sqsubset}_\rho C_2$ if and only if $C_1 \ll_\rho C_2$.* $\square$

This alternate characterization can be further strengthened to set inclusion in the case of Aπ. This is a consequence of Lemma 10 which is not true in Boreale's setting. Note that Theorem 5 renders the interaction-path preorder to a proof tool rather than a part of the characterization.

Lemma 10 *Let $r,s \in \mathcal{L}^*[\rho,\chi]$. Then $r \prec s$ implies $\overline{s} \prec \overline{r}$.* $\square$

Theorem 5 *Let $[\rho_1,\chi_1]$ be the minimal interface of C_1 and $[\rho_2,\chi_2]$ that of C_2. For ρ such that $\rho_1,\rho_2 \subset \rho$, if $C_1 \ll_\rho C_2$ and $\chi = (\chi_1 \cup \chi_2) - \rho$, then $\langle\!\langle C_1\rangle\!\rangle_{\chi}^{\rho} \stackrel{s}{\Longrightarrow}$ implies $\langle\!\langle C_2\rangle\!\rangle_{\chi}^{\rho} \stackrel{s}{\Longrightarrow}$.* $\square$

5 Variants of Aπ

We now present the two variants of Aπ without mismatch. First, we consider the restricted version which allows comparison of a received name with only a local name. For this variant, the *COND* rule is replaced by

$$\mathit{CASE:}\quad \frac{\forall 1 \le i \le n \quad \rho_i; f_i \vdash C_i}{(\cup_i(\rho_i \cup y_i)); f \vdash \texttt{case}\ x\ \texttt{of}\ (y_1 : C_1, \ldots, y_n : C_n)} \quad \texttt{if}\ \begin{array}{l} y_i \neq y_j\ \texttt{for}\ i \neq j,\ \texttt{and} \\ f_i\ \texttt{are mutually compatible} \end{array}$$

where

$$f(x) = \begin{cases} (f_1 \oplus f_2 \oplus \ldots \oplus f_n)(x) & \texttt{if } x \in \cup_i \rho_i \\ \bot & \texttt{otherwise}. \end{cases}$$

Note that unlike in *COND*, y_i's are included in the set of actors of the resulting configuration, thus making them local. This ensures that y_i's are constants because by the *ACT* rule they cannot be bound by input prefixes, and hence cannot be received names. The `case` construct can be seen as a macro for the following π-calculus term.

$$[\![\texttt{case } x \texttt{ of } (y_1 : C_1, \ldots y_n : C_n)]\!] = [x = y_1][\![C_1]\!] | \ldots | [x = y_n][\![C_n]\!]$$

We can not use this translation directly in Aπ instead of the `case` construct because it need not be well-typed. Although only one of the C_i's is activated because y_i's are distinct, the type system is not clever enough to accept the translation. Specifically, since more than one C_i may contain the same actor, the *COMP* rule could be violated. It is possible to enhance the type system to accept such terms; but that would make the type system significantly complex. We therefore adopt the simpler approach of using a multi-branching `case` statement.

For the variant with general match that allows comparison of any two names, the *COND* rule is replaced with

$$\textit{MATCH:} \quad \frac{\forall 1 \le i \le n \quad \rho_i; f_i \vdash C_i}{(\cup_i \rho_i); (f_1 \oplus f_2 \oplus \ldots \oplus f_n) \vdash \texttt{if } x \texttt{ is } (y_1 : C_1, \ldots, y_n : C_n)} \quad \texttt{if } f_i \texttt{ are mutually compatible}$$

Unlike in `case`, y_i's can be received names and therefore, it is possible that more than one branch is true during the match. In such cases, one of the branches is non-deterministically chosen. Thus, we deviate slightly from the Actor model by allowing actor behaviors to be non-deterministic. But this is only internal non-determinism because the choice can not be influenced by external interactions. In fact, the `if` construct can be seen as a macro for the following π-calculus term that does not involve the choice operator.

$$\begin{aligned} &[\![\texttt{if } x \texttt{ is } (y_1 : C_1, \ldots, y_n : C_n)]\!] = \\ &\qquad (\nu u, v_1, \ldots, v_n)([x = y_1]\overline{u}v_1 | \ldots | [x = y_n]\overline{u}v_n | \\ &\qquad\qquad u(w).[\![(\texttt{case } w \texttt{ of } (v_1 : C_1, \ldots, v_n : C_n))]\!]) \qquad u, v_i, w \text{ fresh} \end{aligned}$$

Note that this encoding can not be used directly in Aπ instead of `if`, because it will not type check when $\cup \rho_i$ contains more than one element (violates the *ACT* rule). For the same reason given for `case` we need a new construct to simplify the type system. Lemmas 1 and 2 are true for For both the variants of Aπ.

For reduction semantics of the variants, the structural congruence rules of Aπ are left unchanged, but the reduction rules are changed by replacing *IF* and *ELSE* rules with one of the following:

$$\begin{array}{lll} \textit{BCASE:} & \texttt{case } x \texttt{ of } (y_1 : C_1, \ldots, y_n : C_n) \longrightarrow C_i & \texttt{if } x = y_i \quad \texttt{and} \\ \textit{BMATCH:} & \texttt{if } x \texttt{ of } (y_1 : C_1, \ldots, y_n : C_n) \longrightarrow C_i & \texttt{if } x = y_i \end{array}$$

Both Lemma 3 and Theorem 1 hold for the variants. The definition of may testing for variants is the same as that for Aπ, and Theorem 2 holds for both. The labeled transition system for variants is also the same, and all lemmas and theorems in Section 4.1 hold in both.

For the alternate characterization of may testing in the variant with general match, we need to weaken the trace preorder (relate more paths) by adding some laws to Table 4. Since without mismatch capability an observer cannot fully discriminate between free and bound outputs (of the configuration it observes), we need the following law.

$$\textbf{(L6)} \quad s_1.\overline{x}w.(s_2\{w/y\}) \;\prec\; s_1.\overline{x}(y).s_2$$

Note that substitutions such as the above may lead to internalization of messages. Specifically, if an observer receives the name of one of its own actors instead of a fresh name, the messages that the observer sends to the argument will now be internalized and can not be consumed by the environment. Furthermore, these internalized messages can themselves be consumed by the observer in a successful computation. We account for these possibilities by a new law.

$$\textbf{(L7)} \quad s_1.(\hat{y})s_2 \;\prec\; s_1.(\hat{y})xy.\overline{x}y.s_2 \qquad \texttt{if } (\hat{y})s_2 \texttt{ is defined}$$

where $(\hat{y})s_2$ is s_2 if $\hat{y}$ is empty, and otherwise is the path obtained from s_2 by binding the first free occurence of y as the argument of an input action. If the first free occurence of y is not the argument of an input action then $(\hat{y})s_2$ is undefined. With these new laws, Theorem 4 holds for the variant with $\ll_\rho$ defined as in Definition 9. However, Lemma 10 and Theorem 5 do not hold. We note that laws **L6** and **L7** also appear in the alternate characterization of may testing for asynchronous π-calculus with match operator [11]. In fact, the characterizations for the two calculi are essentially the same.

For the variant with restricted match, the usual approach for characterization does not work for several reasons. First, since an observer can only match a name against its local names, it cannot fully discriminiate between outputs containing local names and outputs containing non-local names. Thus, in contrast to *L6*, any output argument that is not a local name (not just bound names) can be substituted with arbitrary names. Second, different such outputs with the same argument can be substituted with different names, because the observer can not compare two received names with each other. Furthermore, since different observers can use names they receive in different ways to send messages, the result of such general substitutions is one of many possible paths depending on the data flow in the observer's computation.

We demonstrate the need for a different approach to characterization through an example. Consider $C_1 = (\nu u)(\overline{x}u|\overline{y}u|u(w).\overline{w}w)$ that can exhibit $s = \overline{x}(u).\overline{y}u.u(w).\overline{w}w$. The following observers can be satisfied by s:

$$\begin{array}{rcl} O_1 & = & (\nu w)(x(t).\overline{t}w|y(t').0|w(v).[v=w]\overline{\mu}\mu) \\ O_2 & = & (\nu w)(x(t).0|y(t').\overline{t'}w|w(v).[v=w]\overline{\mu}\mu) \end{array}$$

Let $C_2 = (\nu v, u_1, u_2)(v(t).v(t').(\overline{x}t|\overline{y}t')|\overline{v}u_1|\overline{v}u_2|u_1(w).\overline{w}w)$. Now, C_2 can satisfy O_1 with $r_1 = \overline{x}(u_1).\overline{y}(u_2).u_1(w).\overline{w}w$, and O_2 with $r_2 = \overline{x}(u_2).\overline{y}(u_1).u_1(w).\overline{w}w$. But, it cannot exhibit a single path that can satisfy both O_1 and O_2. In fact, it is the case that $C_1 \mathrel{\overset{\sqsubset}{\sim}}_\emptyset C_2$. This example shows that the alternate characterization of $C_1 \mathrel{\overset{\sqsubset}{\sim}}_\rho C_2$ should only require that, for a given path s that C_1 exhibits, C_2 can exhibit a set of paths P such that if s satisfies an observer O then there is a path $r \in P$ that can satisfy O. The choice of $r \in P$ depends on the dataflow in a successful computation of O. In comparison, the characterizations for the other two variants imposed the additional constraint that P is a singleton, which is too strong for the variant with restricted match.

We can formalize the concepts above through the notions of *templates* and *matches* relation. A template t° of path s represents the dataflow of non-local names in an observer that exhibits $\overline{s}$. The template explicit represents the dependencies from outputs in s with non-local names as arguments (inputs received by the observer) to names (targets and arguments) in input actions of s (outputs emitted by the observer). Clearly, there can be several possible templates of a path. We say a path s matches template t° if it can satisfy an observer that exhibits the dataflow t° during a successful computation. For the alternate characterization, we first define for a path s and set of names ρ, the set $T(s, \rho)$ as the set of all possible templates that can be obtained from s, assuming names in ρ as non-local. This covers all possible dataflows in any observer's computation while exhibiting $\overline{s}.\overline{\mu}\mu$. We say $C_1 \ll_\rho C_2$, if for each path s that C_1 can exhibit, C_2 can exhibit a set of paths P such that for any template $t^\circ \in T(s, \rho)$ there is a path r in P such that r matches t°. Formal definition of these concepts and proof that this characterization is correct is omitted here due to space constraints.

6 Discussion and Related Work

A possible direction of future work is to give a complete axiomatization for finite configurations, i.e. configurations that do not use recursive behavior definitions. It is also interesting to see if our approach for alternate characterization of may testing for Aπ with restricted match, can be also be applied to get a characterization for Lπ [13].

We have not considered fairness property of the Actor model in this paper as it does not affect the notion of may testing. May testing is concerned only with the occurrence of an event after a finite computation, while fairness requires eventual delivery of messages, thereby affecting only potentially infinite computations. The reader is referred to [18], where a version of Aπ with fairness is presented. An interesting consequence of fairness is that must equivalence implies may equivalence, which was shown for a specific Actor based language in [2]. It can be shown by a similar argument that this result holds in Aπ also.

Several calculi [5, 6, 10, 16] and programming languages [2, 5] have been proposed for actors. Since these works were motivated by different reasons, such as design of high-level languages or type systems for certain generic problems in object-oriented languages, their systems are not faithful to the pure Actor model [1]. For instance, they either are equipped with high level programming constructs that are not intrinsic to actors, or ignore actor properties such as uniqueness and persistence. Furthermore, these systems are not directly comparable to π-calculus. In contrast, our aim was to investigate a theory for the pure Actor model and compare it with that of π-calculus and its variants.

Notions of equivalence and semantic models have been studied for actors, such as asynchronous bisimulation [6], testing equivalences [2], event diagrams [4], and interaction paths [17]. We have not only related may testing [2] to the interaction paths model [17], but also related our characterizations to that of asynchronous π-calculus given in [11].

References

[1] G. Agha. *Actors: A Model of Concurrent Computation in Distributed Systems.* MIT Press, 1986.

[2] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *Journal of Functional Programming*, 7:1–72, 1997.

[3] G. Boudol. Asynchrony and the π-Calculus. Technical Report 1702, Department of Computer Science, Inria Univeristy, May 1992.

[4] W.D. Clinger. *Foundations of Actor Semantics.* PhD thesis, Massachusetts Institute of Technology, AI Laboratory, 1981.

[5] F.Dagnat, M.Pantel, M.Colin, and P.Sallé. Typing concurrent objects and actors. In *L'Objet – Mthodes formelles pour les objets (L'OBJET)*, volume 6, pages 83–106, 2000.

[6] M. Gaspari and G. Zavattaro. An Algebra of Actors. Technical Report UBLCS-97-4, Department of Computer Science, Univeristy of Bologna (Italy), May 1997.

[7] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[8] K. Honda and M. Tokoro. An Object Calculus for Asynchronous Communication. In *Fifth European Conference on Object-Oriented Programming*, July 1991. LNCS 512, 1991.

[9] J-L.Colaço, M.Pantel, F.Dagnat, and P.Sallé. Safety analysis for non-uniform service availability in actors. In *Formal Methods for Open Object-based Distributed Systems*, February 1999.

[10] J-L.Colaço, M.Pantel, and P.Sallé. Analyse de linéarité par typage dans un calcul d'acteurs primitifs. In *Actes des Journées Francophones des Langages Applicatifs (JFLA)*, 1997.

[11] R. Pugliese M. Boreale, R. De Nicola. A theory of may testing for asynchronous languages. In *Foundations of Software Science and Computation Structures*, pages 165–179, 1999. LNCS 1578.

[12] M.Boreale and R. De Nicola. Testing equivalence for mobile systems. In *Information and Computation*, volume 120, pages 279–302, 1995.

[13] M. Merro and D. Sangiorgi. On Asynchrony in Name-Passing Calculi. In *Proceeding of ICALP '98*. Springer-Verlag, 1998. LNCS 1443.

[14] R. Milner, J. Parrow, and D. Walker. A calculus of Mobile Processes, Part I. Technical Report ECS-LFCS-89-85, Department of Computer Science, Edinburgh University, June 1989.

[15] B. C. Pierce and D. Sangiorgi. Typing and Subtyping for Mobile Processes. *Journal of Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

[16] A. Ravara and V. Vasconcelos. Typing non-uniform concurrent objects. In *CONCUR*, pages 474–488, 2000. LNCS 1877.

[17] C. Talcott. Composable semantic models for actor theories. *Higher-Order and Symbolic Computation*, 11(3), 1998.

[18] Prasannaa Thati. Towards an Algebraic Formulation of Actors. Master's thesis, Computer Science, University of Illinois at Urbana Champaign, 2000.

A Appendix

Proof of Lemma 1: By structural induction on C. $\square$

It is straightforward to show that Lemma 1 also holds in both variants of Aπ. The type system (including that of the variants) respects alpha equivalence , i.e. if $C_1 \equiv_\alpha C_2$ then $\rho; f \vdash C_1$ if and only if $\rho; f \vdash C_2$. This can be proved by induction on the length of a derivation of $C_1 \equiv_\alpha C_2$, and is left to the reader.

Proof of Lemma 2: Since the type system respects alpha equivalence, without loss of generality, we may assume the hygiene condition that $\sigma(x) = x$ for all $x \in bn(C)$, and $bn(C) \cap \sigma(fn(C)) = \emptyset$.

The proof is by induction on the length of a derivation of $\rho; f \vdash C$. It is straightforward to verify the base cases where the derivation is a direct application of *NIL, MSG* or *INST*. For the induction step, we consider only two cases; the others are simple.

1. $C = x(y).C'$: Then the last step of derivation is

$$\textit{ACT:}\quad \frac{\rho'; f' \vdash C'}{\{x\} \cup \hat{z}; ch(x, \hat{z}) \vdash x(y).C'} \quad \texttt{if} \quad \begin{array}{l} \rho' - \{x\} = \hat{z},\ y \notin \rho',\ \texttt{and} \\ f' = \begin{cases} ch(x, \hat{z}) & \texttt{if } x \in \rho' \\ ch(\epsilon, \hat{z}) & \text{otherwise} \end{cases} \end{array}$$

   Note that by hygiene condition $C\sigma = \sigma(x)(y).(C'\sigma)$. Since σ is an injection on $\{x\} \cup \hat{z}$ so it is on ρ', and thus by induction hypothesis $\sigma(\rho'); f'\sigma \vdash C'\sigma$. This, together with $\rho' - \{x\} = \hat{z}$, also implies $\sigma(\rho') - \sigma(\{x\}) = \sigma(\hat{z})$. By Lemma 1, we have $\rho' \subset fn(C')$, and hence by the hygiene condition $y \notin \sigma(\rho')$. Since σ is an injection on $\{x\} \cup \hat{z}$ and $\rho' \subset \{x\} \cup \hat{z}$, we have $f'\sigma = ch(x, \hat{z})\sigma = ch(\sigma(x), \sigma(\hat{z}))$ if $\sigma(x) \in \sigma(\rho')$, and $ch(\epsilon, \hat{z})\sigma = ch(\epsilon, \sigma(\hat{z}))$ otherwise. We can now apply the *ACT* rule to get
   $\sigma(\{x\}) \cup \sigma(\hat{z}); ch(\sigma(x), \sigma(\hat{z})) \vdash \sigma(x)(y).(C'\sigma)$, i.e. $\sigma(\{x\} \cup \hat{z}); ch(x, \hat{z})\sigma \vdash \sigma(x)(y).(C'\sigma)$.

2. $C = [x = y](C_1, C_2)$: Then the last step of the derivation is

$$\textit{COND:}\quad \frac{\rho_1; f_1 \vdash C_1 \quad \rho_2; f_2 \vdash C_2}{\rho_1 \cup \rho_2; f_1 \oplus f_2 \vdash [x = y](C_1, C_2)} \quad \texttt{if } f_1 \texttt{ and } f_2 \texttt{ are compatible}$$

   Since σ is an injection on $\rho_1 \cup \rho_2$, so it is on ρ_1 and ρ_2. Then, by induction hypothesis $\sigma(\rho_1); f_1\sigma \vdash C_1\sigma$ and $\sigma(\rho_2); f_2\sigma \vdash C_2\sigma$. The reader may verify that the facts f_1 and f_2 are compatible, and σ is an injection on $\rho_1 \cup \rho_2$, together imply $f_1\sigma$ and $f_2\sigma$ are compatible. We can now apply the *COND* rule to get $\sigma(\rho_1) \cup \sigma(\rho_2); f_1\sigma \oplus f_2\sigma \vdash [\sigma(x) = \sigma(y)](C_1\sigma | C_2\sigma)$. The result follows from the following fact that $f_1\sigma \oplus f_2\sigma = (f_1 \oplus f_2)\sigma$, which can be verified easily. $\square$

It is easy to show that Lemma 2 also holds for both variants of Aπ.

Proof of Lemma 3: The proof for $fn(C_1) = fn(C_2)$ is easy and is left to the reader. Now, since $\equiv$ is an equivalence relation we are done if we show that $\rho; f \vdash C_1$ implies $\rho; f \vdash C_2$. The proof is by induction on the length of a derivation of $C_1 \equiv C_2$.

For the base case, the derivation is a direct application of one of the 5 laws in Definition 3. We consider each in order.

1. $C_1 \equiv_\alpha C_2$: We have seen that the type system respects alpha equivalence.
2. $'|'$ is commutative and associative with 0 as the identity: The lemma follows from the fact that $\oplus$ is associative and commutative over mutually compatible functions, and has $\{\}$ as the identity.
3. The argument is simple and is left to the reader.
4. For some C_1', C_2', we have $C_1 = (\nu x)C_1'|C_2'$, $C_2 = (\nu x)(C_1'|C_2')$, and $x \notin fn(C_2')$. For some ρ_1, ρ_2, f_1, f_2, we have $\rho_1; f_1 \vdash C_1'$ and $; f_2 \vdash C_2'$. Then by *RES* and *COMP* rules, $(\rho_1 - \{x\}) \cap \rho_2 = \emptyset$, and $\rho = (\rho_1 - \{x\}) \cup \rho_2$, $f = f_1|(\rho_1 - \{x\}) \oplus f_2$. By Lemma 1, $x \notin \rho_2$, and hence $\rho_1 \cap \rho_2 = \emptyset$, $\rho = (\rho_1 \cup \rho_2) - \{x\}$, and $f = (f_1 \oplus f_2)|(\rho_1 \cup \rho_2 - \{x\})$. Then by *COMP* and *RES* rules we have $\rho; f \vdash C_2$.
5. For some $B \overset{def}{=} (\tilde{x}; \tilde{y})x_1(z).C$, and $\tilde{u}, \tilde{v}$, we have $C_1 = B\langle\tilde{u}; \tilde{v}\rangle$ and $C_2 = (x_1(z).C)\sigma$ where $\sigma = \{\tilde{u}, \tilde{v}/\tilde{x}, \tilde{y}\}$. By *INST* rule $\{\tilde{u}\}; ch(\tilde{u}) \vdash B\langle\tilde{u}; \tilde{v}\rangle$. Since the behavior definition B is well typed, we also have $\{\tilde{x}\}; ch(\tilde{x}) \vdash x_1(z).C$. Since $len(\tilde{u}) = 2$ implies $u_1 \neq u_2$, σ is an injection on $\{\tilde{x}\}$. Then by Lemma 2, we get $\sigma(\{\tilde{x}\}); ch(\tilde{x})\sigma \vdash (x_1(z).C)\sigma$, i.e $\{\tilde{u}\}; ch(\tilde{u}) \vdash (x_1(z).C)\sigma$, since $ch(\tilde{u}) = ch(\tilde{x})\sigma$.

For the induction step, the last step of the derivation is a congruence law. This cases is simple and is left to the reader. □

Proof of Theorem 1: The proof is by induction on length of a derivation of $C \longrightarrow C'$. There are three base cases, depending on whether the derivation is a direct application of *RECV, IF* or *ELSE* rules. We only consider the *RECV* and *ELSE* cases.

1. *RECV*: Then we have $C = x(y).C_1|\overline{x}z$ and $C' = C_1\{z/y\}$, for some x, y, z, C_1. Then by *ACT*, *MSG*, and *COMP* rules, for some ρ_1, f_1 we have $\rho_1; f_1 \vdash C_1$, $y \notin \rho_1$, $\rho = \rho_1 \cup \{x\}$, and $f_1 = f|\rho_1$. Then $\sigma = \{z/y\}$ is an injection on ρ_1, and by Lemma 2 it follows that $\sigma(\rho_1); f_1\sigma \vdash C_1\sigma$. But since $y \notin \rho_1$, we have $\sigma(\rho_1) = \rho_1$, $f_1\sigma = f_1$. Thus, we have $\rho_1; f_1 \vdash C'$, and the theorem follows from the fact that $\rho_1 \subset \rho$, and $f_1(x) = f(x)$ for $x \in \rho_1$.
2. *ELSE*: For some x, C_1, C_2, we have $C = [x = x](C_1, C_2)$ and $C' = C_2$. By *COND* rule, $\rho_1; f_1 \vdash C_1$, $\rho_2; f_2 \vdash C_2$, and $\rho = \rho_1 \cup \rho_2, f = f_1 \oplus f_2$, and f_1, f_2 are compatible. From the definition of $\oplus$ it is clear that $f_2(y) = \perp$ if $f(y) = \perp$. Since, f_1, f_2 are compatible, we have $f_2(y) \in \{f(y), \perp\}$ if $f(y) \neq \perp$, because otherwise $(f_1 \oplus f_2)(y) \neq (f_2 \oplus f_1)(y)$, and f_1, f_2 will not be compatible. The theorem thus follows.

For the induction step, there are three cases depending on which rule is used in the last step of the derivation.

1. *HIDE*: For some x, C_1, C_1', we have $C = (\nu x)C_1$ and $C' = (\nu x)C_1'$ and the last derivation step is

$$\mathit{HIDE}: \ \frac{C_1 \longrightarrow C_1'}{(\nu x)C_1 \longrightarrow (\nu x)C_1'}$$

   By *RES* rule, for some ρ_1, f_1, we have $\rho_1; f_1 \vdash C_1$, $\rho = \rho_1 - \{x\}$, and $f = f_1|\rho$. By induction hypothesis, $\rho_1'; f_1' \vdash C_1'$ for some $\rho_1' \subset \rho_1$, f_1'. Then by *RES* rule $\rho_1' - \{x\}; f_1'|(\rho_1' - \{x\}) \vdash (\nu x)C_1'$. Note that $\rho_1' - \{x\} \subset \rho_1 - \{x\} = \rho$. Now, the result follows if we show that $f_1'|(\rho_1' - \{x\})$ satisfies the conditions in theorem statement. For $y \in \rho_1' - \{x\}$ we consider only the case where $f(y) = *$; the other cases where $f(y) = \perp$ and $f(y) \in \rho$ are easier. Since $f = f_1|(\rho_1 - \{x\})$, it follows that $f_1(y) \in \{x, *\}$. We consider the case where $f_1(y) = x$ and leave the other easier case to the reader. By induction hypothesis, $f_1'(y) \in \{x, \perp\}$, and hence $(f_1'|(\rho_1' - x))(y) \in \{*, \perp\}$, and the theorem follows.

2. *PAR*: For some C_1, C_1', C_2, we have $C = C_1|C_2$, $C' = C_1'|C_2$ and the last derivation step is

$$\mathit{PAR}: \ \frac{C_1 \longrightarrow C_1'}{C_1|C_2 \longrightarrow C_1'|C_2}$$

   By *COMP* rule, for some ρ_1, ρ_2, f_1, f_2, we have $\rho_1; f_1 \vdash C_1$, $\rho_2; f_2 \vdash C_2$, $\rho_1 \cap \rho_2 = \emptyset$, $\rho = \rho_1 \cup \rho_2$, and $f = f_1 \oplus f_2$. By induction hypothesis, we have $\rho_1'; f_1' \vdash C_1'$ for some $\rho_1' \subset \rho_1$, and f_1'. Then $\rho_1' \cap \rho_2 = \emptyset$, and by *COMP* rule it follows that $\rho_1' \cup \rho_2; f_1' \oplus f_2 \vdash C_1'|C_2$. Note that $\rho_1' \cup \rho_2 \subset \rho$. Now the result follows if we show that $f_1' \oplus f_2$ satisfies the required conditions. For $y \in \rho_1' \cup \rho_2$, we consider only the case where $f(y) = x \in \rho$, and leave the other similar cases to the reader. If $y \in \rho_1 - \rho_2$ or $y \in \rho_2 - \rho_1$ the argument is simple. We consider only the case $y \in \rho_1 \cap \rho_2$. Since f_1 and f_2 are compatible, we have the following possible cases:$f_1(y) = f_2(y) = x$, or $f_1(y) = x, f_2(y) = \perp$, or $f_1(y) = \perp, f_2(y) = x$. We consider only the first. Since $f_1(y) = x$, by induction hypothesis it follows that $f_1'(y) \in \{x, \perp\}$. In any case we have $(f_1' \oplus f_2)(y) = x$ which satisfies the condition. The theorem thus follows.

3. *REQV*: The result follows by a simple application of Lemma 3. □

It is easy to verify that Theorem 1 also holds for both variants of Aπ.

Lemma 11 *Let* $(\nu x)C_1 \longrightarrow C_2$. *Then* $C_2 \equiv (\nu x)C_2'$ *for some* C_2' *such that* $C_1 \longrightarrow C_2'$.
Proof: By induction on the length of a derivation of $(\nu x)C_1 \longrightarrow C_2$. □

Proof of Theorem 2: Let $C_1 \mathrel{\overset{\sqsubset}{\sim}}_{\rho_1} C_2$. Suppose $\rho; f \vdash O$, $\rho \cap \rho_2 = \emptyset$, and $C_1 \ \underline{may}\ O$. Since $\rho_1 \subset \rho_2$, we have $\rho \cap \rho_1 = \emptyset$. Then since $C_1 \mathrel{\overset{\sqsubset}{\sim}}_{\rho_1} C_2$, we have $C_2 \ \underline{may}\ O$. Hence $C_1 \mathrel{\overset{\sqsubset}{\sim}}_{\rho_2} C_2$.

Let $fn(C_1) \cup fn(C_2) \subset \rho_1$ and $C_1 \mathrel{\overset{\sqsubset}{\sim}}_{\rho_2} C_2$. For O such that $\rho; f \vdash O$, $\rho \cap \rho_1 = \emptyset$, let $C_1 \ \underline{may}\ O$. We have to show $C_2 \ \underline{may}\ O$. Now, for $\tilde{x}$ such that $\{\tilde{x}\} = \rho$, we have $\emptyset; \{\} \vdash (\nu\tilde{x})O$, $fn(C_1) \cap \{\tilde{x}\} = \emptyset$, and $fn(C_2) \cap \{\tilde{x}\} = \emptyset$. Then $C_1|O \Longrightarrow C'|\overline{\mu}\mu$ implies $C_1|(\nu\tilde{x})O \equiv (\nu\tilde{x})(C_1|O) \Longrightarrow (\nu\tilde{x})(C'|\overline{\mu}\mu) \equiv (\nu\tilde{x})C'|\overline{\mu}\mu$. Hence $C_1 \ \underline{may}\ (\nu\tilde{x})O$. Now, since $C_1 \mathrel{\overset{\sqsubset}{\sim}}_{\rho_2} C_2$ we have

$C_2\ \underline{may}\ (\nu\tilde{x})O$. Since $fn(C_2) \cap \{\tilde{x}\} = \emptyset$, we have $(\nu\tilde{x})(C_2|O) \equiv C_2|(\nu\tilde{x})O \Longrightarrow C_3|\overline{\mu}\mu$, for some C_3. From Lemma 11, we deduce $C_2|O \Longrightarrow C_4$ such that $(\nu\tilde{x})C_4 \equiv C_3|\overline{\mu}\mu$. From this we deduce that $C_4 \equiv C_5|\overline{\mu}\mu$ for some C_5. It follows that $C_2\ \underline{may}\ O$. □

Proof of Lemma 4: Reflexivity and transitivity of $\leq$ is immediate from Definition 5. Let $[\rho_1, \chi_1] \leq [\rho_2, \chi_2]$ and $[\rho_2, \chi_2] \leq [\rho_1, \chi_1]$. Since $\rho_1 \subset \rho_2$ and $\rho_2 \subset \rho_1$ we have $\rho_1 = \rho_2$. From $\chi_1 \subset \rho_2 \cup \chi_2$, $\rho_1 = \rho_2$, and $\chi_1 \cap \rho_1 = \emptyset$, it follows that $\chi_1 \subset \chi_2$. By a similar argument $\chi_2 \subset \chi_1$, and hence $\chi_1 = \chi_2$. So $\leq$ is antisymmetric, and thus a partial order. □

Proof of Theorem 3: All the transition rules in Table 3 monotonically increase ρ and χ. Therefore $\rho_1 \subset \rho_2$ and $\chi_1 \subset \chi_2$. We show $C_2 : [\rho_2, \chi_2]$ by induction on the length of a derivation of a $\langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{\alpha} \langle\!\langle C_2 \rangle\!\rangle^{\rho_2}_{\chi_2}$. We are given that $\rho; f \vdash C_1$ and $[\rho, fn(C_1) - \rho] \leq [\rho_1, \chi_1]$.

There are two base cases. The derivation is by a direct application of

1. *IN*: We have
$$\mathit{IN}:\quad \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})xy} \langle\!\langle C_1 \mid \overline{x}y \rangle\!\rangle^{\rho_1}_{(\chi_1 \cup \{y\}) - \rho_1}$$
where $\hat{y} \cap (\rho_1 \cup \chi_1) = \emptyset$, $x \in \rho_1$, $C_2 = C_1 \mid \overline{x}y$, $\rho_2 = \rho_1$, and $\chi_2 = (\chi_1 \cup \{y\}) - \rho_1$. Then by *MSG* and *COMP* rules we have $\rho; f \vdash C_2$. Also, since $fn(C_1) \subset \rho_1 \cup \chi_1$, we have $fn(C_2) = fn(C_1) \cup \{x, y\} \subset \rho_1 \cup \chi_1 \cup \{x, y\} = \rho_2 \cup \chi_2$. The last equality is because $x \in \rho_1$. Thus, $C_2 : [\rho_2, \chi_2]$.

2. *OUT*: We have $C_1 = (\nu\hat{y})(C_2|\overline{x}y)$,
$$\mathit{OUT}:\quad \langle\!\langle (\nu\hat{y})(C_2|\overline{x}y) \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})\overline{x}y} \langle\!\langle C_2 \rangle\!\rangle^{\rho_1 \cup \hat{y}}_{\chi_1}$$
$\rho_2 = \rho_1 \cup \hat{y}$, $\chi_2 = \chi_1$, where $\hat{y} \cap (\rho_1 \cup \chi_1) = \emptyset$, and $x \in \chi_1$, By *MSG*, *COMP*, and *RES* rules, it follows that $\rho'; f' \vdash C_2$ where $\rho' \subset \rho \cup \hat{y}$ and $f = f'|\rho$. Since $\rho \subset \rho_1$, we have $\rho' \subset \rho_1 \cup \hat{y} = \rho_2$. Also, $fn(C_2) \subset fn(C_1) \cup \hat{y}$. Since $fn(C_1) \subset \rho_1 \cup \chi_1$, we have $fn(C_2) \subset \rho_1 \cup \hat{y} \cup \chi_1 = \rho_2 \cup \chi_2$. Thus, $C_2 : [\rho_2, \chi_2]$.

For the induction step there are two cases depending on the rule used for the last derivation step.

1. *TAU*: We have $\rho_1 = \rho_2, \chi_1 = \chi_2$ and
$$\mathit{TAU}:\quad \frac{C_1 \longrightarrow C_2}{\langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{\tau} \langle\!\langle C_2 \rangle\!\rangle^{\rho_1}_{\chi_1}}$$
By Theorem 1, we have $\rho'; f' \vdash C_2$ for some $\rho' \subset \rho$. Hence $\rho' \subset \rho \subset \rho_1$ Further, it is easy to show that free names in the target of a transition also occur free in the source. So $fn(C_2) \subset fn(C_1)$. Now, since $fn(C_1) \subset \rho_1 \cup \chi_1$, we have $fn(C_2) \subset \rho_1 \cup \chi_1$. Hence, $fn(C_2) - \rho' \subset \rho_1 \cup \chi_1$. We thus have $C_2 : [\rho_1, \chi_1]$.

2. *LEQV*: The theorem follows directly from Lemma 3. □

We only sketch the proof of Lemma 6 as its complete version is very tedious. Before the proof, a few definitions and lemmas are in order. For a term C, a subterm of C is said to be at the top level in C if it does not occur under an input prefix or inside a conditional construct. If $\rho; f \vdash C$ we define $rcp(C) = \rho$ and $ext(C) = fn(C) - \rho$. The derivation tree for a reduction step $C \longrightarrow C'$ has exactly one leaf $C_0 \longrightarrow C_0'$ that is an instance of *RECV*, *IF*, or *ELSE*. We call $C_0 \longrightarrow C_0'$ the *core* of the derivation. We can show that C_0 appears (with bound names possibly renamed) at the top level in C, and similarly C_0' (with possible renaming) appears at the top level in C'. A context is a configuration with a hole. Structural congruence is extended to contexts the obvious way.

Lemma 12 (core) *If $C_0 \longrightarrow C_0'$ is the core of reductions $C \longrightarrow C_1$ and $C \longrightarrow C_2$, then $C_1 \equiv C_2$.*

Proof: By induction on the length of the derivation tree of $C \longrightarrow C_1$, we can show that $C \equiv \mathcal{C}[C_0]$ and $C_1 \equiv \mathcal{C}[C_0']$. Similarly there is a $\mathcal{C}'$ such that $C \equiv \mathcal{C}'[C_0]$ and $C_2 \equiv \mathcal{C}'[C_0']$. From the fact that $\mathcal{C}[C_0] \equiv C \equiv \mathcal{C}'[C_0]$, it follows that $\mathcal{C}$ and $\mathcal{C}'$ are congruent. Therefore, $\mathcal{C}[C_0'] \equiv \mathcal{C}'[C_0']$. □

Lemma 13 (one-step-unzip) *If $rcp(C_1) \cap rcp(C_2) = \emptyset$ and $C_1|C_2 \longrightarrow C'$ then one of the following holds:*

1. *$C' \equiv C_1'|C_2'$, where $C_1 \longrightarrow C_1'$ and $C_2 = C_2'$.*
2. *$C' \equiv C_1'|C_2'$, where $C_2 \longrightarrow C_2'$ and $C_1 = C_1'$.*
3. *$C' \equiv C_1'|C_2'$, where $C_1 \equiv C_1'|\overline{x}y$, $C_2|\overline{x}y \longrightarrow C_2'$, and $x \in rcp(C_2)$.*
4. *$C' \equiv C_1'|C_2'$, where $C_2 \equiv C_2'|\overline{x}y$, $C_1|\overline{x}y \longrightarrow C_1'$, and $x \in rcp(C_1)$.*
5. *For any $y \notin fn(C_1) \cup fn(C_2)$, $C' = (\nu y)(C_1'|C_2')$, where $C_1 \equiv (\nu y)(C_1'|\overline{x}y)$, $C_2|\overline{x}y \longrightarrow C_2'$, and $x \in rcp(C_2)$.*
6. *For any $y \notin fn(C_1) \cup fn(C_2)$, $C' = (\nu y)(C_1'|C_2')$, where $C_2 \equiv (\nu y)(C_2'|\overline{x}y)$, $C_1|\overline{x}y \longrightarrow C_1'$, and $x \in rcp(C_1)$.*

Proof: Let $C_0 \longrightarrow C_0'$ be the core of $C_1|C_2 \longrightarrow C'$. There are three cases depending on whether the core is an instance of *IF*, *ELSE* or *RECV*. Suppose the core is an instance of *IF* or *ELSE*. Then it follows that C_0 (with its bound names possibly alpha renamed) occurs at the top level in either C_1 or C_2. If it occurs in C_1, case 1 of the lemma statement applies, else case 2 applies. If the core is an instance *RECV*, we have $C_0 \equiv C_{01}|C_{02}$, where $C_{01} = \overline{x}y$ and $C_{02} = x(z).C_{02}'$. There are four cases depending on where C_{01} and C_{02} appear (with their bound names possibly alpha-renamed) in $C_1|C_2$:

1. **(C_{01} and C_{02} are subterms of C_1)**: Since C_{01} and C_{02} are at the top level in C_1, it follows $C_1 \equiv (\nu\tilde{t})(C'|C_{01}|C_{02})$ for some $\tilde{t}$ and C'. Using the rules of Table 3, we can derive $C_1|C_2 \longrightarrow (\nu\tilde{t})(C'|C_0')|C_2$ with $C_0 \longrightarrow C_0'$ as its core. Then, by Lemma 12 we have $C' \equiv C_1'|C_2'$, where $C_1' = (\nu\tilde{t})(C'|C_0')$ and $C_2' = C_2$. Thus, statement 1 of the lemma applies.

2. **(C_{01} and C_{02} are subterms of C_2)**: Similar to case 1.

3. **(C_{01} is a subterm of C_1 and C_{02} a subterm of C_2)**: Since C_{01} (C_{02}) is at the top level in C_1 (C_2). Depending on whether y is restricted in C_1, we have two subcases:

   - $C_1 \equiv (\nu\tilde{u})(C_1''|\overline{x}y)$ and $C_2 \equiv (\nu\tilde{v})(x(z).C_{02}'|C_2'')$ such that $x, y \notin \{\tilde{u}, \tilde{v}\}$. Therefore, $x \in \mathit{rcp}(C_2)$. We have $C_1|C_2 \equiv (\nu\tilde{u})(C_1'')|(\nu\tilde{v})(\overline{x}y|x(z).C_{02}'|C_2'')$. Thus, we can derive $C_1|C_2 \longrightarrow (\nu\tilde{u})(C_1'')|(\nu\tilde{v})(C_0'|C_2'')$ with $C_0 \longrightarrow C_0'$ as its core. Then by Lemma 12, $C' \equiv C_1'|C_2'$ where $C_1' \equiv (\nu\tilde{u})C_1''$ and $C_2' \equiv (\nu\tilde{v})(C_0'|C_2'')$. Hence, statement 3 of the lemma applies.
   - For any $y \notin \mathit{fn}(C_1) \cup \mathit{fn}(C_2)$, we can write

     $$C_1 \equiv (\nu\tilde{u}, y)(C_1''|\overline{x}y) \quad \texttt{and} \quad C_2 \equiv (\nu\tilde{v})(x(z).C_{02}'|C_2'')$$

     such that $x, y \notin \{\tilde{u}, \tilde{v}\}$. Therefore, $x \in \mathit{rcp}(C_2)$. We have

     $$C_1|C_2 \equiv (\nu y)((\nu\tilde{u})(C_1'')|(\nu\tilde{v})(\overline{x}y|x(z).C_{02}'|C_2'')).$$

     Thus, we can derive $C_1|C_2 \longrightarrow (\nu y)((\nu\tilde{u})(C_1'')|(\nu\tilde{v})(C_0'|C_2''))$ with $C_0 \longrightarrow C_0'$ as its core. Then by Lemma 12, $C' \equiv (\nu y)(C_1'|C_2')$ where $C_1' \equiv (\nu\tilde{u})C_1''$ and $C_2' \equiv (\nu\tilde{v})(C_0'|C_2'')$. Hence, statement 5 of the lemma applies.

4. **(C_{01} is a subterm of C_2 and C_{02} a subterm of C_1)**: Similar to case 3. □

Proof of Lemma 6 (zip-unzip):
(if : zip) The proof is by induction on length of s.

For the base case, we have $s = \epsilon$, and

$$\langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \Longrightarrow \langle\!\langle C_1' \rangle\!\rangle^{\rho_1}_{\chi_1}, \ \langle\!\langle C_2 \rangle\!\rangle^{\rho_2}_{\chi_2} \Longrightarrow \langle\!\langle C_2' \rangle\!\rangle^{\rho_2}_{\chi_2}$$

By Lemma 5, we have $C_1 \Longrightarrow C_1'$ and $C_2 \Longrightarrow C_2'$. Then by repeated application of PAR we have $C_1|C_2 \Longrightarrow C_1'|C_2 \Longrightarrow C_1'|C_2'$. The result follows from the fact that $bn(\epsilon) = \emptyset$.

For the induction step, there are two cases out of which we only consider the case $s = s'.(\hat{y})xy$. From IN, OUT, and LEQV we have

$$\begin{array}{ll}
\langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \overset{s'}{\Longrightarrow} \langle\!\langle C_3 \rangle\!\rangle^{\rho_1'}_{\chi_1'} \xrightarrow{(\hat{y})xy} \langle\!\langle C_3|\overline{x}y \rangle\!\rangle^{\rho_1'}_{(\chi_1' \cup \{y\})-\rho_1'} \Longrightarrow \langle\!\langle C_1' \rangle\!\rangle^{\rho_1'}_{(\chi_1' \cup \{y\})-\rho_1'} & \hat{y} \cap (\rho_1' \cup \chi_1') = \emptyset, x \in \rho_1' \\
\langle\!\langle C_2 \rangle\!\rangle^{\rho_2}_{\chi_2} \overset{\overline{s}'}{\Longrightarrow} \langle\!\langle (\nu\hat{y})(C_4|\overline{x}y) \rangle\!\rangle^{\rho_2'}_{\chi_2'} \xrightarrow{(\hat{y})\overline{x}y} \langle\!\langle C_4 \rangle\!\rangle^{\rho_2' \cup \hat{y}}_{\chi_2'} \Longrightarrow \langle\!\langle C_2' \rangle\!\rangle^{\rho_2' \cup \hat{y}}_{\chi_2'} & \hat{y} \cap (\rho_2' \cup \chi_2') = \emptyset, x \in \chi_2'
\end{array}$$

From Lemma 5, $C_3|\overline{x}y \Longrightarrow C_1'$ and $(\nu\hat{y})(C_4|\overline{x}y) \Longrightarrow C_2'$. Let $\tilde{z} = bn(s')$. By induction hypothesis $C_1|C_2 \Longrightarrow C'$ where $C' \equiv (\nu\tilde{z})(C_3|(\nu\hat{y})(C_4|\overline{x}y))$. By Theorem 3, we have $C_3 : [\rho_1', \chi_1']$. It follows $\mathit{fn}(C_3) \subset \rho_1' \cup \chi_1'$. Now, since $\hat{y} \cap (\rho_1' \cup \chi_1') = \emptyset$, it follows that $\hat{y} \cap \mathit{fn}(C_3) = \emptyset$. Using this, and by repeated application of PAR and RES we have $C' \equiv (\nu\tilde{z}, \hat{y})(C_3|\overline{x}y|C_4) \Longrightarrow (\nu\tilde{z}, \hat{y})(C_1'|C_2')$. The lemma follows from the observation that $\{\tilde{z}, \hat{y}\} = bn(s)$.

(only if : unzip) Proof by induction on the length of the computation path $C_1|C_2 \Longrightarrow C$. For the base case where the length is zero, the result holds with $s = \epsilon$. For the induction step, let the length be $n > 0$. By induction hypothesis, we can unzip the first $n-1$ steps of $C_1|C_2 \Longrightarrow C'' \longrightarrow C$ as

$$\begin{array}{lcl} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} & \overset{s'}{\Longrightarrow} & \langle\!\langle C'_1 \rangle\!\rangle^{\rho'_1}_{\chi'_1} \\ \langle\!\langle C_2 \rangle\!\rangle^{\rho_2}_{\chi_2} & \overset{\overline{s'}}{\Longrightarrow} & \langle\!\langle C'_2 \rangle\!\rangle^{\rho'_2}_{\chi'_2} \end{array} \quad \texttt{and } C'' \equiv (\nu\tilde{z})(C'_1|C'_2), \{\tilde{z}\} = bn(s').$$

It is easy to show $\rho'_1 \cap \rho'_2 = \emptyset$. We are done if we show $(\nu\tilde{z})(C'_1|C'_2) \longrightarrow C$ can be unzipped. From Lemma 11 we have $C'_1|C'_2 \longrightarrow C'$ where $C \equiv (\nu\tilde{z})(C')$. Now, there are four cases according to Lemma 13 out of which we only consider cases 1 and 5 (the others are similar).

1. **(case 1):** We have $C' \equiv C''_1|C''_2$, $C'_1 \longrightarrow C''_1$, $C''_2 = C'_2$. From $C'_1 \longrightarrow C''_1$ we construct $\langle\!\langle C_1 \rangle\!\rangle^{\rho}_{\chi} \overset{s'}{\Longrightarrow} \langle\!\langle C'_1 \rangle\!\rangle^{\rho'_1}_{\chi'_1} \Longrightarrow \langle\!\langle C''_1 \rangle\!\rangle^{\rho'_1}_{\chi'_1}$. So, $C \equiv (\nu\tilde{z})(C') \equiv (\nu\tilde{z})(C''_1|C''_2) \equiv (\nu\tilde{z})(C''_1|C'_2)$. The result now holds with $s = s'$.
2. **(case 5):** By Theorem 3, $C'_1 : [\rho'_1, \chi'_1]$ and $C'_2 : [\rho'_2, \chi'_2]$, from which it follows that $fn(C'_1) \subset \rho'_1 \cup \chi'_1$ and $fn(C'_2) \subset \rho'_2 \cup \chi'_2$. Then by picking $y \notin (\rho'_1 \cup \rho'_2 \cup \chi'_1 \cup \chi'_2)$ for case 5 of Lemma 13, we have $C' \equiv (\nu y)(C''_1|C''_2)$, $C'_1 \equiv (\nu y)(C''_1|\overline{x}y)$, $C'_2|\overline{x}y \longrightarrow C''_2$, and $x \in rcp(C'_2)$.

   Then we have

   $$\langle\!\langle C'_1 \rangle\!\rangle^{\rho'_1}_{\chi'_1} \xrightarrow{\overline{x}(y)} \langle\!\langle C''_1 \rangle\!\rangle^{\rho'_1 \cup \{y\}}_{\chi'_1} \quad \texttt{and} \quad \langle\!\langle C'_2 \rangle\!\rangle^{\rho'_2}_{\chi'_2} \xrightarrow{x(y)} \langle\!\langle C'_2|\overline{x}y \rangle\!\rangle^{\rho'_2}_{\chi'_2 \cup \{y\}} \xrightarrow{\tau} \langle\!\langle C''_2 \rangle\!\rangle^{\rho'_2}_{\chi'_2 \cup \{y\}}$$

   Then $C \equiv (\nu\tilde{z})C' \equiv (\nu\tilde{z}, y)(C''_1|C''_2)$. The result follows by setting $s = s'.\overline{x}(y)$ and noting that $bn(s) = \{\tilde{z}, y\}$. □

Note that zip-unzip lemma holds for both variants of Aπ, because the core of a reduction is now *RECV* or either *BCASE* or *BMATCH*. The latter can be treated in the same way as *IF* and *ELSE*

Proof of Lemma 7:

1. The proof is by induction on the length of s. For the base case $s = \epsilon$, by *TAU* rule we have $\rho' = \rho, \chi' = \chi$, and the result follows. For the induction step we consider only the case $s = s'.(\hat{y})xy$, and leave the case $s = s'.(\hat{y})\overline{x}y$ to the reader. We have

   $$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{s'}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})xy} \langle\!\langle C_2 \rangle\!\rangle^{\rho_1}_{(\chi_1 \cup \{y\}) - \rho_1} \Longrightarrow \langle\!\langle C' \rangle\!\rangle^{\rho_1}_{(\chi_1 \cup \{y\}) - \rho_1}$$

   and $\rho' = \rho_1$, $\chi' = (\chi_1 \cup \{y\}) - \rho_1$. From induction hypothesis, $\rho_1 = rcp([\rho, \chi], s')$ and $\chi_1 = ext([\rho, \chi], s')$. Then $\rho' = \rho_1 = rcp([\rho, \chi], s') = rcp([\rho, \chi], s'.(\hat{y})xy)$, and $\chi' = (\chi_1 \cup \{y\}) - \rho_1 = (\{y\} \cup ext([\rho, \chi], s')) - rcp([\rho, \chi], s') = ext([\rho, \chi], s'.(\hat{y})xy)$.
2. From part 1 of this lemma and *IN* rule.

3. We have $s = s'.(\hat{y})\overline{x}y$ and

$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{s'}{\Longrightarrow} \langle\!\langle (\nu\hat{y})(C_1|\overline{x}y) \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})\overline{x}y}$$

By part 1 of this lemma, $\rho_1 = rcp([\rho,\chi], s')$ and $\chi_1 = ext([\rho,\chi], s')$. By OUT rule, $x \in \chi_1 = ext([\rho,\chi], s')$. Let $C'_1 = (\nu\hat{y})(C_1|\overline{x}y)$. By Theorem 3, $\rho'_1; f'_1 \vdash C'_1$, for some $[\rho'_1, fn(C'_1) - \rho'_1] \leq [\rho_1, \chi_1]$. If $\hat{y} = \emptyset$ then either $y \in \rho'_1$ or $y \in fn(C'_1) - \rho'_1$. In either case, from $[\rho'_1, fn(C'_1) - \rho'_1] \leq [\rho_1, \chi_1]$ it follows that $y \in \rho_1 \cup \chi_1 = rcp([\rho,\chi], s') \cup ext([\rho,\chi], s')$. Conversely, if $y \in rcp([\rho,\chi], s') \cup ext([\rho,\chi], s')$ then by the side condition of OUT rule, we have $\hat{y} = \emptyset$.

4. The proof is by induction on length of s. The base case is obvious. For the induction step there are two cases out of which we only consider $s = s'.(\hat{y})xy$. By Definition 7, we have $rcp([\rho,\chi], s) \cup ext([\rho,\chi], s) = rcp([\rho,\chi], s') \cup ((\{y\} \cup ext([\rho,\chi], s')) - rcp([\rho,\chi], s')) = \{y\} \cup rcp([\rho,\chi], s') \cup ext([\rho,\chi], s') = \{x, y\} \cup rcp([\rho,\chi], s') \cup ext([\rho,\chi], s')$. The last equality follows from part 2 of this lemma. From induction hypothesis we have $rcp([\rho,\chi], s') \cup ext([\rho,\chi], s') = n(s') \cup \chi \cup \rho$. Using this, we get $rcp([\rho,\chi], s) \cup ext([\rho,\chi], s) = \{x, y\} \cup n(s') \cup \chi \cup \rho$. Now, the result follows from the observation that $\{x, y\} \cup n(s') = n(s)$.

5. We have

$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{s_1}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{\alpha} \langle\!\langle C_2 \rangle\!\rangle^{\rho_2}_{\chi_2} \overset{s_2}{\Longrightarrow}$$

By IN and OUT rules, we have $bn(\alpha) \cap (\rho_1 \cup \chi_1) = \emptyset$. By parts 1 and 4 of this lemma $\rho_1 \cup \chi_1 = rcp([\rho,\chi], s_1) \cup ext([\rho,\chi], s_1) = n(s_1) \cup \rho \cup \chi$, and the result follows. $\square$

Lemma 14 *Let $r, s \in \mathcal{L}^*[\rho,\chi]$, and $r = r_0 \prec r_1 \prec \ldots \prec r_n \prec s$. Then $r_i \in \mathcal{L}^*[\rho,\chi]$ for $1 \leq i \leq n$.*

Proof: Suppose for some $1 \leq i \leq n$, $r_i \notin \mathcal{L}^*[\rho,\chi]$. Then r_i violates either property 2 or 3 of Lemma 7. We consider only the later, the former is similar. We have $r_i = t_1.(\hat{y})\overline{x}y.t_2$ and at least one of the following conditions holds.

1. $x \notin ext([\rho,\chi], t_1)$: There are four subcases depending on which law $r_{i-1} \prec r_i$ is an instance of.

   - *L1* or *L2*: Then $r_{i-1} = t_1.(\hat{y})\overline{x}y.t_3$ for some t_3, and hence $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$.
   - *L3*: Then $r_{i-1} = t'_1.(\hat{y})\overline{x}y.t'_2$ such that $ext([\rho,\chi], t'_1) = ext([\rho,\chi], t_1)$. Then $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$.
   - *L4*: Then from the observation that output actions do not change the set of external names it follows that $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$.
   - *L5*: Then we have $t_1 = t'_1.(\hat{v})uv$, and $r_{i-1} = t'_1.(\hat{y})\overline{x}y.(\hat{v})uv.t_2$. But then again $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$ because $ext([\rho,\chi], t'_1) \subset ext([\rho,\chi], t_1)$.

   Thus, in all cases $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$. When this argument is applied repeatedly we get the contradiction that $r \notin \mathcal{L}^*[\rho,\chi]$.

2. $y \notin \mathit{rcp}([\rho,\chi],t_1) \cup \mathit{ext}([\rho,\chi],t_1)$ and $\hat{y} = \emptyset$: A similar case analysis as in case 1 shows that $r_{i-1} \notin \mathcal{L}^*[\rho,\chi]$, which again when repeatedly applied leads to the contradiction that $r \notin \mathcal{L}^*[\rho,\chi]$.

3. $y \in \mathit{rcp}([\rho,\chi],t_1) \cup \mathit{ext}([\rho,\chi],t_1)$ and $\hat{y} = \{y\}$: By property 1 of Lemma 15, $y \in n(t_1) \cup \rho \cup \chi$. Then, since $\hat{y} = \{y\}$ property 2 of Lemma 15 is violated. Contradiction.

In all cases we have arrived at a contradiction. Thus $r_{i-1} \in \mathcal{L}^*[\rho,\chi]$ for all $1 \le i \le n$. (For the case where we start with the initial assumption that r_i violates property 2 of Lemma 7, we use a similar argument as above, but move up the path to arrive at the contradiction that $s \notin \mathcal{L}^*[\rho,\chi]$.) □

Proof of Lemma 8: We have $\overline{r}, \overline{s} \in \mathcal{L}^*[\rho,\chi]$, $r \preceq s$. Let $r \prec r_1 \prec \ldots \prec r_n \prec s$. By Lemma 10, $\overline{s} \prec \overline{r_1} \prec \ldots \prec \overline{r_n} \prec \overline{r}$. Then by Lemma 14, we have $\overline{r_i} \in \mathcal{L}^*[\rho,\chi]$ for $1 \le i \le n$. So the lemma follows by a simple induction on n if we prove it just for the case $r \prec s$.

Let $r \prec s$. There are five cases one for each law in Table 4. We consider only *L1, L3* and *L5*.

1. *L1*: Let $r = s.(\hat{y})\overline{x}y$. We know that $\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s}}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1}$. By Lemma 7, we have $\rho_1 = \mathit{rcp}([\rho,\chi],\overline{s})$, $\chi_1 = \mathit{ext}([\rho,\chi],\overline{s})$. Then from $\overline{r} \in \mathcal{L}^*[\rho,\chi]$ it follows $x \in \rho_1$, $\hat{y} \cap (\rho_1 \cup \chi_1) = \emptyset$. Now, by *IN* rule we have $\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s}}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})xy}$.

2. *L3*: Let $s = s_1.(\hat{y})xy.(\hat{v})uv.s_2$ and $r = s_1.(M)uv.(N)xy.s_2$ where M and N are as defined in the side condition of *L3*. We know that
$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s_1}}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})\overline{x}y} \langle\!\langle C_2 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{\chi_1} \Longrightarrow \langle\!\langle C_3 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{\chi_1} \xrightarrow{(\hat{v})\overline{u}v} \langle\!\langle C_4 \rangle\!\rangle^{\rho_1\cup\hat{y}\cup\hat{v}}_{\chi_1} \overset{\overline{s_2}}{\Longrightarrow} .$$
From *OUT* we deduce $C_1 \equiv (\nu\hat{y})(C_2|\overline{x}y)$ and $C_3 \equiv (\nu\hat{v})(C_4|\overline{u}v)$. Then we have
$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s_1}}{\Longrightarrow} \langle\!\langle (\nu\hat{y})(C_2|\overline{x}y) \rangle\!\rangle^{\rho_1}_{\chi_1} \Longrightarrow \langle\!\langle (\nu\hat{y},\hat{v})(C_4|\overline{x}y|\overline{u}v) \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(M)\overline{u}v}$$
$$\langle\!\langle (\nu N)(C_4|\overline{x}y) \rangle\!\rangle^{\rho_1\cup M}_{\chi_1} \xrightarrow{(N)\overline{x}y} \langle\!\langle C_4 \rangle\!\rangle^{\rho_1\cup\hat{y}\cup\hat{v}}_{\chi_1} \overset{\overline{s_2}}{\Longrightarrow}$$
because $M \cup N = \hat{y} \cup \hat{v}$.

3. *L5*: Let $s = s_1.(\hat{y})xy.(\hat{v})\overline{u}v.s_2$ and $r = s_1.(\hat{v})\overline{u}v.(\hat{y})xy.s_2$, where $u, v \notin \hat{y}$. We know that
$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s_1}}{\Longrightarrow} \langle\!\langle C_1 \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{y})\overline{x}y} \langle\!\langle C_2 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{\chi_1} \Longrightarrow \langle\!\langle C_3 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{\chi_1} \xrightarrow{(\hat{v})uv} \langle\!\langle C_4 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{(\chi_1\cup\{v\})-(\rho_1\cup\hat{y})} \overset{\overline{s_2}}{\Longrightarrow} .$$
From *IN* we deduce $C_4 \equiv (C_3|\overline{u}v)$, and from *OUT* we deduce $C_1 \equiv (\nu\hat{y})(C_2|\overline{x}y)$, $\hat{y} \cap (\rho_1 \cup \chi_1) = \emptyset$. Then we have
$$\langle\!\langle C \rangle\!\rangle^{\rho}_{\chi} \overset{\overline{s_1}}{\Longrightarrow} \langle\!\langle (\nu\hat{y})(C_2|\overline{x}y) \rangle\!\rangle^{\rho_1}_{\chi_1} \xrightarrow{(\hat{v})uv} \langle\!\langle (\nu\hat{y})(C_2|\overline{x}y|\overline{u}v) \rangle\!\rangle^{\rho_1}_{(\chi_1\cup\hat{v})-\rho_1} \xrightarrow{(\hat{y})\overline{x}y}$$
$$\langle\!\langle C_2|\overline{u}v \rangle\!\rangle^{\rho_1\cup\hat{y}}_{(\chi_1\cup\hat{v})-\rho_1} \Longrightarrow \langle\!\langle C_4 \rangle\!\rangle^{\rho_1\cup\hat{y}}_{(\chi_1\cup\hat{v})-\rho_1} \overset{\overline{s_2}}{\Longrightarrow} .$$
because $(\chi_1 \cup \hat{v}) - \rho_1 = (\chi_1 \cup \hat{v}) - (\rho_1 \cup \hat{y})$. □

We define name subsitution on paths the obvious way, with the usual renaming of bound names to avoid capture of free names during substitution.

Proof of Lemma 9: The proof is by induction on length of s. For the base case, we have $s = \epsilon$. It follows from the definition of $O([\rho, \chi], \epsilon)$ that $\overline{r}$ contains only inputs, and r only outputs. Lemma follows from repeated application of **(L1)**. For the induction step, there are three cases:

1. $\mathbf{s} = \overline{\mathbf{x}}(\mathbf{y}).\mathbf{s}'$: Then $x \in \chi$, and $y \notin \rho \cup \chi$. Since $s \in \mathcal{L}^*[\rho, \chi]$, we have $s' \in \mathcal{L}^*[\rho \cup \{y\}, \chi]$. Note that $O([\rho, \chi], \overline{x}(y).s')$ first waits for a message $\overline{x}w$ for some $w \notin \rho \cup \chi$ before generating an event or sending any messages. From this observation, it follows that r is of form $(\hat{v_1})\overline{u_1}v_1. \ldots (\hat{v_n})\overline{u_n}v_n.(\hat{w})\overline{x}w.r_0$. Since $r \in \mathcal{L}^*[\rho, \chi]$ and $w \notin \rho \cup \chi$, it follows that $\hat{w} = \{w\} - \cup_{1 \leq n} \hat{v_i}$, i.e. r has a bound output with argument w. Then $\overline{r}$ has a bound input with argument w. Then, since $\overline{r} \in \mathcal{L}^*[\chi', \rho]$, we have $w \notin \chi' \cup \rho$. Then we have
$$\langle\!\langle O([\rho, \chi], s) \rangle\!\rangle_{\rho}^{\chi'} \overset{x(w)}{\Longrightarrow} \langle\!\langle O([\rho \cup \{w\}, \chi], s'\{w/y\}) \rangle\!\rangle_{\rho \cup \{w\}}^{\chi'} \overset{\overline{r'}}{\Longrightarrow}$$
where $\overline{r'} = (\hat{v'_1})u_1v_1. \ldots (\hat{v'_n})u_nv_n.\overline{r_0}$, $\hat{v'_i} = \hat{v_i} - \{w\}$. Clearly, $r' \in \mathcal{L}^*[\rho \cup \{w\}, \chi]$. It is easy to show since $w \notin \rho \cup \chi'$, $s'\{w/y\} \in \mathcal{L}^*[\rho \cup \{w\}, \chi]$. Further, $\chi' = \chi \cup ffi([\rho \cup \{w\}, \chi], s'\{w/y\})$. By induction hypothesis, $r' \preceq s'\{w/y\}$. Then $\overline{x}(w).r' \preceq \overline{x}(w).(s'\{w/y\})$. By repeated application of **L4** we deduce $r \preceq \overline{x}(w).r'$. The result follows from transitivity of $\preceq$ and that s is alpha equivalent to $\overline{x}(w).(s'\{w/y\})$.

2. $\mathbf{s} = \overline{\mathbf{x}}\mathbf{y}.\mathbf{s}'$: Then $x \in \chi$, and $y \in \rho \cup \chi$. Since $s \in \mathcal{L}^*[\rho, \chi]$, we have $s' \in \mathcal{L}^*[\rho, \chi]$. Note that $O([\rho, \chi], \overline{x}y.s')$ first waits for a message $\overline{x}y$ before generating an event or sending any messages. From this observation, it follows that $\overline{r}$ is of form $(\hat{v_1})u_1v_1. \ldots (\hat{v_n})u_nv_n.xy.\overline{r_0}$, where $y \notin \cup_i \hat{v_i}$. Then we also have
$$\langle\!\langle O([\rho, \chi], s) \rangle\!\rangle_{\rho}^{\chi'} \overset{xy}{\Longrightarrow} \langle\!\langle O([\rho, \chi], s') \rangle\!\rangle_{\rho}^{\chi'} \overset{\overline{r'}}{\Longrightarrow}$$
where $\overline{r'} = (\hat{v_1})u_1v_1. \ldots (\hat{v_n})u_nv_n.\overline{r_0}$. Further, since $r \in \mathcal{L}^*[\rho, \chi]$ it is clear that $r' \in \mathcal{L}^*[\rho, \chi]$. Moreover, since $ffi([\rho, \chi], s) = ffi([\rho, \chi], s')$, we have $\chi' = \chi \cup ffi([\rho, \chi], s')$. By induction hypothesis, $r' \preceq s'$. Then $\overline{x}y.r' \preceq \overline{x}y.s'$. By repeated application of **L4** we deduce $r \preceq \overline{x}y.r'$. The result follows from transitivity of $\preceq$.

3. $\mathbf{s} = \mathbf{x}\mathbf{y}.\mathbf{s}'$: Then $x \in \rho$, and since $s \in \mathcal{L}^*[\rho, \chi]$, we have $s' \in \mathcal{L}^*[\rho, \chi \cup \{y\} - \rho]$. Now, we can show
$$O([\rho, \chi], s) \equiv \overline{x}y \mid O([\rho, \chi \cup \{y\} - \rho], s')$$
There are two possible cases depending on whether $\overline{x}y$ fires or not. We consider only the case where it fires, the other is similar. Since $\overline{x}y$ fires, it follows that $\overline{r} = \overline{r_1}.\overline{x}y.\overline{r_2}$, where $y \notin bn(r_1)$, because $y \in \chi' \cup \rho$. Then it is the case that
$$\langle\!\langle O([\rho, \chi \cup \{y\} - \rho], s') \rangle\!\rangle_{\rho}^{\chi'} \overset{\overline{r_1.r_2}.\overline{\mu}\mu}{\Longrightarrow}$$

Further, since $r \in \mathcal{L}^*[\rho, \chi]$ and $y \notin \mathit{bn}(r_1)$, we have $r_1.r_2 \in \mathcal{L}^*[\rho, \chi \cup \{y\} - \rho]$. Since $\chi' = \chi \cup \mathit{ffi}([\rho, \chi], s)$, we have $\chi' = (\chi \cup \{y\} - \rho) \cup \mathit{ffi}([\rho, \chi \cup \{y\} - \rho], s')$. By induction hypothesis, $r_1.r_2 \preceq s'$. Then $xy.r_1.r_2 \preceq xy.s'$. By repeated application of *L3, L5*, we have $r \preceq xy.r_1.r_2$. The result follows by transitivity of $\preceq$.

4. $\mathbf{s = x(y).s'}$: Then $x \in \rho$, and $y \notin \rho \cup \chi$. Since $s \in \mathcal{L}^*[\rho, \chi]$, we have $s' \in \mathcal{L}^*[\rho, \chi \cup \{y\}]$. Now, we can show

$$O([\rho, \chi], s) \equiv (\nu y)(\overline{x}y \mid O([\rho, \chi \cup \{y\}], s'))$$

There are two possible cases depending on whether $\overline{x}y$ fires or not. We consider only the case where it does not fire, the other is similar. Since $\overline{x}y$ never fires, all the interactions are performed by $(\nu y)(O([\rho, \chi \cup \{y\}], s'))$, that is

$$\langle\!\langle (\nu y)(O([\rho, \chi \cup \{y\}], s')) \rangle\!\rangle_{\rho}^{\chi'} \xRightarrow{\overline{r}.\overline{\mu}\mu}$$

During the computation above, y may be alpha renamed to other names. Furthermore, either the name is exported through some other message, or never exported at all. The second case is simpler; so we only consider the case where the name is exported at some point as a fresh name w, i.e $\overline{r} = \overline{r_1}.\overline{u}(w).\overline{r_2}$ where $w \notin \mathit{rcp}([\chi', \rho], \overline{r_1}) \cup \mathit{ext}([\chi', \rho], \overline{r_1})$. Then we can deduce

$$\langle\!\langle O([\rho, \chi \cup \{w\}], s'\{w/y\}) \rangle\!\rangle_{\rho}^{\chi' \cup \{w\}} \xRightarrow{\overline{r'}.\overline{\mu}\mu}$$

where $\overline{r'} = \overline{r_1}.\overline{u}w.\overline{r_2}$. Further, since $r \in \mathcal{L}^*[\rho, \chi]$, we have $r' \in \mathcal{L}^*[\rho, \chi \cup \{w\}]$. It is easy to show that since $w \notin \rho \cup \chi'$, we have $s'\{w/y\} \in \mathcal{L}^*[\rho, \chi \cup \{w\}]$. Since $\chi' = \chi \cup \mathit{ffi}([\rho, \chi], s)$, we have $\chi' \cup \{w\} = (\chi \cup \{w\}) \cup \mathit{ffi}([\rho, \chi \cup \{w\}], s'\{w/y\})$. By induction hypothesis, $r' \preceq s'\{w/y\}$. Then $x(w).r' \preceq x(w).s'\{w/y\}$. By repeated application of *L3, L5*, we have $r.xw \preceq x(w).r'$, and by *L2*, we have $r \preceq r.xw$. The result follows by transitivity of $\preceq$, and that s is alpha equivalent to $x(w).s'\{w/y\}$. □

Proof of Theorem 4: Let $[\rho_1, \chi_1]$ be the minimal interface of C_1 and $[\rho_2, \chi_2]$ that of C_2.

(if) Let $C_1 \ll_\rho C_2$, and C_1 $\underline{\mathit{may}}$ O. We have $\rho_1, \rho_2 \subset \rho$. Let $\chi = (\chi_1 \cup \chi_2) - \rho$. Let $[\rho'', \chi'']$ be the minimal interface of O, $\rho' = \rho'' \cup \chi$ and $\chi' = \chi'' - \chi$. Then $O : [\rho', \chi']$, and since $\rho'' \cap \rho = \emptyset$ we have $\rho' \cap \rho = \emptyset$. From Lemma 6, it follows that the computation $C_1|O \Longrightarrow C'|\overline{\mu}\mu$ can be unzipped into $\langle\!\langle C_1 \rangle\!\rangle_{\chi}^{\rho} \xRightarrow{s}$ and $\langle\!\langle O \rangle\!\rangle_{\chi'}^{\rho'} \xRightarrow{\overline{s}} \langle\!\langle O'|\overline{\mu}\mu \rangle\!\rangle_{\chi'''}^{\rho'''}$. Then $\langle\!\langle C_2 \rangle\!\rangle_{\chi}^{\rho} \xRightarrow{r}$ for some $r \preceq s$, and $\langle\!\langle O \rangle\!\rangle_{\chi'}^{\rho'} \xRightarrow{\overline{s}.\overline{\mu}\mu}$. Since $\mu \notin \rho \cup \chi$, we have $r.\mu\mu, s.\mu\mu \in \mathcal{L}^*[\rho \cup \{\mu\}, \chi]$, and we can show by induction on the length of a derivation of $r \preceq s$ that $r.\mu\mu \preceq s.\mu\mu$. By a similar induction we can show that, since $\overline{s}.\overline{\mu}\mu \in \mathcal{L}^*[\rho', \chi']$ and $r \in \mathcal{L}^*[\rho, \chi]$ we have $\overline{r}.\overline{\mu}\mu \in \mathcal{L}^*[\rho', \chi']$. Then by Lemma 8, $\langle\!\langle O \rangle\!\rangle_{\chi'}^{\rho'} \xRightarrow{\overline{r}.\overline{\mu}\mu}$ and we can zip up these computations to produce $C_2|O \Longrightarrow C_2'|\overline{\mu}\mu$. Hence C_2 $\underline{\mathit{may}}$ O.

(only if) Let $C_1 \mathrel{\underset{\sim}{\sqsubset}}_\rho C_2$, $\chi = (\chi_1 \cup \chi_2) - \rho$. Let $\langle\!\langle C_1 \rangle\!\rangle_{\chi}^{\rho} \xRightarrow{s}$. Let $\chi' = \chi \cup \mathit{ffi}([\rho, \chi], s)$. It is clear from Definition 10 that $\langle\!\langle O([\rho, \chi], s) \rangle\!\rangle_{\rho}^{\chi'} \xRightarrow{\overline{s}.\overline{\mu}\mu}$. We can zip these up to get C_1 $\underline{\mathit{may}}$ $O([\rho, \chi], s)$, and therefore C_2 $\underline{\mathit{may}}$ $O([\rho, \chi], s)$. The computation $C_2|O([\rho, \chi], s) \xRightarrow{\overline{\mu}\mu}$ can be unzipped into $\langle\!\langle C_2 \rangle\!\rangle_{\chi}^{\rho} \xRightarrow{r}$ and $\langle\!\langle O([\rho, \chi], s) \rangle\!\rangle_{\rho}^{\chi'} \xRightarrow{\overline{r}.\overline{\mu}\mu}$, for some $r \in \mathcal{L}^*[\rho, \chi]$. Then by Lemma 9, $r \preceq s$, and hence $C_1 \ll_\rho C_2$. □

Lemma 15 *For* $s \in \mathcal{L}^*[\rho, \chi]$

1. $rcp([\rho, \chi], s) \cup ext([\rho, \chi], s) = n(s) \cup \rho \cup \chi$, *and*

2. $s = s'.\alpha$ *implies* $bn(\alpha) \cap (n(s') \cup \rho \cup \chi) = \emptyset$.

Proof: By induction on s. $\square$

Proof of Lemma 10: There are five cases for $r \prec s$ one for each law in Table 4. We consider only **(L5)** in detail. We have $r = s_1.(\hat{v})\overline{u}v.(\hat{y})xy.s_2$, $s = s_1.(\hat{y})xy.(\hat{v})\overline{u}v.s_2$, and $u, v \notin \hat{y}$. Applying Lemma 15 to s, we have $x, y \notin \hat{v}$. Then by **(L5)** $\overline{s} \preceq \overline{r}$. For the other cases, the reader may verify that **(L1)** and **(L2)** complement each other, and so do **(L3)** and **(L4)**. $\square$

Proof of Theorem 5: Let $\langle\!\langle C_1 \rangle\!\rangle^{\rho}_{\chi} \stackrel{s}{\Longrightarrow}$. Then $\langle\!\langle C_2 \rangle\!\rangle^{\rho}_{\chi} \stackrel{r}{\Longrightarrow}$ for some $r \preceq s$. From Lemma 10, we have $\overline{s} \preceq \overline{r}$. From Lemma 8, we conclude $\langle\!\langle C_2 \rangle\!\rangle^{\rho}_{\chi} \stackrel{s}{\Longrightarrow}$. $\square$