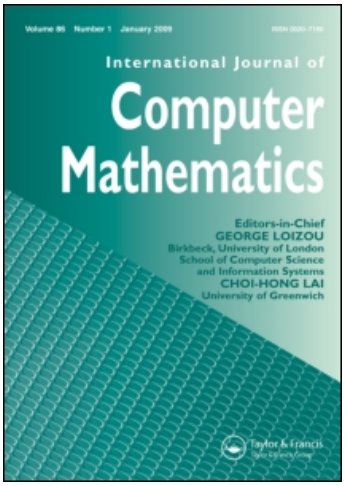

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713455451

Applying a testing approach to the Calculus of Fair Ambients

Xiaojuan Cai[a]; Han Zhu[a]
[a] Basics Lab, Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China

Online publication date: 10 December 2009

To cite this Article Cai, Xiaojuan and Zhu, Han(2009) 'Applying a testing approach to the Calculus of Fair Ambients', International Journal of Computer Mathematics, 86: 12, 2040 — 2060
To link to this Article: DOI: 10.1080/00207160903243148
URL: http://dx.doi.org/10.1080/00207160903243148

International Journal of Computer Mathematics
Vol. 86, No. 12, December 2009, 2040–2060

Taylor & Francis
Taylor & Francis Group

Applying a testing approach to the Calculus of Fair Ambients

Xiaojuan Cai* and Han Zhu

Basics Lab, Department of Computer Science, Shanghai Jiao Tong University, Shanghai, China

(*Received 4 November 2008; revised version received 20 April 2009; second revision received 20 July 2009; final version received 3 August 2009*)

We apply a testing approach to the Calculus of Fair Ambients and investigate the resulting testing equivalence. We prove that variant conditions on its definition do not change its discriminating power, and it is congruent on finite processes. On a proper subset of processes, open bisimilarity is strictly included in testing equivalence. It is also proved that the translation from Pi-Calculus to Fair Ambients is fully abstract with respect to testing equivalence.

Keywords: theoretical computer science; concurrency theory; process calculus; fair ambients; testing equivalence

2000 AMS Subject Classification: 68Q85

1. Introduction

Process calculi are mathematical models for describing and analysing properties of distributed concurrent systems. Formal studies have shown that concurrent processes exhibit far more complex behaviour than traditional sequential programs. For every process calculus, bisimulation equivalence provides a canonical equivalence relation. These bisimulation equivalence relations are subject to a fine-grained examination by the observers. They are often too strong to be suitable for many applications.

In practice, the trace of actions, the timing of choices and divergence are the factors of the most concern. The testing approach pioneered by De Nicola and Hennessy [7] addresses these issues at a moderate level. It is probably the best known non-bisimulation equivalence for processes. In the testing approach, we care only about the *result* of the observation, not the *course* of the observation. Informally, an observer is an ordinary process and a test is the set of computations. The result of a test is indicated by whether or not computations are successful. Two processes are identified if, and only if, no observer can tell any difference between them by the results of tests. This simple machinery has been applied to many process calculi [1,8,9].

The Calculus of Mobile Ambients, MA for short, is introduced by Cardelli and Gordon [3–5] as a model that provides a uniform account for both mobile computation and mobile computing [2].

*Corresponding author. Email: cxj@sjtu.edu.cn

ISSN 0020-7160 print/ISSN 1029-0265 online

DOI: 10.1080/00207160903243148
http://www.informaworld.com

The apparatus introduced by MA is that of ambients; an ambient is a program found at a specified location. An ambient may move from one ambient to another. Several variants of MA, such as Safe Ambients [16], Safe Ambients with Passwords [17], Controlled Ambients [24] and ROAM (The Calculus of Robust Ambients) [14] to name a few, have been proposed to enhance the control power of MA. The Calculus of Fair Ambients [11], or FA, imposes strict rules on when ambients can interact. The strict rules are designed to stick to the following principles:

(1) First class citizenship: All actions are interactions between ambients.
(2) Authorization: The two ambients involved in an interaction must obtain the authorizations from each other.
(3) Authentication: The two ambients involved in an interaction must know each other's identities.

It is proved in [11] that the π-calculus [21] is a sub-calculus of FA.

We apply a may/must-testing approach to FA from an interactive point of view. The observers for FA are a little bit more involved than those for the π-calculus, because FA does not expose a flat structure and interactions between ambients are subject to the changes of ambient structure. Since we need to test all the behaviours of an ambient when it is located in another ambient, we must use ambient contexts $C[\_]$ as observers.

It turns out that a special form of observers $o[\_|O]|O'$ have the same discriminating power as all observers, just like the context lemma [13]. Our result is more interesting than that of Cardelli and Gordon since it holds for testing equivalence, which is the intersection of may-equivalence and must-equivalence. To achieve this, we should exploit the control power provided by FA semantics skillfully. By some carefully designed facility ambients, we make sure that the result sets are equal and there are no unwanted additional computations.

We also show that two definitions of successful states, one from a structural viewpoint and the other from an interactive viewpoint, result in the same relation. In further studies, testing equivalence proves to be congruent on finite FA processes and strictly includes bisimilarity on a subset of processes.

Finally, we take a look at the robustness of our definition of testing equivalence on FA by comparing it to the testing equivalence for the π-calculus. The correspondence between the actions of a π-process P and its translation $[\![P]\!]$ in FA is established. We give a full abstraction result w.r.t. testing equivalence for the encoding from the π-calculus to FA.

1.1 *Related work*

Many equivalence relations have been proposed to give ambient calculi a reasonable semantic equivalence. The original MA is based on reduction semantics. Gordon and Cardelli develop a theory of Morris-style contextual equivalence in [6,13]. It is essentially a variant form of may-testing. Also based on reduction semantics, the notion of barbed congruence – a contextually defined, bisimulation-based relation – is proposed in [16,25] to be a behavioural equality. Based on the labelled transition system (LTS), standard bisimulation equivalence relations are established [11,18,19]. Recently, Rathke and Sobocinski [22] have developed a systematic procedure for deriving LTSs in the structural style from the underlying reduction semantics and observability for MAs. Toru Kato [15] has given an extension of Cardelli and Gordon's type system [6] and presented an equivalence relation for the extended typed ambient calculus.

1.2 *Outline of the paper*

This paper is structured as follows. Section 2 provides some background of the Calculus of FAs. Definitions of testing equivalence on FA and the coincidence of variant conditions are

established in Section 3. Section 4 discusses the relationship between testing equivalence and open bisimilarity, and then examines the congruence property. The fully abstract encoding from the π-calculus to FA is studied in Section 5. Finally Section 6 concludes by pointing out some future research directions. Additional information can be found in appendices.

2. The calculus of Fair Ambients

FA differs from MA in that the semantics imposes strict conditions on the possible interactions through the use of actions and co-actions. Its semantics is defined purely in terms of a labelled transition system, whose rules are carefully chosen to adhere to the three principles.

We presume a countable set of names $\mathcal{N}$, ranged over by $a, b, c, \ldots, x, y, z, \ldots$ and their decorated forms. We write $\tilde{x}$ for some finite set of names $\{x_1, \ldots, x_n\}$. The abstract syntax of processes is defined by the following grammar:

$$P ::= \mathbf{0} \mid \kappa.P \mid P \mid P \mid a[P] \mid (a)P \mid !P$$

where κ is a moving capability or communicating capability:

$$\kappa ::= \text{in } a \mid \overline{\text{in}}\, a \mid \text{out } a \mid \overline{\text{out}}\, a \mid \text{open } a \mid \overline{\text{open}}\, a \mid a(x) \mid \bar{a}x$$

Interactions between ambients are essentially higher order. FA uses concretion and contexts to avoid higher order labels. *Concretions* have the form of $\langle P \rangle P'$ where $\langle P \rangle$ is the moving part and P' the remaining part. *Contexts* are built up by the following BNF:

$$C[\_] ::= \_ \mid C[\_] \mid P \mid P \mid C[\_] \mid (a)C[\_] \mid a[C[\_]]$$

Ambient contexts are contexts of the form $a[C[\_]]$. In any ambient context there exists exactly one hole.

Action labels of operational semantics are defined as:

$\lambda ::=$	Informal action meanings:
$\kappa \mid \tau$	capabilities and τ-actions
$\mid a[\text{in } b]$	a moves into b
$\mid a[\overline{\text{in}}\, b]$	a imports b
$\mid [\text{out } b]$	moves out to reach b
$\mid a[[\text{out } b]]$	an ambient moves out of a to reach b
$\mid a[\overline{\text{out}}\, b]$	a extracts an ambient out of b
$\mid a[\text{open } b]$	a opens up b
$\mid a[\overline{\text{open}}\, b]$	a is opened up by b
$\mid \bar{a}(x)$	exports the bound name x to a
$\mid a[bx]$	a imports the name x from b
$\mid a[\bar{b}x]$	a exports the name x to b
$\mid (x)a[\bar{b}x]$	a exports the bound name x to b

The rules defining the labelled transition system can be divided into four groups.

(1) Structural rules:

$$\frac{}{\kappa.P \xrightarrow{\kappa} P} \qquad \frac{P \xrightarrow{\lambda} U}{P \mid Q \xrightarrow{\lambda} U \mid Q} \qquad \frac{P \xrightarrow{\tau} P'}{a[P] \xrightarrow{\tau} a[P']}$$

$$\frac{P \xrightarrow{\lambda} U \quad n \text{ is not in } \lambda}{(n)P \xrightarrow{\lambda} (n)U} \qquad \frac{P \mid !P \xrightarrow{\lambda} U}{!P \xrightarrow{\lambda} U}$$

(2) Ambient rules:

$$\frac{P \xrightarrow{\text{in}\, b} P'}{a[P] \xrightarrow{a[\text{in}\, b]} \langle a[P']\rangle \mathbf{0}} \qquad \frac{P \xrightarrow{\overline{\text{in}}\, b} P'}{a[P] \xrightarrow{a[\overline{\text{in}}\, b]} a[\_ \mid P']}$$

$$\frac{P \xrightarrow{\text{out}\, b} P'}{a[P] \xrightarrow{[\text{out}\, b]} \langle a[P']\rangle \mathbf{0}} \qquad \frac{P \xrightarrow{[\text{out}\, b]} (\tilde{m})\langle A\rangle P'}{a[P] \xrightarrow{a[[\text{out}\, b]]} (\tilde{m})(A \mid a[P'])} \qquad \frac{P \xrightarrow{\overline{\text{out}}\, b} P'}{a[P] \xrightarrow{a[\overline{\text{out}}\, b]} a[P']}$$

$$\frac{P \xrightarrow{\text{open}\, b} P'}{a[P] \xrightarrow{a[\text{open}\, b]} a[\_ \mid P']} \qquad \frac{P \xrightarrow{\overline{\text{open}}\, b} P'}{a[P] \xrightarrow{a[\overline{\text{open}}\, b]} \langle P'\rangle \mathbf{0}}$$

(3) Interaction rules:

$$\frac{P \xrightarrow{b[\text{in}\, a]} (\tilde{m})\langle B\rangle P' \quad Q \xrightarrow{a[\overline{\text{in}}\, b]} C[\_]}{P \mid Q \xrightarrow{\tau} (\tilde{m})(P' \mid C[B])} \qquad \frac{P \xrightarrow{b[[\text{out}\, a]]} P' \quad Q \xrightarrow{a[\overline{\text{out}}\, b]} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P \xrightarrow{a[\text{open}\, b]} C[\_] \quad Q \xrightarrow{b[\overline{\text{open}}\, a]} (\tilde{m})\langle Q''\rangle Q'}{P \mid Q \xrightarrow{\tau} (\tilde{m})(C[Q''] \mid Q')}$$

(4) Communication rules:

$$\frac{}{a(x).P \xrightarrow{ay} P\{y/x\}} \qquad \frac{}{\bar{a}x.P \xrightarrow{\bar{a}x} P} \qquad \frac{P \xrightarrow{\bar{a}x} P'}{(x)P \xrightarrow{\bar{a}(x)} P'}$$

$$\frac{P \xrightarrow{ax} P'}{b[P] \xrightarrow{b[ax]} b[P']} \qquad \frac{P \xrightarrow{\bar{a}(x)} P'}{b[P] \xrightarrow{(x)b[\bar{a}x]} b[P']} \qquad \frac{P \xrightarrow{\bar{a}x} P'}{b[P] \xrightarrow{b[\bar{a}x]} b[P']} \qquad \frac{P \xrightarrow{b[\bar{a}x]} P'}{(x)P \xrightarrow{(x)b[\bar{a}x]} P'}$$

$$\frac{P \xrightarrow{b[ax]} P' \quad Q \xrightarrow{a[\bar{b}x]} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{b[ax]} P' \quad Q \xrightarrow{(x)a[\bar{b}x]} Q' \quad x \notin fn(P)}{P \mid Q \xrightarrow{\tau} (x)(P' \mid Q')}$$

We write $\rightarrow$ for $\xrightarrow{\tau}$, and the reflexive and transitive closure of $\rightarrow$ is denoted by $\Rightarrow$. The meaning of $\overset{\lambda}{\Rightarrow}$ is $\Rightarrow\xrightarrow{\lambda}\Rightarrow$ if $\lambda \neq \tau$, and is $\Rightarrow$ if $\lambda = \tau$. We sometimes write $P \sim \Rightarrow \approx Q$ in case $\exists P', P''$ s.t. $P \sim P' \Rightarrow P'' \approx Q$. The set of free names in P is denoted by $fn(P)$, which is defined in a standard manner.

FA uses the standard quasi-open style bisimulation method as its equivalence relation. Readers are advised to consult Fu [10] and Sangiorgi and Walker [23] for quasi-open bisimilarity for the π-calculus. Similar to quasi-open bisimilarity for the π-calculus, open bisimilarity for FA involves substitutions of names.

A substitution σ of names, or simply *substitution*, is a map from $\mathcal{N}$ to $\mathcal{N}$ such that the set $\{x \mid x \neq \sigma(x)\}$ is finite. A substitution σ *respects* $\tilde{x}$ if $\forall x \in \tilde{x}.x\sigma = x$ and $\forall y \notin \tilde{x}.y\sigma \notin \tilde{x}$.

DEFINITION 2.1 *A family of symmetric relations* $\{\mathcal{R}^{\tilde{z}}\}_{\tilde{z}\subseteq_f\mathcal{N}}$ *on processes is an* open bisimulation *if whenever* $P\,\mathcal{R}^{\tilde{z}}\,Q$ *and* σ *respects* $\tilde{z}$ *then* $P\sigma\,\mathcal{R}^{\tilde{z}}\,Q\sigma$ *and the following properties hold*:

(1) *If* $P \xrightarrow{\tau} P'$ *then* $Q \Rightarrow Q'\,\mathcal{R}^{\tilde{z}}\,P'$ *for some* Q';
(2) *If* $P \xrightarrow{\kappa} P'$ *and* κ *is not of the form* $\bar{a}(x)$, *then* $Q \overset{\kappa}{\Rightarrow} Q'\,\mathcal{R}^{\tilde{z}}\,P'$ *for some* Q';
(3) *If* $P \xrightarrow{\bar{a}(x)} P'$ *then* $Q \overset{\bar{a}(x)}{\Rightarrow} Q'\,\mathcal{R}^{\tilde{z}x}\,P'$ *for some* Q';
(4) *If* $P \xrightarrow{\lambda} P'$ *and* λ *is a label in the set* $\{a[\overline{\text{out}}\,b], a[[\text{out}\,b]], a[bx], a[\bar{b}x] \mid a, b \in \mathcal{N}\}$ *then* $Q \overset{\lambda}{\Rightarrow} Q'\,\mathcal{R}^{\tilde{z}}\,P'$ *for some* Q';
(5) *If* $P \xrightarrow{(x)a[\bar{b}x]} P'$ *then* $Q \overset{(x)a[\bar{b}x]}{\Rightarrow} Q'\,\mathcal{R}^{\tilde{z}x}\,P'$ *for some* Q';
(6) *If* $P \xrightarrow{[\text{out}\,a]} (\tilde{m})(\langle A\rangle P')$ *then for every name* d *and every process* O *satisfying* $\{\tilde{m}\}\cap fn(O)=\emptyset$ *there exist some* $\tilde{n}, B, Q', Q''$ *such that* $Q \Rightarrow \xrightarrow{[\text{out}\,a]} (\tilde{n})(\langle B\rangle Q')$ *and* $(\tilde{n})(B \mid d[Q' \mid O]]) \Rightarrow Q''\,\mathcal{R}^{\tilde{z}}\,(\tilde{m})(A \mid d[P' \mid O])$;
(7) *If* $P \xrightarrow{a[\text{in}\,b]} (\tilde{m})(\langle A\rangle P')$ *then for every* O *satisfying* $\{\tilde{m}\}\cap fn(O)=\emptyset$ *some* $\tilde{n}, B, Q', Q''$ *exist such that* $Q \Rightarrow \xrightarrow{a[\text{in}\,b]} (\tilde{n})(\langle B\rangle Q')$ *and* $(\tilde{n})(Q' \mid b[B \mid O]) \Rightarrow Q''\,\mathcal{R}^{\tilde{z}}\,(\tilde{m})(P' \mid b[A \mid O])$;
(8) *If* $P \xrightarrow{a[\overline{\text{in}}\,b]} (\tilde{m})(\langle M\rangle P')$ *then for every* O *satisfying* $\{\tilde{m}\}\cap fn(O)=\emptyset$ *there exist some* $\tilde{n}, N, Q', Q''$ *such that* $Q \Rightarrow \xrightarrow{a[\overline{\text{in}}\,b]} (\tilde{n})(\langle N\rangle Q')$ *and* $(\tilde{n})(Q' \mid b[N \mid O]]) \Rightarrow Q''\,\mathcal{R}^{\tilde{z}}\,(\tilde{m})(P' \mid b[M \mid O])$;
(9) *If* $P \xrightarrow{a[\overline{\text{in}}\,b]} C[\_]$ *then for every process* N *satisfying* $fn(N)\cap bn(C[\_])=\emptyset$ *there exist some* $C'[\_]$, Q' *such that* $Q \Rightarrow \xrightarrow{a[\overline{\text{in}}\,b]} C'[\_]$ *and* $C'[b[N]] \Rightarrow Q'\,\mathcal{R}^{\tilde{z}}\,C[b[N]]$;
(10) *If* $P \xrightarrow{a[\text{open}\,b]} C[\_]$ *then for every process* N *satisfying* $fn(N)\cap bn(C[\_])=\emptyset$ *there exist some* $C'[\_]$, Q' *such that* $Q \Rightarrow \xrightarrow{a[\text{open}\,b]} C'[\_]$ *and* $C'[N] \Rightarrow Q'\,\mathcal{R}^{\tilde{z}}\,C'[N]$.

We write $\{\approx^{\tilde{z}}\}_{\tilde{z}\subseteq_f\mathcal{N}}$ *for the largest open bisimulation and refer to* $\approx^{\tilde{z}}$ *as the* open $\tilde{z}$-bisimilarity. Open bisimilarity $\approx$ *is the open* $\emptyset$*-bisimilarity* $\approx^{\emptyset}$.

The definitions of *strong open bisimilarity* which is denoted by $\sim$ and *open bisimilarity up to* $\sim$ can be found in [11].

The most important property of open bisimilarity is local observability [11].

THEOREM 2.1 (Local observability) *For FA processes* P *and* Q, *it holds that* $P \approx Q$ *if, and only if,* $C[P] \approx C[Q]$ *for every ambient context* $C[\_]$.

This property enables us to figure out the relationship between testing equivalence and open bisimilarity, which will be studied in Section 4.

3. Testing equivalence on FA

We give the definition of testing equivalence on FA in this section.

Ambient calculi are distinguished in many aspects, one of which is that ambients have nested structures. In the π-calculus, all processes are essentially concurrent to each other. In FA the ambient $a[P \mid Q]$ for example indicates that $P \mid Q$ is located at a. Moreover P and Q may contain further locations, giving rise to a tree-like structure. Processes are situated at different

nodes of this tree. Only those in the same node can interact with each other. For example

$$a[b[\text{in}\, c.P] \mid c[\overline{\text{in}}\, b.Q]] \mid c[R] \xrightarrow{\tau} a[c[b[P] \mid Q]] \mid c[R],$$

but

$$a[b[\text{in}\, c.P] \mid c[Q]] \mid c[\overline{\text{in}}\, b.R] \stackrel{\tau}{\not\to} a[c[Q]] \mid c[b[P] \mid R].$$

This structure and the fact that moving capabilities can change this structure dynamically in a computation force us to consider a process in all ambient contexts.

DEFINITION 3.1 *The central notions of the testing approach are defined as follows.*

- Observers *are ambient contexts which have at most one ambient with the special name* ω. *Ambient observers are denoted by* $C_o[\_], \ldots$
- Experiments *are conducted by composing the process into the hole of the observer*:

$$C_o[P], \ldots$$

- Computations *are any nonempty sequences of states, either finite or infinite, linked by* τ *actions*:

$$C_o[P] \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_n \stackrel{\tau}{\not\to}$$

  or

$$C_o[P] \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_n \xrightarrow{\tau}$$

  The set of all computations of an experiment $C_o[P]$ *is denoted by* $c(C_o[P])$.
- Successful states *are states with some predefined special property. A state is successful if it can perform the action* $\omega[\text{open}\, \omega]$*:*

$$P_m \xrightarrow{\omega[\text{open}\, \omega]}$$

- *If a computation contains a successful state it is also called successful, otherwise unsuccessful. The result set of an experiment* $C_o[P]$, *denoted by* $r(C_o[P])$, *is determined by the following properties:*
  - $\top \in r(C_o[P])$ *if there exists a successful computation in* $c(C_o[P])$;
  - $\bot \in r(C_o[P])$ *if there exists an unsuccessful computation in* $c(C_o[P])$.

  Obviously any result set is a subset of $\{\top, \bot\}$.
- *We say that* P ***may*** $C_o[\_]$ *if, and only if,* $\top \in c(C_o[P])$ *and* P ***must*** $C_o[\_]$ *if, and only if,* $c(C_o[P]) = \{\top\}$.

DEFINITION 3.2 *Three* ground testing equivalence *relations are defined as follows.*

(1) *Ground may equivalence* $P \approx_{\textbf{may}} Q$*: For every observer* $C_o[\_]$, P ***may*** $C_o[\_]$ *if, and only if,* Q ***may*** $C_o[\_]$.
(2) *Ground must equivalence* $P \approx_{\textbf{must}} Q$*: For every observer* $C_o[\_]$, P ***must*** $C_o[\_]$ *if, and only if,* Q ***must*** $C_o[\_]$.
(3) *Ground testing equivalence*: $P \approx_{\textbf{test}} Q$ *if, and only if, both* $P \approx_{\textbf{may}} Q$ *and* $P \approx_{\textbf{must}} Q$.

FA is a calculus of mobile processes, having the ability of instantiating dummy names dynamically. Therefore closure under substitution must also be imposed.

DEFINITION 3.3 *Three* testing equivalence *relations are defined as follows:*

(1) $P =_{\textbf{may}} Q$ *if, and only if,* $P\sigma \approx_{\textbf{may}} Q\sigma$ *for every substitution* σ.

(2) $P =_{\textbf{must}} Q$ *if, and only if,* $P\sigma \approx_{\textbf{must}} Q\sigma$ *for every substitution* σ.
(3) $P =_{\textbf{test}} Q$ *if, and only if,* $P\sigma \approx_{\textbf{test}} Q\sigma$ *for every substitution* σ.

Definitions from different viewpoints are not always different, as the following result indicates; the structural view and the interactive view of successful states coincide.

THEOREM 3.1 (Success) *The following two definitions of successful states lead to the same testing equivalence*:

(1) $P_m \xrightarrow{\omega[\text{open}\,\omega]}$.
(2) $P_m \equiv \omega[R] \mid R'$ *for some* R *and* R'.

Proof Let $r^*(C_o[P])$ be the result set of experiment $C_o[P]$ using the second definition, and $=^*_{\textbf{test}}$ be the resulting testing equivalence relation. We will prove that

$$\forall P, Q : (\forall C_o[\_] : r(C_o[P]) = r(C_o[Q]) \iff \forall C_o[\_] : r^*(C_o[P]) = r^*(C_o[Q])),$$

which then implies

$$\forall P, Q : (P =_{\textbf{test}} Q \iff P =^*_{\textbf{test}} Q).$$

'$\Longrightarrow$': By contradiction. Suppose there exists an observer $C_o[\_]$ such that

$$r^*(C_o[P]) \neq r^*(C_o[Q]).$$

Without loss of generality, there are two cases:

- $\top \in r^*(C_o[P])$ and $\top \notin r^*(C_o[Q])$
  In this case, we know that in a computation of $c(C_o[P])$ there is a successful state, which can do an $\omega[\text{open}\,\omega]$ action, and obviously ambient ω is in its top level. Since $\omega[\text{open}\,\omega]$ does not have its complementary actions in $C_o[\_]$, prefix open ω will remain in ambient ω. If a state has ambient ω in its top level, it definitely can do $\omega[\text{open}\,\omega]$. But since $\top \notin r^*(C_o[Q])$, no state in each computation of $C_o[Q]$ has ambient ω in its top level. Therefore $\top \notin r(C_o[Q])$ and $\top \in r(C_o[P])$.
- $\bot \in r^*(C_o[P])$ and $\bot \notin r^*(C_o[Q])$
  The argument of this case is similar to the above.

'$\Longleftarrow$': Also by contradiction. Suppose there exists an observer $C_o[\_]$ such that

$$r(C_o[P]) \neq r(C_o[Q]),$$

then we construct an observer $C^*_o[\_]$ such that

$$r(C^*_o[P]) \neq r(C^*_o[Q]).$$

Again, without loss of generality, there are two cases:

- $\top \in r(C_o[P])$ and $\top \notin r(C_o[Q])$
  In this case, the successful state has ambient ω in its top level. If this ambient can do an $\omega[\text{open}\,\omega]$ action, then we let $C^*_o[\_] \equiv C_o[\_]$. Otherwise, suppose the ambient ω in it has the form $\omega[R]$, we modify $\omega[R]$ to $\omega[R \mid \text{open}\,\omega]$ since we want the action $\omega[\text{open}\,\omega]$ to be observable. Let $C^*_o[\_]$ be the resulting observer, then surely we have $\top \in r^*(C_o[P])$ and $\top \notin r^*(C_o[Q])$.
- $\bot \in r(C_o[P])$ and $\bot \notin r(C_o[Q])$.
  A similar argument points out that the required observer $C^*_o[\_]$ can be easily constructed. ■

Remark 1 If we deem the theory of concurrent processes to be a theory of interaction, a successful state should have some observable properties. An empty ambient is equal to the null process

$$\omega[\mathbf{0}] \approx \mathbf{0}$$

though it has ambient ω in its top level. A special action has more meaning than merely a special ambient in the top level. We prefer the definition that refers to interactions to the structural definition, as it is more consistent with the theory of interaction.

As stated before, to test a process we must place it in an ambient context. Does there exist a subset of observers such that they are sufficient to have the full discriminating power? The next theorem provides a positive answer.

THEOREM 3.2 (Context) *The following two definitions of observers lead to the same testing equivalence:*

(1) *The set of all observers.*
(2) *The set of those observers in the form of* $o[\_ \mid O] \mid O'$.

The main point of the proof of this theorem is to explain why this set of special observers has the full discriminating power which the set of all observers has. For processes P, Q, if an observer $C[\_]$ exists such that $r(C[P]) \neq r(C[Q])$, then we are going to construct an ambient context of the form $C_o[\_] \equiv o[\_ \mid O] \mid O'$ such that $r(C_o[P]) \neq r(C_o[Q])$.

Intuitively, O' has a tree structure similar to $C[\_]$ and for every FA process R the role of O is to pack R and then transport it to its destination in O' where the hole '_' lies with the help of O'. The system evolves into $C[R]$ finally. The three major steps are:

(1) R is *packed up* into the form $n_1[n_2[o[R]]]$ in order to prevent any interaction between R and O' before R is in its right place;
(2) $n_1[n_2[o[R]]]$ *travels* to R's destination in O';
(3) n_1, n_2, o are *opened up* one by one and the whole system will do computations as $C[R]$ will, producing the same result set.

All these are done by a careful design of the ambients that constitute the implementation to ensure that the packed R is incapable of interacting with other parts of the observer and to avoid introducing unwanted computations, which may change the results of the experiment.

Proof Let $=^*_{\mathbf{test}}$ be the resulting testing equivalence relation using the first definition of observers. It is straightforward that $=^*_{\mathbf{test}}$ implies $=_{\mathbf{test}}$, which can be stated as

$$\forall P, Q: (P =^*_{\mathbf{test}} Q \Longrightarrow P =_{\mathbf{test}} Q).$$

We will show the reverse direction by contradiction

$$\forall P, Q: (P =^*_{\mathbf{test}} Q \Longleftarrow P =_{\mathbf{test}} Q). \tag{1}$$

Equation (1) can be written as

$$\forall P, Q: (\forall C[\_] : r(C[P]) = r(C[Q]) \Longleftarrow \forall C_o[\_] : r(C_o[P]) = r(C_o[Q])).$$

Suppose there exists an ambient context $C[\_]$ such that

$$r(C[P]) \neq r(C[Q]),$$

we construct an ambient context in the form of $C_o[\_] \equiv o[\_ \mid O] \mid O'$ such that

$$r(o[P \mid O] \mid O') \neq r(o[Q \mid O] \mid O').$$

If $C[\_]$ has the form of $o[\_ \mid O] \mid O'$, we are done. Otherwise we construct O and O' according to the structure of $C[\_]$ in order to make the following statement valid,

$$\forall R\colon r(o[R \mid O] \mid O') = r(C[R]). \tag{2}$$

It follows that

$$r(o[P \mid O] \mid O') = r(C[P]) \neq r(C[Q]) = r(o[Q \mid O] \mid O').$$

More specifically, $o[\_ \mid O] \mid O'$ is constructed in the following manner:

$$\begin{aligned}
O &\equiv \text{in}\, n_1.\text{in}\, n_2.\text{out}\, n_3.OPEN \\
O' &\equiv n_1[N_1] \mid C^* \\
n_1[N_1] &\equiv n_1[OPEN \mid \overline{\text{in}}\, o.n_2[N_2] \mid s_1[S_1] \mid IN] \\
n_2[N_2] &\equiv n_2[\text{out}\, n_3.OPEN \mid \overline{\text{in}}\, o.s_2[S_2]] \\
s_1[S_1] &\equiv s_1[\overline{\text{out}}\, n_2.OUT] \\
s_2[S_2] &\equiv s_2[\text{out}\, s_1].
\end{aligned}$$

Names $o, n_1, n_2, n_3, s_1, s_2$ are fresh names. Ambients IN, OUT, OPEN and C^* are constructed according to the place of '_' in $C[\_]$. Since ambient contexts are of the form $a[C[\_]]$ and the case of $o[\_ \mid O] \mid O'$ is trivial, we now investigate the case that '_' is nested in at least two ambients. If '_' is nested in at least three ambients, without loss of generality we can write $C[\_]$ in the following form:

$$a[A \mid \cdots b[B \mid c[C' \mid \_]] \cdots] \mid A',$$

where the ellipsis '$\cdots$' stands for the nested ambients. The construction of IN, OUT, OPEN are

$$\begin{aligned}
\text{IN} &\equiv \text{in}\, a.\cdots \text{in}\, b \\
\text{OUT} &\equiv \text{out}\, a \\
\text{OPEN} &\equiv \overline{\text{open}}\, c
\end{aligned}$$

and C^* is constructed by modifying $C[\_]$. The modification process $C[\_] \mapsto C^*$ is

$$\begin{aligned}
a[A \mid \ldots] &\longmapsto a[\overline{\text{out}}\, n_1.\overline{\text{in}}\, n_1.(A \mid \ldots)] \\
&\vdots \\
b[B \mid c[\ldots]] &\longmapsto b[\overline{\text{in}}\, n_1.(B \mid c[\ldots] \mid n_3[N_3])] \\
c[C' \mid _] &\longmapsto c[\text{open}\, n_1.\text{open}\, n_2.\text{open}\, o.C'] \\
n_3[N_3] &\equiv n_3[\overline{\text{out}}\, c.\overline{\text{out}}\, c].
\end{aligned}$$

Here '...' represents the omitted contents in ambients. If '_' is nested in exactly two ambients, that is $C[\_] \equiv a[A \mid c[C' \mid \_]] \mid A'$, then the construction is

$$\begin{aligned} \text{IN} &\equiv \text{in}\, a \\ \text{OUT} &\equiv \text{out}\, a \\ \text{OPEN} &\equiv \overline{\text{open}}\, c \\ C^* &\equiv a[\overline{\text{out}}\, n_1.\overline{\text{in}}\, n_1.(A \mid c[\text{open}\, n_1.\text{open}\, n_2.\text{open}\, o.C'] \mid n_3[\overline{\text{out}}\, c.\overline{\text{out}}\, c])] \mid A'. \end{aligned}$$

We now comment on this construction. Ambients n_1, n_2 are here to pack $o[R]$. Ambients s_1, s_2 are used to send to C^* a signal that o has been in n_2, then n_1 can start to travel. Ambient n_3 is in parallel with the ambient where '_' is in, helping it open n_1, n_2 and o in turn. We can easily verify that

$$\forall R\colon C_o[R] \equiv o[R \mid O] \mid O' \Longrightarrow C[R]. \tag{3}$$

The rest of our work is to prove Equation (2). Since we have Equation (3), it follows that

$$\forall R\colon r(C_o[R]) \supseteq r(C[R]).$$

To establish the opposite direction

$$\forall R\colon r(C_o[R]) \subseteq r(C[R]),$$

the most crucial observation is that R will eventually be in its right place of $C[\_]$. This is because in its path to destination, the processes in each ambient are prefix guarded. The ambients along the path $(a, \ldots, b, c)$ are of the following form:

$$\begin{gathered} a[\overline{\text{out}}\, n_1.\overline{\text{in}}\, n_1.(A \mid \ldots)] \\ \vdots \\ b[\overline{\text{in}}\, n_1.(B \mid c[\ldots] \mid n_3[N_3])] \\ c[\text{open}\, n_1.\text{open}\, n_2.\text{open}\, o.C']. \end{gathered}$$

These prefixes prevent the ambients along this path doing moving capabilities, thereby protecting the path from changing. By the fact that in the modification of $C[\_]$ to C^* only ambients along the path are modified, we can easily check that if

$$o[R \mid O] \mid O' \Longrightarrow o[R \mid O] \mid O'' \Longrightarrow C'[R],$$

then we must have

$$C[R] \Longrightarrow C''[R] \Longrightarrow C'[R]$$

such that if $O'' \xrightarrow{\omega[\text{open}\,\omega]}$ then $C''[R] \xrightarrow{\omega[\text{open}\,\omega]}$ for some $C''[\_]$, and vice versa.

Now we have

$$\forall R : r(C_o[R]) \subseteq r(C[R]).$$

We are done. ■

The following example shows how $C_o[\_]$ is constructed and how the construction works.

Example 3.3 (An illustrative running example) Suppose

$$C[\_] \equiv a[A \mid b[B \mid c[C' \mid \_]]].$$

We now construct an observer $o[\_ \mid O] \mid O'$ following the method described in the proof of Theorem 3.5

$$\begin{aligned}
O &\equiv \text{in}\, n_1.\text{in}\, n_2.\text{out}\, n_3.\overline{\text{open}}\, c \\
O' &\equiv n_1[N_1] \mid C^* \\
n_1[N_1] &\equiv n_1[\overline{\text{open}}\, c \mid \overline{\text{in}}\, o.n_2[N_2] \mid s_1[S_1] \mid \text{in}\, a.\text{in}\, b] \\
n_2[N_2] &\equiv n_2[\text{out}\, n_3.\overline{\text{open}}\, c \mid \overline{\text{in}}\, o.s_2[S_2]] \\
s_1[S_1] &\equiv s_1[\overline{\text{out}}\, n_2.\text{out}\, a] \\
s_2[S_2] &\equiv s_2[\text{out}\, s_1] \\
C^* &\equiv a[\overline{\text{out}}\, n_1.\overline{\text{in}}\, n_1.(A \mid b[B'])] \\
b[B'] &\equiv b[\overline{\text{in}}\, n_1.(B \mid c[C''] \mid n_3[N_3])] \\
c[C''] &\equiv c[\text{open}\, n_1.\text{open}\, n_2.\text{open}\, o.C'] \\
n_3[N_3] &\equiv n_3[\overline{\text{out}}\, c.\overline{\text{out}}\, c].
\end{aligned}$$

The computation from $C_o[R]$ to $C[R]$ can be depicted as follows. For simplicity, only the ambient structure is shown; their contents are omitted if unnecessary.

$$\begin{aligned}
C_o[R] &\equiv o[R \mid O] \mid n_1 \mid a \\
&\xrightarrow{o[\text{in}\, n_1], n_1[\overline{\text{in}}\, o]} n_1[o[R \mid \text{in}\, n_2.\text{out}\, n_3.\overline{\text{open}}\, c] \mid n_2 \mid s_1] \mid a \\
&\xrightarrow{o[\text{in}\, n_2], n_2[\overline{\text{in}}\, o]} n_1[n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c] \mid s_2] \mid s_1] \mid a \\
&\xrightarrow{n_2[[\text{out}\, s_1]], s_1[\overline{\text{out}}\, n_2]} n_1[n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]] \mid s_1] \mid a \\
&\xrightarrow{n_1[[\text{out}\, a]], a[\overline{\text{out}}\, n_1]} n_1[n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]]] \mid a[\overline{\text{in}}\, n_1.(A \mid b)] \\
&\xrightarrow{n_1[\text{in}\, a], a[\overline{\text{in}}\, n_1]} a[A \mid b \mid n_1[n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]]]] \\
&\xrightarrow{n_1[\text{in}\, b], b[\overline{\text{in}}\, n_1]} a[A \mid b[B \mid c \mid n_3 \mid n_1[n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]]]]] \\
&\xrightarrow{c[\text{open}\, n_1], n_1[\overline{\text{open}}\, c]} a[A \mid b[B \mid c[\text{open}\, n_2.\text{open}\, o.C' \mid n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]]] \mid n_3]] \\
&\xrightarrow{c[[\text{out}\, n_3]], n_3[\overline{\text{out}}\, c]} a[A \mid b[B \mid c[\text{open}\, n_2.\text{open}\, o.C'] \mid n_2[o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]] \mid n_3]] \\
&\xrightarrow{c[\text{open}\, n_2], n_2[\overline{\text{open}}\, c]} a[A \mid b[B \mid c[\text{open}\, o.C' \mid o[R \mid \text{out}\, n_3.\overline{\text{open}}\, c]] \mid n_3]] \\
&\xrightarrow{c[[\text{out}\, n_3]], n_3[\overline{\text{out}}\, c]} a[A \mid b[B \mid c[\text{open}\, o.C'] \mid o[R \mid \overline{\text{open}}\, c]]] \\
&\xrightarrow{c[\text{open}\, o], o[\overline{\text{open}}\, c]} a[A \mid b[B \mid c[C' \mid R]]] \equiv C[R]
\end{aligned}$$

We can easily check that $\forall R\colon r(C_o[R]) \subseteq r(C[R])$.

The definition of testing equivalence and variant conditions on it are shown in this section. In the next section we will investigate some properties of testing equivalence.

4. Relationship with open bisimilarity and congruence

This section is dedicated to the study of some properties of testing equivalence.

We now take a look at the relationship between testing equivalence and bisimulation equivalence. As stated before, open bisimilarity is somewhat too strong in practice, but it is the standard equivalence relation in most process calculi. For a newly proposed equivalence relation, its relation to open bisimilarity is of interest.

A process P is called *divergent* if it has an infinite computation. It is *terminating* if it is not divergent.

Testing equivalence assumes that a divergent computation without direct ability to fire the special action is a failure, whereas open bisimilarity ignores the divergent computation. It follows that open bisimilarity is not a subset of $=_{\mathbf{test}}$, and obviously $=_{\mathbf{test}}$ is not a subset of $\approx$. If we restrict consideration to a subset of processes rich enough to cover most of the practically interesting processes, open bisimilarity is strictly finer than testing equivalence.

If $P \stackrel{\lambda}{\Rightarrow} P'$, we say that P' is a *descendant* of P. Note that by definition P is a descendant of itself.

DEFINITION 4.1 *A process P is* hereditarily terminating *if P' is terminating whenever it is a descendant of P.*

THEOREM 4.1 *For hereditarily terminating processes, open bisimilarity is strictly included in testing equivalence.*

Proof The proof is derived from two points.

(1) $\approx \subseteq =_{\mathbf{test}}$

The important property here is that weak bisimilarity respects local observability (see Theorem 2.2):

$$P \approx Q \implies C[P] \approx C[Q] \text{ for every ambient context } C[\_].$$

Without loss of generality, suppose $\top \in r(C_o[P])$. In this case P_m exists such that $C_o[P] \Rightarrow P_m \xrightarrow{\omega[\text{open}\,\omega]}$. Thanks to the local observability, we have $C_o[P] \approx C_o[Q]$ and we can find Q_m such that $C_o[Q] \Rightarrow Q_m \xrightarrow{\omega[\text{open}\,\omega]}$. Therefore $\top \in r(C_o[Q])$ holds.

The case of $\bot \in r(C_o[P]) \Longrightarrow \bot \in r(C_o[Q])$ is similar. Because we are dealing with hereditarily terminating processes, the $\bot$ computation of $C_o[P]$ must be an infinite sequence of interactions between $C_o[\_]$ and P if $C_o[\mathbf{0}]$ is also hereditarily terminating. It follows that there must exist an $\bot$ infinite sequence of interactions between $C_o[\_]$ and Q. If $C_o[\mathbf{0}]$ is not hereditarily terminating, then $C_o[Q]$ will surely have a $\bot$ computation otherwise we can tell them apart by a simpler hereditarily terminating observer.

(2) $=_{\mathbf{test}} \not\subseteq \approx$

The following two processes A and B form a counterexample

$$A \equiv (d)(e[\text{open}\,d.(\overline{\text{out}}\,a.\overline{\text{out}}\,b \mid \overline{\text{out}}\,a.\overline{\text{out}}\,c)] \mid d[\overline{\text{open}}\,e])$$

$$B \equiv (d)(e[\text{open}\,d.\overline{\text{out}}\,a.(\overline{\text{out}}\,b \mid \overline{\text{out}}\,a.\overline{\text{out}}\,c)] \mid e[\text{open}\,d.\overline{\text{out}}\,a.(\overline{\text{out}}\,a.\overline{\text{out}}\,b \mid \overline{\text{out}}\,c)] \mid d[\overline{\text{open}}\,e]).$$

We can verify that $A =_{\mathbf{test}} B$, but $A \not\approx B$. ■

Congruence is another important property for an equivalence relation, for it can make the equivalence relation fully algebraic, in which processes can be manipulated like terms in algebra. The testing equivalence is congruent on finite FA processes. A *finite* FA process is one that does not contain the replication operator '!'. Testing equivalence is not closed under the replication operator.

LEMMA 4.2 *If* $P =_{\textbf{test}} Q$, *then* $\forall R\colon P \mid R =_{\textbf{test}} Q \mid R$.

Proof By Theorem 3.5, $P =_{\textbf{test}} Q$ is equal to

$$\forall o, O, O' : r(o[P \mid O] \mid O') = r(o[Q \mid O] \mid O'),$$

which is equal to

$$\forall R, o, O, O' : r(o[P \mid R \mid O] \mid O') = r(o[Q \mid R \mid O] \mid O').$$

Let $C_o[\_] \equiv o[\_ \mid O] \mid O'$. It follows that

$$\forall R\colon \forall C_o[\_] : r(C_o[P \mid R]) = r(C_o[Q \mid R]).$$

Therefore we have $\forall R\colon P \mid R =_{\textbf{test}} Q \mid R$. ■

LEMMA 4.3 *If* $P =_{\textbf{test}} Q$, *then* $\kappa.P =_{\textbf{test}} \kappa.Q$.

Proof For each observer $o[\_ \mid O] \mid O'$, we can verify that if

$$o[\kappa.P \mid O] \mid O' \Longrightarrow C'[\kappa.P]$$

for some context $C'[\_]$ then it must be true that

$$o[\kappa.Q \mid O] \mid O' \Longrightarrow C'[\kappa.Q].$$

If $C'[\kappa.P] \xrightarrow{\tau} C''[P]$ for some $C''[\_]$, then $C'[\kappa.Q] \xrightarrow{\tau} C''[Q]$.

Since $=_{\textbf{test}}$ is closed under substitution, by Theorem 3.5 and the premise $P =_{\textbf{test}} Q$, we know that $r(C''[P]) = r(C''[Q])$ holds and, in consequence,

$$r(o[\kappa.P \mid O] \mid O') = r(o[\kappa.Q \mid O] \mid O').$$ ■

LEMMA 4.4 *If* $P =_{\textbf{test}} Q$, *then* $a[P] =_{\textbf{test}} a[Q]$.

Proof By Theorem 3.5, $P =_{\textbf{test}} Q$ implies $\forall C[\_] : r(C[P]) = r(C[Q])$. For every o, O, O', $o[a[\_] \mid O] \mid O'$ is just a special form of context. It follows that

$$\forall o, O, O' : r(o[a[P] \mid O] \mid O') = r(o[a[Q] \mid O] \mid O'),$$

which is what we need. ■

LEMMA 4.5 *If* $P =_{\textbf{test}} Q$, *then* $(a)P =_{\textbf{test}} (a)Q$.

Proof This is true by restricting observers to those whose set of free names does not contain a, i.e. $a \notin fn(C_o[\_])$. ■

THEOREM 4.6 *Testing equivalence is congruent on finite FA processes.*

Proof Since finite FA processes are those built up by prefix, composition, ambient constructor and restriction. At this point, we have proved that testing equivalence is congruent on finite FA processes. ■

The replication operator breaks down the congruence of the testing equivalence. It is easy to show that $=_{\mathbf{may}}$ is preserved by replication, but $=_{\mathbf{must}}$ is not. For example

$$a[\text{out}\, b] =_{\mathbf{must}} (c)(a[\text{open}\, c] \mid c[\overline{\text{open}}\, a.\text{out}\, b]),$$

but

$$!a[\text{out}\, b] \neq_{\mathbf{must}} !(c)(a[\text{open}\, c] \mid c[\overline{\text{open}}\, a.\text{out}\, b]).$$

This is because for observer $C_o[\_] \equiv o[\_] \mid b[\overline{\text{out}}\, o.\text{in}\, \omega] \mid \omega[\overline{\text{in}}\, b.\text{open}\, \omega]$, we have $!a[\text{out}\, b]$ **must** $C_o[\_]$. But $!(c)(a[\text{open}\, c] \mid c[\overline{\text{open}}\, a.\text{out}\, b])$ **must** $C_o[\_]$ does not hold since $!(c)(a[\text{open}\, c] \mid c[\overline{\text{open}}\, a.\text{out}\, b])$ is divergent.

PROPOSITION 4.7 *Testing equivalence is not closed under the replication operator.*

Open bisimilarity is divergence insensitive while the testing equivalence (especially must-equivalence) is divergence sensitive. When comparing these two equivalence relations, we restrict our consideration to hereditarily terminating processes. We are currently working on a new formulation for testing equivalence from a true interactive viewpoint. It will lead to a more coherent and clear relationship between bisimilarity and testing equivalence while satisfying the complete congruence property.

5. A fully abstract encoding from Pi to FA

We now investigate the robustness of our definition of testing equivalence on the Calculus of FAs by analogy with the one on the π-calculus. We establish a fully abstract result w.r.t. testing equivalence for the encoding from the π-calculus to FA [1]. This result adds weight to our definition of testing equivalence.

The π-calculus considered here is

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

This variant of the π-calculus has all the power to code up the lazy λ-calculus [20], which has been used as a test bed for the descriptive power of a computing model. The definitions of testing equivalence on this fragment of the π-calculus are listed in Appendix B.

The encoding from the π-calculus to FA is defined as follows, with an auxiliary map $[\![\_]\!]_u$ relative to a fresh name u.

$[\![\_]\!]_u$:

$$[\![\mathbf{0}]\!]_u \stackrel{\text{def}}{=} \mathbf{0}$$

$$[\![a(x).P]\!]_u \stackrel{\text{def}}{=} (v)(a[\text{out}\, u.v.a(x).[\![P]\!]_v] \mid v[\text{out}\, u.\bar{a}.!\overline{\text{out}}\, a])$$

$$[\![\bar{a}y.Q]\!]_u \stackrel{\text{def}}{=} (v)(a[\text{out}\, u.v.\bar{a}y.[\![Q]\!]_v] \mid v[\text{out}\, u.\bar{a}.!\overline{\text{out}}\, a])$$

$$[\![P \mid Q]\!]_u \stackrel{\text{def}}{=} [\![P]\!]_u \mid [\![Q]\!]_u$$

$$[\![(x)P]\!]_u \stackrel{\text{def}}{=} (x)[\![P]\!]_u$$

$$[\![!a(x).P]\!]_u \stackrel{\text{def}}{=} (v)(a[\text{out}\,u.v.!a(x).[\![P]\!]_v] \mid v[\text{out}\,u.\bar{a}.!\overline{\text{out}}\,a])$$

$[\![\_]\!]$:

$$[\![\mathbf{0}]\!] \stackrel{\text{def}}{=} \mathbf{0}$$

$$[\![a(x).P]\!] \stackrel{\text{def}}{=} (v)(a[a(x).[\![P]\!]_v] \mid v[!\overline{\text{out}}\,a])$$

$$[\![\bar{a}y.Q]\!] \stackrel{\text{def}}{=} (v)(a[\bar{a}y.[\![Q]\!]_v] \mid v[!\overline{\text{out}}\,a])$$

$$[\![P \mid Q]\!] \stackrel{\text{def}}{=} [\![P]\!] \mid [\![Q]\!]$$

$$[\![(x)P]\!] \stackrel{\text{def}}{=} (x)[\![P]\!]$$

$$[\![!a(x).P]\!] \stackrel{\text{def}}{=} (v)(a[!a(x).[\![P]\!]_v] \mid v[!\overline{\text{out}}\,a])$$

As an example, we use the above encoding to model a communication in FA. The interpretation of $a(x).P \mid \bar{a}y.Q$ simulates the π-communication in the very first step:

$$[\![a(x).P \mid \bar{a}y.Q]\!] \xrightarrow{\tau} (v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{\text{out}}\,a]) \mid (w)(a[[\![Q]\!]_w] \mid w[!\overline{\text{out}}\,a])$$

If $P\{y/x\}$ cannot perform any observable actions, then it is bisimilar to $\mathbf{0}$. In this case $(v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{\text{out}}\,a])$ is bisimilar to $\mathbf{0}$. If $P\{y/x\}$ is of the form $b(z).P'$ then

$$(v)(a[[\![P\{y/x\}]\!]_v] \mid v[!\overline{\text{out}}\,a]) \xrightarrow{\tau}\xrightarrow{\tau}\xrightarrow{\tau} \sim [\![P\{y/x\}]\!].$$

In other words, $[\![P\{y/x\}]\!]_v$ must come out of a before it can simulate $b(z).P'$. This is the general idea of the translation.

Intuitively, we model π-interactions with FA interactions, but after an interaction the result system should not remain in the ambient where the interaction occurs. The name u in the above translation is a handle to pull out the translations from surrounding ambients and make some synchronization between them in order not to interleave two different π-interactions. The pulling out and synchronization are also conducted by exploiting the interaction rules of FA.

Whether this encoding is fully abstract with respect to testing equivalence interests us most. The answer turns out to be positive. It can be stated as the following theorem.

THEOREM 5.1 (Full abstraction) *Suppose P, Q are π-processes, then $P =_{\textit{test}} Q$ if, and only if, $[\![P]\!] =_{\textit{test}} [\![Q]\!]$.*

Roughly speaking, there are two points behind the intuition of this theorem.

(1) $[\![P]\!]$ can do only the communication capabilities in addition to τ, when P is a π-process;
(2) Two testing equivalent processes can do the same sequence of actions in FA, as depicted in Figure 1.

The correspondence between π-actions and FA communication actions is crucial in establishing the full abstraction result. Due to the existence of co-actions within FA, we can control the structural change to some extent. This is what the encoding relies on to simulate π-interactions faithfully.

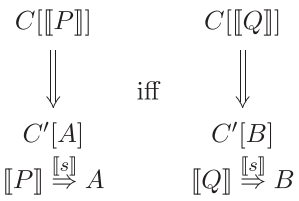

Figure 1. Corresponding actions.

Before the actual proof, we give some useful notations and make some important observations. We use L to denote the set of π-actions, namely $L = \{\tau, ax, \bar{a}x, (x)\bar{a}x \mid a, x \in \mathcal{N}\}$, and $L' = \{\tau, a[ax], a[\bar{a}x], (x)a[\bar{a}x] \mid a, x \in \mathcal{N}\}$ be the set of corresponding communication actions in FA. The complementary action of λ is denoted by $\bar{\lambda}$. The map $[\![\ \_\ ]\!]$ from L to L' is defined as follows:

$$[\![\tau]\!] = \tau$$
$$[\![ax]\!] = a[ax]$$
$$[\![\bar{a}x]\!] = a[\bar{a}x]$$
$$[\![(x)\bar{a}x]\!] = (x)a[\bar{a}x]$$

Note that L' is strictly contained in the set of communication capabilities of FA. We also have $[\![\bar{\lambda}]\!] = \overline{[\![\lambda]\!]}$.

For $s \in L^*$, a sequence of π-actions, we use $\stackrel{s}{\rightarrow}$ to stand for $\stackrel{\lambda_1}{\rightarrow} \cdots \stackrel{\lambda_n}{\rightarrow}$ and $\stackrel{s}{\Rightarrow}$ for $\Rightarrow\stackrel{\lambda_1}{\rightarrow}\Rightarrow \cdots \Rightarrow\stackrel{\lambda_n}{\rightarrow}\Rightarrow$ if $s = \lambda_1 \cdots \lambda_n$. We also write $\stackrel{[\![s]\!]}{\rightarrow}$ for $\stackrel{[\![\lambda_1]\!]}{\rightarrow} \cdots \stackrel{[\![\lambda_n]\!]}{\rightarrow}$, and $\stackrel{[\![s]\!]}{\Rightarrow}$ for $\Rightarrow\stackrel{[\![\lambda_1]\!]}{\rightarrow}\Rightarrow \cdots \Rightarrow\stackrel{[\![\lambda_n]\!]}{\rightarrow}\Rightarrow$ if $[\![s]\!] = [\![\lambda_1]\!] \cdots [\![\lambda_n]\!]$. Here we denote the reflexive and transitive closure of $\rightarrow$ by $\Rightarrow$. We write $\bar{s}$ for $\bar{\lambda}_1 \cdots \bar{\lambda}_n$ if $s = \lambda_1 \cdots \lambda_n$.

Two observations of the operational semantics of the encoding are stated in the following two lemmas [11].

LEMMA 5.2 *If $P \stackrel{\lambda}{\rightarrow} P'$ then $[\![P]\!] \stackrel{[\![\lambda]\!]}{\rightarrow}\Rightarrow\sim [\![P']\!]$, where $\lambda \in L$.*

This lemma says that the encoding $[\![\ \_\ ]\!]$ preserves the operational semantics.

LEMMA 5.3 *If $[\![P]\!] \stackrel{[\![\lambda]\!]}{\Rightarrow} S$ then $P \stackrel{\lambda}{\Rightarrow} P'$ for some P' such that $S \Rightarrow\sim [\![P']\!]$ and $S \approx [\![P']\!]$, where $\lambda \in L$.*

This lemma shows that the encoding $[\![\ \_\ ]\!]$ also reflects the operational semantics.

After stating these two lemmas on operational semantics, we make important observations concerning testing equivalence.

LEMMA 5.4 *For two π-processes P, Q, if $P =_{\textbf{\textit{test}}} Q$ and $P \stackrel{s}{\Rightarrow} P'$, then $Q \stackrel{s}{\Rightarrow} Q'$ for some Q'.*

Proof It is easy to prove this lemma by considering the observers of the form $\bar{s}.\omega$. Here, $\bar{s}.\omega$ means $\bar{\lambda}_1.\bar{\lambda}_2.\cdots.\bar{\lambda}_n.\omega$ if $s = \lambda_1\lambda_2\cdots\lambda_n$. ■

Lemma 5.4 can be strengthened.

LEMMA 5.5 *For two π-processes P, Q, if $P =_{\textbf{\textit{test}}} Q$ and $[\![P]\!] \stackrel{[\![s]\!]}{\Rightarrow}\sim [\![P']\!]$, then $[\![Q]\!] \stackrel{[\![s]\!]}{\Rightarrow}\sim [\![Q']\!]$ for some Q'.*

Proof We get this by induction on the length of s. First we show that if $[\![P]\!] \overset{[\![\lambda]\!]}{\Rightarrow}\sim [\![P']\!]$ then $[\![Q]\!] \overset{[\![\lambda]\!]}{\Rightarrow}\sim [\![Q']\!]$. According to Lemma 5.3, there is P' such that $P \overset{\lambda}{\Rightarrow} P'$, and from Lemma 5.4 we know there is Q' such that $Q \overset{\lambda}{\Rightarrow} Q'$. Then by Lemma 5.2 we get that $[\![Q]\!] \overset{[\![\lambda]\!]}{\Rightarrow}\sim [\![Q']\!]$, using the reasoning in the diagram

$$\begin{array}{ccccc}
[\![Q]\!] & \xrightarrow{\lambda_1} & A_1 & \sim & [\![Q_1]\!] \\
 & & \downarrow{\scriptstyle \lambda_2} & & \downarrow{\scriptstyle \lambda_2} \\
 & & A_2 & \sim & B_2 \quad \sim \quad [\![Q_2]\!] \\
 & & \downarrow{\scriptstyle \lambda_3} & & \downarrow{\scriptstyle \lambda_3} \qquad \downarrow{\scriptstyle \lambda_3} \\
 & & \cdots & \sim & \cdots \quad \sim \quad \cdots
\end{array}$$

The induction step is shown in a similar way. Suppose $[\![P]\!] \overset{[\![\lambda_1]\!][\![s_1]\!]}{\rightarrow\Rightarrow}\sim [\![P']\!]$, then we have

$$\begin{array}{ccccc}
[\![Q]\!] & \xrightarrow{\lambda_1} & A_1 & \sim & [\![Q_1]\!] \\
 & & \Downarrow{\scriptstyle [\![s_1]\!]} & & \Downarrow{\scriptstyle [\![s_1]\!]} \\
 & & A_2 & \sim & B_2 \quad \sim \quad [\![Q']\!] .
\end{array}$$

Combining the two steps above completes the proof of this lemma. ■

Lemma 5.5 renders it possible to describe the role of $[\![P]\!]$ in an ambient computation, which is stated in the following lemma.

LEMMA 5.6 *Every computation of $C[[\![P]\!]]$ can be represented by $C[[\![P]\!]] \Rightarrow C'[A]$ where $C[\_]$ and $C'[\_]$ are two ambient contexts and $[\![P]\!] \overset{[\![s]\!]}{\rightarrow} A$ for some $[\![s]\!] \in L'^*$.*

Proof We notice that the possible actions that $[\![P]\!]$ can do are contained in $\{\tau, a[ax], a[\bar{a}x], (x)a[\bar{a}x] \mid a, x \in \mathcal{N}\}$. There are no ambient capabilities in it, for example $a[\text{in}\, b]$, which can change the structure of context $C[\_]$ directly. Therefore we can write $[\![P]\!] \overset{[\![s]\!]}{\Rightarrow} A$ for $[\![P]\!]$'s contribution in such a computation by concatenating all actions from $[\![P]\!]$ to $[\![s]\!] \in L'^*$. ■

Lemma 5.6 allows us to give the correspondence between π-actions and FA communication actions.

LEMMA 5.7 *Let P and Q be two testing equivalent π-processes. If $C[[\![P]\!]] \Rightarrow C'[A]$, in which $[\![P]\!] \overset{[\![s]\!]}{\Rightarrow} A$, is a computation of $C[[\![P]\!]]$, then $C[[\![Q]\!]] \Rightarrow C'[B]$ is a computation of $C[[\![Q]\!]]$ where $[\![Q]\!] \overset{[\![s]\!]}{\Rightarrow} B$.*

Proof It can be derived from Lemmas 5.5 and 5.6. ■

As mentioned above, we are now ready to prove Theorem 5.1.

Proof of Theorem 5.1 Two directions:

'$\Longrightarrow$': For each observer $C[\_] \equiv o[\_ \mid O] \mid O'$, we are going to show that $r(C[[\![P]\!]]) = r(C[[\![Q]\!]])$. For every computation c of $C[[\![P]\!]]$, by Lemma 5.6, we can write c as $C[[\![P]\!]] \Rightarrow C'[A]$ and $[\![P]\!] \stackrel{[\![s]\!]}{\Rightarrow} A$ for some $[\![s]\!]$. Since $P =_{\mathbf{test}} Q$, by Lemma 5.6 there is a computation of $C[[\![Q]\!]]$ which can be written as $C[[\![Q]\!]] \Rightarrow C'[B]$ where $[\![Q]\!] \stackrel{[\![s]\!]}{\Rightarrow} B$. Due to the fact that $A \sim [\![P']\!]$ and $B \sim [\![Q']\!]$, $\omega[\text{open}\,\omega]$ can only be observed in $C'[\_]$, while not in A or B. We have

$$C'[A] \xrightarrow{\omega[\text{open}\,\omega]} \text{ iff } C'[B] \xrightarrow{\omega[\text{open}\,\omega]} .$$

Therefore $r(C[[\![P]\!]]) = r(C[[\![Q]\!]])$ holds.

'$\Longleftarrow$': For any π-observer o, we select the ambient context

$$C[\_] \equiv o'[\_ \mid [\![o]\!] \mid \omega[\bar{\omega}.\text{out}\,\omega'.\text{open}\,\omega]] \mid \omega'[\overline{\text{out}}\,o']$$

as our FA observer. Obviously, if $P \mid o$ has a successful computation, that is $P \mid o \Rightarrow \stackrel{\omega}{\rightarrow}$, then $[\![P]\!] \mid [\![o]\!] \equiv [\![P \mid o]\!] \Rightarrow \stackrel{\omega[\omega]}{\rightarrow}$ by Lemma 5.2. It follows that

$$C[[\![P]\!]] \Longrightarrow o[A] \mid \omega[\text{open}\,\omega] \mid \omega'[\,] \xrightarrow{\omega[\text{open}\,\omega]}$$

for some A, which in turn implies that $C[[\![P]\!]]$ has a successful computation in FA. Since $[\![P]\!] =_{\mathbf{test}} [\![Q]\!]$, $C[[\![Q]\!]]$ also has a successful computation in FA. According to the structure of $C[\_]$, this is true only if $[\![Q \mid o]\!] \equiv [\![Q]\!] \mid [\![o]\!] \stackrel{\omega[\omega]}{\rightarrow}$. By Lemma 5.3, $Q \mid o \stackrel{\omega}{\rightarrow}$ holds, which implies $Q \mid o$ has a successful computation. Therefore $r(P \mid o) = r(Q \mid o)$ follows. ■

The significance of the full abstraction result is twofold. Firstly, it reinforces the view that the π-calculus is a sub-calculus of FA. Secondly, it assures that the testing equivalence we have introduced for FA is consistent with the testing equivalence for the π-calculus.

6. Conclusion

Interactions—activities by which two or more concurrent processes affect each other—are the link between the testing method and FAs. The testing approach uses interactions, which is the sole way to know the behaviour of processes, to test a process. Also by interactions, FA strengthens ambient calculi in controlling the movement of ambients. As far as we are aware, the testing approach has never been applied to ambient calculi (both may-testing and must-testing). Our preliminary study has demonstrated how testing equivalence can be defined for the ambient calculi.

The basic idea of the testing approach is to use the observing contexts to tell the processes apart. In the framework of mobile processes, however, the observing contexts are not strong enough to give rise to interesting equivalence relations. Closure under substitution must be imposed. This unsatisfactory phenomenon is caused by the confusion of the two different types of names in mobile processes, the constant names and the name variables. We are currently working on a new formulation of process calculi where there is a clear distinction between them. This simplification in addition to the new formulation of testing from a true interactive viewpoint leads to a simple and beautiful testing theory.

There are several other problems that deserve further study. One is to investigate the expressiveness results of ambient calculi and other process calculi using the testing approach, like investigations using subbisimilarity [12]. In addition, the problem of axiomatizing testing equivalence on FA remains untouched, as does the problem of axiomatizing open bisimilarity. These problems, in our opinion, will provide deeper insight into both the testing approach and process calculi.

Acknowledgements

The authors would like to express their gratitude to Prof. Yuxi Fu for his inspiration, guidance and many fruitful discussions on this topic. The authors are indebted to members of Basics Lab for their proofreading of this paper and many useful suggestions on improvements. The authors also wish to thank anonymous referees for their constructive comments. This work is supported by the National 973 Project (2003CB317005) and the National Nature Science Foundation of China (60573002, 60703033).

References

[1] M. Boreale and R. de Nicola, *Testing equivalence for mobile processes*, Inform. Comput. 120 (1995), pp. 279–303.
[2] L. Cardelli, *Abstraction for mobile computation*, Proceedings of Secure Internet Programming 1999, Lecture Notes in Computer Science, vol. 1603, Springer, Berlin, 1999, pp. 51–94.
[3] L. Cardelli and A. Gordon, *Types for mobile ambients*, *Proceedings of POPL'99*, ACM Press, New York, 1999, pp. 79–92.
[4] L. Cardelli and A. Gordon, *Anytime, anywhere: modal logics for mobile ambients*, Proceedings of POPL'00, ACM Press, New York, 2000, pp. 365–377.
[5] L. Cardelli and A. Gordon, *Mobile ambients*, Theoret. Comput. Sci. 240 (2000), pp. 177–213.
[6] L. Cardelli, G. Ghelli, and A. Gordon, *Types for the ambient calculus*, Inform. Comput. 177 (2002), pp. 60–194.
[7] R. de Nicola and M. Hennessy, *Testing equivalences for processes*, Theoret. Comput. Sci. 34 (1984), pp. 83–133.
[8] Y. Deng, R. van Glabbeek, M. Hennessy, C. Morgan, and C. Zhang, *Characterising testing preorders for finite probabilistic processes*, Proceedings of LICS'07, IEEE Computer Society, Silver Spring, MD, 2007, pp. 313–325.
[9] Y. Fu, *Testing congruence for mobile processes*, J. Comput. Sci. Technol. 17 (2002), pp. 73–82.
[10] Y. Fu, *On quasi open bisimulation*, Theoret. Comput. Sci. 338 (2005), pp. 96–126.
[11] Y. Fu, *Fair ambients*, Acta Informatica 43 (2007), pp. 535–594.
[12] Y. Fu and H. Lu, *On the expressiveness of interaction* (2009), Working paper. Available at http://basics.sjtu.edu.cn/nyuxi/papers/expressiveness_of_interaction.pdf.
[13] A. Gordon and L. Cardelli, *Equational properties of mobile ambients*, Math. Struct. Comput. Sci. 13 (2003), pp. 371–408.
[14] X. Guan, Y. Yang, and J. You, *Typing evolving ambients*, Inform. Process. Lett. 80 (2001), pp. 265–270.
[15] T. Kato, *An equivalence relation for the typed ambient calculus*, IPSJ Digital Courier 3 (2007), pp. 369–390.
[16] F. Levi and D. Sangiorgi, *Controlling interference in ambients*, Proceedings of POPL'00, ACM Press, New York, 2000, pp. 352–364.
[17] M. Merro and M. Hennessy, *Bisimulation congruences in safe ambients*, Proceedings of POPL'02, ACM Press, New York, 2002, pp. 71–80.
[18] M. Merro and M. Hennessy, *A bisimulation-based semantic theory of safe ambients*, ACM Trans. Programm. Lang. Syst. 28 (2006), pp. 290–330.
[19] M. Merro and F. Zappa Nardelli, *Bisimulation proof methods for mobile ambients*, Proceedings of ICALP'03, Lecture Notes in Computer Science, vol. 2179, Springer, Berlin, 2003, pp. 584–598.
[20] R. Milner, *Functions as processes*, Tech. Rep. 1154, INRIA, Sophia Antipolis, France, 1990.
[21] R. Milner, J. Parrow, and D. Walker, *A calculus of mobile processes (part i and part ii)*, Inform. Comput. 100 (1992), pp. 1–77.
[22] J. Rathke and P. Sobocinski, *Deriving structural labelled transitions for mobile ambients*, Proceedings of CONCUR'08, Lecture Notes in Computer Science, vol. 5201, Springer, Berlin, 2008, pp. 462–476.
[23] D. Sangiorgi and D. Walker, *On barbed equivalence in π-calculus*, Proceedings of CONCUR'01, Lecture Notes in Computer Science, vol. 2154, Springer, Berlin, 2001, pp. 292–304.
[24] D. Teller, P. Zimmer, and D. Hirschkoff, *Using ambients to control resources*, Proceedings of CONCUR'02, Lecture Notes in Computer Science, vol. 2421, Springer, Berlin, 2002, pp. 288–303.
[25] M. Vigliotti and I. Phillips, *Barbs and congruences for safe mobile ambients*, Electronic Notes in Theoretical Computer Science, vol. 66, 2002, pp. 37–51.

Appendix A. Examples of fair ambients processes

Example A.1 (Renaming)

$$b[\text{open}\, a] \mid a[\overline{\text{open}}\, b \mid P] \xrightarrow{\tau} b[P]$$

Example A.2 (Nondeterministic Choice) Let $d[\overline{\text{open}}\, a] + d[\overline{\text{open}}\, b]$ be

$$(c)(c[\overline{\text{open}}\, d.\overline{\text{open}}\, a] \mid c[\overline{\text{open}}\, d.\overline{\text{open}}\, b] \mid d[\text{open}\, c])$$

then clearly

$$d[\overline{\text{open}}\, a] + d[\overline{\text{open}}\, b] \xrightarrow{\tau} \sim d[\overline{\text{open}}\, a]$$

$$d[\overline{\text{open}}\, a] + d[\overline{\text{open}}\, b] \xrightarrow{\tau} \sim d[\overline{\text{open}}\, b]$$

Thus the nondeterministic choice $a[P] + b[Q]$ is then defined by

$$(d)((d[\overline{\text{open}}\, a] + d[\overline{\text{open}}\, b]) \mid a[\text{open}\, d.P] \mid b[\text{open}\, d.Q])$$

Example A.3 This example shows the transportation of resources from one ambient to another ambient can be made more secure by local names.

$$\begin{aligned} & a[N \mid (\nu)\bar{b}\nu.\nu[\text{out}\, b.\overline{\text{open}}\, b.P]] \mid b[a(x).\overline{\text{out}}\, a.\text{open}\, x.Q] \\ & \xrightarrow{\tau} (\nu)(a[N \mid \nu[\text{out}\, b.\overline{\text{open}}\, b.P]] \mid b[\overline{\text{out}}\, a.\text{open}\, \nu.Q]) \\ & \xrightarrow{\tau} (\nu)(\nu[\overline{\text{open}}\, b.P] \mid a[N] \mid b[\text{open}\, \nu.Q]) \\ & \xrightarrow{\tau} (\nu)(a[N] \mid b[P \mid Q]) \end{aligned}$$

The communication of a local name makes sure that an ambient will not choose a wrong target in the middle of the movement from one ambient to another.

Appendix B. Testing equivalence on the fragment of Pi

Testing equivalence for the π-calculus has been given in [1]. The definition we adopt in this paper is from [9].

The standard way of indicating that a process is in some successful state is for the process to perform a special action. In de Nicola and Hennessy's approach, the special action ω is used to indicate the success of tests. We now give the definitions of the testing equivalence under this assumption.

DEFINITION B.1 *Testing equivalence features observers, tests and results.*

- *Observers are ordinary processes supposed to perform tests:*

$$o, o_0, o_1, \ldots$$

- *Experiments are the parallel composition*

$$P \mid o,\ P_0 \mid o_0,\ P_1 \mid o_1, \ldots$$

  where $P, P_0, P_1, \ldots$ are processes and $o, o_0, o_1, \ldots$ observers.
- *Computations of $P_0 \mid o_0$ are just any maximal sequences of τ-transitions starting from $P_o \mid o_0$:*

$$P_0 \mid o_0 \xrightarrow{\tau} (\tilde{x}_1)(P_1 \mid o_1) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (\tilde{x}_n)(P_n \mid o_n) \xrightarrow{\tau} \not\rightarrow$$

  or

$$P_0 \mid o_0 \xrightarrow{\tau} (\tilde{x}_1)(P_1 \mid o_1) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (\tilde{x}_n)(P_n \mid o_n) \xrightarrow{\tau}$$

  The set of all computations of an experiment $(P \mid o)$ is denoted by $c(P \mid o)$.
- *A state is successful if it can do a special ω action:*

$$(\tilde{x}_m)(P_m \mid o_m) \xrightarrow{\omega}$$

  If a computation contains a successful state, then it is also called successful, otherwise unsuccessful.
- *The result set of an experiment $P \mid o$, denoted by $r(P \mid o)$, is defined as follows:*
  - $\top \in r(P \mid o)$ *if there exists a successful computation in $c(P \mid o)$;*
  - $\bot \in r(P \mid o)$ *if there exists an unsuccessful computation in $c(P \mid o)$.*

  Obviously any result set is a subset of $\{\top, \bot\}$.
- *We say P **may** o if $\top \in r(P \mid o)$ and P **must** o if $\bot \notin r(P \mid o)$.*

Using **may** and **must** relations, we can introduce three ground testing equivalence relations in the obvious way.

DEFINITION B.2 *Three ground testing equivalence relations are defined as follows:*

(1) *Ground may equivalence* $P \approx_{\mathbf{may}} Q$*: For every observer* o*,* P ***may*** o *if, and only if,* Q ***may*** o*.*
(2) *Ground must equivalence* $P \approx_{\mathbf{must}} Q$*: For every observer* o*,* P ***must*** o *if, and only if,* Q ***must*** o*.*
(3) *Ground testing equivalence* $P \approx_{\mathbf{test}} Q$*: Both* $P \approx_{\mathbf{may}} Q$ *and* $P \approx_{\mathbf{must}} Q$*.*

It is well known that these ground testing equivalence relations are not closed under prefix operator. For instance it is obvious that

$$[x = a]a(z) \approx_{\mathbf{test}} [x = b]b(z),$$

but it is not the case that

$$a(x).[x = a]a(z) \approx_{\mathbf{test}} a(x).[x = b]b(z).$$

Therefore closure under substitution must be imposed on them.

DEFINITION B.3 *Three testing equivalence relations are defined as follows:*

(1) $P =_{\mathbf{may}} Q$*, if and only if,* $P\sigma \approx_{\mathbf{may}} Q\sigma$ *for every substitution* σ*.*
(2) $P =_{\mathbf{must}} Q$*, if and only if,* $P\sigma \approx_{\mathbf{must}} Q\sigma$ *for every substitution* σ*.*
(3) $P =_{\mathbf{test}} Q$*, if and only if,* $P\sigma \approx_{\mathbf{test}} Q\sigma$ *for every substitution* σ*.*