Testing Equivalence for Mobile Processes

Michele Boreale and Rocco De Nicola
Dipartimento di Scienze dell'Informazione
Università di Roma "La Sapienza"
Email: michele@dsi.uniroma1.it, denicola@vm.cnuce.cnr.it

Abstract

The impact of applying the testing approach to a calculus of processes with dynamic communication topology is investigated. A proof system is introduced that consists of two groups of laws: those for strong observational equivalence plus those needed to deal with invisible actions. Soundness and completeness of this proof system w.r.t. a testing preorder are shown. A fully abstract denotational model for the language is presented that takes advantage of reductions of processes to normal forms.

1 Introduction

Process Algebras [Mil89, Hoa85, BK89] are generally recognized as a good formalism for describing and studying properties of distributed concurrent systems. A process algebra is often defined by specifying its syntax and transitional semantics by means of Structural Operational Semantics [Plo81]. By now, this approach has become a standard tool for specifying basic semantics of process algebras, but it was early recognized that it does not yield extensional accounts of processes. Thus, techniques have been developed to abstract from unwanted details in systems descriptions. Many of these techniques are based on behavioural equivalences; two processes are identified if and only if no observer can notice any difference between their external behaviours. Most of these equivalence notions are based on *bisimulation* (see e.g. [Mil89]) or on *testing* (see e.g. [Hen88]). Complete axiomatizations have been put forward which are of fundamental importance; they permit manipulating process expressions by means of simple axioms and inference rules and are the theoretical basis for a class of verification tools (see e.g. [DIN90, Hui92]).

'Traditional' process algebras, such as CCS [Mil89], can be used to describe systems whose subparts can interact by performing *pure synchronizations*. Only lately, a language with value-

passing has been investigated and a complete axiomatization of a testing-based equivalence has been provided for it [HI91]. A further step toward improving the descriptive power of process algebras has been performed by adding primitives for expressing exchange of more complex objects, such as channel names or processes themselves, see e.g. ECCS of [EN86] and CHOCS, [Tho89]. Building on ECCS, Milner, Parrow and Walker put forward another extension of CCS, named π-calculus [MPW89]. This permits describing *mobile* systems, i.e. systems that may dynamically change their communication topology by exchanging channel names (also called simply *names*). For these languages only *strong* (i.e. not abstracting from internal moves) bisimulation-based theories have been investigated [MPW89, PS93, San93].

The aim of this paper is to investigate applicability of the testing approach to a name-passing process algebra. Its main contributions are an axiomatization of a *weak* (i.e. abstracting from internal moves) testing-based behavioural relation over the π-calculus and a related fully abstract denotational model. The new semantics is defined by following the general testing approach of [DH84, De 87, Hen88], that requires formally defining a *set of observers, a way of observing* and a *general criterion for interpreting results of observations*. If we call $\mathcal{P}$ the set of systems we want to experiment upon, to apply the testing setting, we need to define a set of observers, say $\mathcal{O}$, and explain the evolution of pairs such as $(P, o) \in \mathcal{P} \times \mathcal{O}$ that will represent the interactions between P and o. Interactions may be failing or successful, depending on whether particular states (which *report success*) are reached. For specific process P and observer o, one might be interested in knowing whether a successful interaction does exists, i.e. whether P **may satisfy** o, or whether all interactions are successful, i.e. whether P **must satisfy** o. Two testing preorders over $\mathcal{P}$, naturally arise, associated with each of the above points of view, the *may* preorder and the *must* preorder:

- $P \sqsubseteq_{may} Q$ if and only if for each $o \in \mathcal{O}$: P **may satisfy** o implies Q **may satisfy** o;
- $P \sqsubseteq_{must} Q$ if and only if for each $o \in \mathcal{O}$: P **must satisfy** o implies Q **must satisfy** o.

These preorders can be combined to get a third one, called the *testing* preorder:

- $P \sqsubseteq Q$ if and only if $P \sqsubseteq_{may} Q$ and $P \sqsubseteq_{must} Q$.

In [DH84], the above machinery is applied to CCS, and the view is taken that *observers should be processes themselves*, which interact with other processes by *communicating* with them. This gives rise to three testing preorders with simple axiomatic and denotational characterizations. Applying this framework to the π-calculus is harder; difficulties are mainly due to the distinction between

input and output actions and to the specific instantiation scheme of input actions. Additional complications are also due to the presence of two distinct kinds of output actions. In particular, a π-calculus process can export public names (*free outputs actions*), as well as private ones (*bound outputs actions*) and thus it has, in general, different options when prompted for an output.

To fully discriminate between different behaviours, observers are needed which can test whether two received names are the same or not. This has called for an extension of the experimenters language; the original π-calculus has hence been enriched with a *mismatch* operator, that together with the original *match* operator gives us the needed observational power. The relevance of mismatch to the π-calculus is currently debated. Recently, it has been used in [PS93] to provide axiomatizations for various bisimulation-based equivalences. Here, we establish the importance of mismatch from an observational point of view. We will see that the new operator is necessary to obtain "reasonable" testing equivalences: its omission from the calculus would affect in a counter-intuitive way the testing power of our observers.

The rest of the paper is organized as follows. In Section 2, syntax and transitional semantics of a finite variant of the π-calculus are introduced, together with a presentation of the testing approach and its application to the π-calculus. Evidence is given of the importance of the mismatch operator and the impact of testing on different variants of the transitional semantics is studied. In Section 3, a sound and complete (w.r.t. a testing preorder) proof system for the finite calculus is introduced. It consists of two groups of laws: one containing (essentially) the laws for the strong observational equivalence of [MPW89], the other dealing with internal actions and nondeterminism. In Section 4, a fully abstract denotational model for the finite π-calculus is presented. The model is based on an algebraic cpo, whose compact elements are represented by (equivalence classes of) process normal forms. The generalization to the full calculus is described in Section 5. There, the π-calculus with recursion is presented and a complete proof system and a fully abstract denotational model are worked out as extensions of those for the finite language. All detailed proofs are reported in Section 6. The concluding section contains a few remarks on related work and future research.

2 Applying Testing Equivalence to the π-calculus

2.1 The π-calculus

In this subsection we give a succinct presentation of the π-calculus, by introducing its syntax and transitional semantics and some standard notation. The reader is referred to [MPW89] for

motivations and additional definitions. We confine ourselves to finite agents, i.e. consider a language without recursion, but, for reasons which will be clear in the sequel, add two extra operators: *mismatch* $[x \neq y]$ and *divergence* Ω. In spite of these changes, we shall continue to call this language π-calculus.

Definition 2.1 (Syntax) *The language of the finite π-calculus is built from the operators of* inaction, divergence, silent prefix, input prefix, output prefix, restriction, match, mismatch, sum *and* parallel composition *according to the following BNF-like grammar:*

$$P ::= \mathbf{0} \mid \Omega \mid \tau.P \mid x(y).P \mid \overline{x}y.P \mid (y)P \mid [x = y]p \mid [x \neq y]P \mid P + P \mid P \mid P$$

where $x, y, \ldots$ range over an infinite set $\mathcal{N}$ of names*, called also* variables*. We call $\mathcal{P}$ the set of terms generated by the syntax above; the elements of $\mathcal{P}$ are called* processes*. Metavariables $P, Q, R, \ldots$ range over $\mathcal{P}$.*

The language is essentially an extension of CCS [Mil80] which permits expressing communication of channel names and thus describing systems with a dynamically changing communication structure. Indeed, some operators are essentially the same as in CCS; this is the case for $\mathbf{0}$, the terminated process; $[\,.\,]+[\,.\,]$, the nondeterministic choice of two processes; and $[\,.\,]|[\,.\,]$, the operator for parallel composition.

Action prefix, $\alpha.[\,.\,]$, is only similar to that of CCS; here a distinction is introduced between input and output actions. Restriction $(y)[\,.\,]$ is similar in spirit to the corresponding CCS operator, but its scope may change dynamically, when a process exports a restricted name. The meaning of the match operator $[x = y][\,.\,]$ is that $[x = y]P$ behaves like P if x is syntactically equal to y and like $\mathbf{0}$ otherwise. The operators we introduce are mismatch $[x \neq y][\,.\,]$, and the divergence constant Ω. The latter is used in the testing approach to represent those agents which perform only infinite sequences of internal actions. The mismatch operator $[x \neq y][\,.\,]$ is complementary to match: $[x \neq y]P$ behaves like P if x is different from y and like $\mathbf{0}$ otherwise. It is worth noticing that match and mismatch, together with sum, permit easily expressing the if-then-else construct; indeed $[x = y]P + [x \neq y]Q$ corresponds to `if` $x = y$ `then` P `else` Q. Finally, we use the *bound output prefix* $\overline{x}(y).P$, $x \neq y$, as a shorthand for $(y)(\overline{x}y.P)$.

Definition 2.2 (Actions) *Define:*

- *the set of* input actions *as* $IA = \{x(y) \mid x, y \in \mathcal{N}\}$*;*

- *the set of* output actions *as* $OA = \{\overline{x}y \mid x, y \in \mathcal{N}\} \cup \{\overline{x}(y) \mid x, y \in \mathcal{N} and x \neq y\}$;
- *the set of* visible actions *as* $\Lambda = OA \cup IA$;
- *the set of all* actions *as* $Act = \Lambda \cup \{\tau\}$. *Symbols* α, β *will be used to range over* Act.

If $\alpha = x(y)$ or $\alpha = \overline{x}y$ or $\alpha = \overline{x}(y)$, we let $subj(\alpha) = x$ and $obj(\alpha) = y$. The π-calculus has two kinds of name *binders*: input prefix $x(y).P$ and restriction $(y)P$ bind the name y in P; consequently, the notions of *free names, fn(.), bound names, bn(.)* and α*-equality,* $\equiv_\alpha$, over both process terms and actions, are the expected ones. We let $n(.) = fn(.) \cup bn(.)$.

Substitutions, ranged over by σ, ρ, are functions from $\mathcal{N}$ to $\mathcal{N}$; for any $x \in \mathcal{N}$, $\sigma(x)$ will be written as $x\sigma$; $P\sigma$ denotes the result of applying the substitution σ to P, i.e. the expression obtained from P by simultaneously replacing each $x \in fn(P)$ with $x\sigma$. As usual, we assume that clashes of names are dealt with by means of renaming of bound names with fresh names. Composition of substitutions is denoted by juxtaposition: given two substitutions σ_1 and σ_2, $\sigma_1\sigma_2$ denotes the substitution such that for all x $x(\sigma_1\sigma_2) = (x\sigma_1)\sigma_2$. A set $\{x_1/y_1, \ldots, x_n/y_n\} = \{\tilde{x}/\tilde{y}\}$, with the x_i's pairwise distinct, will denote the following substitution σ: $x\sigma = y_i$ if $x = x_i$ for some $i \in \{1, \ldots, n\}$, $x\sigma = x$ otherwise.

Definition 2.3 (Transitional Semantics) *The transitional semantics of* $\mathcal{P}$ *is given as a Labelled Transition System, specified in the SOS style by means of the inference rules in Table 1.*

Notations

- We assume the following precedences among operators in agent expressions:
  { restriction, prefix, match, mismatch } > parallel composition > summation.
  Moreover, substitutions will have the strongest precedence, thus e.g. $\alpha.P\sigma$ will stand for $\alpha.(P\sigma)$;
- $P[+R]$ denotes an agent expression where the summand R is optional;
- $[P]$ denotes an agent expression that may have a certain number ($n \geq 0$) of restrictions at its top level, i.e. $[P]$ stands for $(y_1)...(y_n)P$;
- $[y \notin Y][\,.\,]$ will be a shorthand for the context $[y \neq y_1] \cdots [y \neq y_k][\,.\,]$, where Y is a finite set of names, $Y = \{y_1, ..., y_k\}, k \geq 0$; if $Y = \emptyset$, $[y \notin Y]$ will denote the empty context $[\,.\,]$;
- $P \Longrightarrow Q$ stands for $P(\stackrel{\tau}{\longrightarrow})^*Q$; $P \stackrel{\alpha}{\Longrightarrow} Q$, with $\alpha \in \Lambda$, stands for: $P \Longrightarrow \stackrel{\alpha}{\longrightarrow} \Longrightarrow Q$;

$$\texttt{Tau} : \frac{-}{\tau.P \xrightarrow{\tau} P}$$

$$\texttt{Inp} : \frac{-}{x(y).P \xrightarrow{x(z)} P\{z/y\}\ z \notin fn((y)P)} \qquad \texttt{Out} : \frac{-}{\overline{x}y.\,P \xrightarrow{\overline{x}y} P}$$

$$\texttt{Sum} : \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 + P_2 \xrightarrow{\alpha} P_1'} \qquad \texttt{Par} : \frac{P_1 \xrightarrow{\alpha} P_1'}{P_1 \mid P_2 \xrightarrow{\alpha} P_1' \mid P_2\ bn(\alpha) \cap fn(P_2) = \emptyset}$$

$$\texttt{Com} : \frac{P_1 \xrightarrow{\overline{x}z} P_2' \quad P_2 \xrightarrow{x(y)} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} P_1' \mid P_2'\{z/y\}} \qquad \texttt{Close} : \frac{P_1 \xrightarrow{\overline{x}(y)} P_2' \quad P_2 \xrightarrow{x(y)} P_2'}{P_1 \mid P_2 \xrightarrow{\tau} (y)(P_1' \mid P_2')}$$

$$\texttt{Res} : \frac{P \xrightarrow{\alpha} P'}{(y)P \xrightarrow{\alpha} (y)P'\ y \notin n(\alpha)} \qquad \texttt{Open} : \frac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(y)} P'\ x \neq y}$$

$$\texttt{Match} : \frac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'} \qquad \texttt{Mismatch} : \frac{P \xrightarrow{\alpha} P'}{[x \neq y]P \xrightarrow{\alpha} P'\ x \neq y}$$

Table 1: SOS for $\mathcal{P}$ (symmetrical versions of rules `Sum`, `Par`, `Com` and `Close` omitted).

- $P|Q \xrightarrow{c} [P'|Q']$ denotes a communication between P and Q, i.e. a transition $P|Q \xrightarrow{\tau} [P'|Q']$ whose derivation (from the inference rules) contains an application of the `Com` or `Close` rule involving P and Q; we will write $P|Q \stackrel{c}{\Longrightarrow} [P'|Q']$ for $P|Q \Longrightarrow \xrightarrow{c} [P'|Q']$;
- The symbol $\equiv$ denotes syntactic equality between agents.

2.2 Testing the π-calculus

In this subsection, we set up a few definitions that permit applying the testing machinery to the π-calculus. In particular, we will first introduce observers, then experiments and finally testing preorders. We will then use the word *agent* to mean either process or observer or experiment.

Definition 2.4 (Observers)

- *The set $\mathcal{O}$ of* observers *is defined like $\mathcal{P}$, but an additional distinct action $\omega \notin Act$ is used to report success.*
- *We let $Act' = Act \cup \{\omega\}$, $\Lambda' = \Lambda \cup \{\omega\}$, $bn(\omega) = fn(\omega) = \emptyset$, $subj(\omega) = obj(\omega) = -$. The actions $\overline{x}y$ (*free output*), τ and ω are* free actions*, the remaining ones are* bound actions*.*
- *The operational semantics for $\mathcal{P}$ extends to $\mathcal{O}$ by adding the obvious rule: $\omega.o \xrightarrow{\omega} o$.*
- *Unless otherwise stated, $\mathcal{O}$ is ranged over by the metavariables $o, o', ...$*

We recall that square brackets [] around a term indicate that it might be prefixed by some restrictions.

Definition 2.5 (Experiments) *The set $\mathcal{E}$ of* experiments *is defined as:*

$$\{[P|o] |\, P \in \mathcal{P},\, o \in \mathcal{O}\}.$$

Metavariables $e, f, ...$ range over $\mathcal{E}$.

Definition 2.6 (Interactions) *Given an experiment $[P|o]$, an* interaction *(or* computation*) from $[P|o]$ is a maximal sequence of τ transitions, i.e. a sequence of τ-transition:*

$$[P|o] \xrightarrow{\tau} [P_1|o_1] \xrightarrow{\tau} ... \xrightarrow{\tau} [P_k|o_k]$$

with $k \geq 0$ and $[P_k|o_k] \not\xrightarrow{\tau}$.

It is worth noting that, since we are considering the finite sub-language, all computations are finite.

The next definition formalizes the concept of partially divergent agent. Here, the term Ω represents the 'totally divergent' or 'totally undefined' agent, i.e. it models the agent that computes an infinite sequence of internal actions, without ever interacting with the external environment. In spite of the fact that we have not yet considered recursion and infinitary behaviours, we have already extended our language with Ω, because this will allow us to smoothly generalize our theory to the full language in Section 6. Furthermore, Ω will be needed when giving the denotational semantics to the finite fragment we are considering: it will represent the *bottom* of our model.

Definition 2.7 (Definedness and convergence predicates)

- $\downarrow$ *is the least (postfix) predicate over* $\mathcal{O}$ *which satisfies:*
  - $\mathbf{0}\downarrow$, $(\alpha.P)\downarrow$, $([x \neq x]P)\downarrow$, $x \neq y$ *implies* $([x = y]P)\downarrow$;
  - $P\downarrow$ *and* $Q\downarrow$ *imply:* $((y)P)\downarrow$, $([x = y]P)\downarrow$, $([x \neq y]P)\downarrow$, $(P|Q)\downarrow$ *and* $(P + Q)\downarrow$.

  We write $P\uparrow$ *if it is not the case that* $P\downarrow$.
- $\Downarrow$ *is the least (postfix) predicate over* $\mathcal{O}$ *which satisfies:*

  $$P\Downarrow \text{ iff } P\downarrow \text{ and for every } P',\ P \Longrightarrow P' \text{ implies } P'\Downarrow.$$

  We write $P\Uparrow$ *if it is not the case that* $P\Downarrow$.

It is easy to prove that $P\uparrow$ if and only if P contains a *unguarded* occurrence of Ω; i.e. a Ω not inside inside a context of the kind $\alpha.[\,.\,]$, or $[x = y][\,.\,]$ with $x \neq y$, or $[x \neq x][\,.\,]$.

Definition 2.8 (May and Must relations)

- P **may** o *if and only if there exists an interaction* $P|o \xrightarrow{\tau} \ldots \xrightarrow{\tau} [P_k|o_k]$ *such that for some* i, $0 \leq i \leq k$, $o_i \xrightarrow{\omega}$.
- P **must** o *if and only if for each interaction* $P|o \xrightarrow{\tau} \ldots \xrightarrow{\tau} [P_k|o_k]$ *there exists* j, $0 \leq j \leq k$, *such that* $o_j \xrightarrow{\omega}$ *and* $\forall i \leq k$, $((P_i|o_i)\uparrow$ *implies* $i \geq j)$.

Definition 2.9 (Testing preorders)

- $P \sqsubseteq_{may} Q$ *if and only if for every* $o \in \mathcal{O}$ *we have:* P **may** o *implies* Q **may** o.
- $P \sqsubseteq_{must} Q$ *if and only if for every* $o \in \mathcal{O}$ *we have:* P **must** o *implies* Q **must** o.

- $P \sqsubseteq Q$ *if and only if* $P \sqsubseteq_{may} Q$ *and* $P \sqsubseteq_{must} Q$.

The negations of P **may** o and P **must** o will be written as P **m$\not$ay** o and P **m$\not$ust** o, respectively. As it might be expected, while $\sqsubseteq_{may}$ is preserved by the + operator, $\sqsubseteq_{must}$ (hence $\sqsubseteq$) is not. A standard way of tackling this kond of problem is "closing" the $\sqsubseteq_{must}$ relation with respect to summation (see [DH84, Mil89]). An alternative characterization of the new testing preorders can also be given in terms of the original preorders and the initial invisible actions of agents. This leads to the following:

Definition 2.10 (Revisited testing preorders)

- $P \sqsubseteq^{+}_{must} Q$ *if and only if* ($P \sqsubseteq_{must} Q$ *and* ($P \Uparrow$ *or* ($Q \stackrel{\tau}{\longrightarrow}$ *implies* $P \stackrel{\tau}{\longrightarrow}$))));
- $P \sqsubseteq^{+} Q$ *if and only if* ($P \sqsubseteq_{may} Q$ *and* $P \sqsubseteq^{+}_{must} Q$).

It can be easily shown that the above preorders coincide with those generated by the closure under + of the original preorders; i.e.:

- $P \sqsubseteq^{+}_{must} Q$ if and only if for each R, $P + R \sqsubseteq_{must} Q + R$;
- $P \sqsubseteq^{+} Q$ if and only if for each R, $P + R \sqsubseteq^{+} Q + R$.

Thus, the above definition provides us with preorders which are substitutive for the sum operator; however, differently from CCS, we have that none of the relations $\sqsubseteq_{may}$, $\sqsubseteq^{+}_{must}$, $\sqsubseteq^{+}$ is a pre-congruence; indeed, they are preserved by all operators, but input prefix. The failure for the latter operator is proved by the following counter-example.

Let x, y, u, w, z be distinct names in $P \equiv [x = y]\overline{x}w.\mathbf{0}$ and $Q \equiv [x = z]\overline{x}w.\mathbf{0}$. We have $P \sqsubseteq_{may} Q$, because both P and Q are trivially equivalent to $\mathbf{0}$. On the other hand we have that $u(y).P$ is not equivalent $u(y).Q$. Indeed, if we take $o \equiv \overline{u}x.x(w).\omega.\mathbf{0}$ we have:

$$(u(y).P)|o \stackrel{\tau}{\longrightarrow} [x = x]\overline{x}w.\mathbf{0}|x(w).\omega.\mathbf{0} \stackrel{\tau}{\longrightarrow} \mathbf{0}|\omega.\mathbf{0} \stackrel{\omega}{\longrightarrow} \mathbf{0}|\mathbf{0}$$

thus $u(y).P$ **may** o. Conversely, the only possible computation from $(u(y).Q)|o$ is:

$$(u(y).Q)|o \stackrel{\tau}{\longrightarrow} [x = z]\overline{x}w.\mathbf{0}|x(w).\omega.\mathbf{0} \not\stackrel{\tau}{\longrightarrow}$$

and thus $u(y).Q$ **m$\not$ay** o.

Similar counter-examples can be exhibited for $\sqsubseteq^{+}_{must}$ and $\sqsubseteq^{+}$. This kind of problem for the input prefix arises also for *strong ground bisimulation*, the equivalence of [MPW89]. Like in

[MPW89], also for our testing preorders a sort of congruence law for the input prefix operator can be given. Equivalence of two given processes P and Q is preserved by prefix $u(y).[\,.\,]$ provided that it is preserved by all substitutions $\{^w/_y\}$, for any name $w \in fn(P) \cup fn(Q) \cup \{y\}$. Indeed, this will be a rule of our proof system; since we have a finite number of names to test (as $fn(P) \cup fn(Q)$ is finite), this is still effective (although not very 'efficient').

2.3 Testing without mismatch

In this subsection we attempt an assessment of the testing power of observers when the mismatch operator is discarded from the language. We will denote by $\sqsubseteq^-_{may}$, $\sqsubseteq^-_{must}$ and $\sqsubseteq^-$ the testing preorders obtained by permitting only observers without mismatch and by $\simeq^-_{may}$, $\simeq^-_{must}$ and $\simeq^-$ the corresponding equivalences. We will see that each of these relations is different from those of the previous subsection obtained by considering also observer that may contain the mismatch operator. Pairs of processes will be exhibited that can be fully discriminated by using observers with mismatch, but cannot be taken apart without mismatch. The counterexamples will make it evident that the testing relations generated without mismatch are counter-intuitive.

We begin with exhibiting two pairs of processes that show that the two may and the two must preorders are indeed different; this result is then extended to the equivalences. The key proposition is:

Proposition 2.11 *Let x, y and w be any three names. Then:*

a. $\overline{x}(w).\mathbf{0} \sqsubseteq^-_{may} \overline{x}y.\mathbf{0}$.

b. $\overline{x}(w).\Omega \sqsubseteq^-_{must} \overline{x}(w).\mathbf{0} + \overline{x}y.\Omega$.

Proof: See Section 6, Proposition 6.23. □

The above proposition holds essentially because, in absence of mismatch, all transitions that can be performed by an agent upon receipt of a bound (hence 'new') name, can also be performed upon receipt of any name. This is no more possible if mismatch is introduced; this operator permits additional discriminations between free and bound names. Indeed, within the richer language, the observer $o \equiv x(z).[z \neq y]\omega.\mathbf{0}$, with $z \neq y$, permits concluding:

- $\overline{x}(w).\mathbf{0} \not\sqsubseteq_{may} \overline{x}y.\mathbf{0}$;
- $\overline{x}(w).\Omega \not\sqsubseteq_{must} \overline{x}(w).\mathbf{0} + \overline{x}y.\Omega$.

Now, we report below three pairs of processes that are (may, must, testing) equivalent when the restricted language is considered and non-equivalent when observers with the mismatch operator are used. When proving equivalence, we will freely use, both for $\sqsubseteq^-_{may}$ and for $\sqsubseteq^-_{must}$, substitutivity under $+$ - contexts[1], associativity and commutativity. We will also use the following laws, that can be easily proven sound:

(L1) $P \simeq^- P + P$

(L2) $P \sqsubseteq^-_{may} P + Q$

(L3) $\alpha.P + \beta.Q \sqsubseteq^-_{must} \alpha.P$ if $\alpha, \beta \in OA$ and $subj(\alpha) = subj(\beta)$.

- May: $\overline{x}y.\mathbf{0} \simeq^-_{may} \overline{x}y.\mathbf{0} + \overline{x}(w).\mathbf{0}$. Indeed:

$$\begin{array}{lll} 1.\ \overline{x}y.\mathbf{0} & \sqsubseteq^-_{may}\ \overline{x}y.\mathbf{0} + \overline{x}(w).\mathbf{0} & \text{(L2);} \\ 2.\ \overline{x}y.\mathbf{0} + \overline{x}(w).\mathbf{0} & \sqsubseteq^-_{may}\ \overline{x}y.\mathbf{0} + \overline{x}y.\mathbf{0} & \text{(Prop. 2.1.a)} \\ & \simeq^-_{may}\ \overline{x}y.\mathbf{0} & \text{(L1)} \end{array}$$

  On the other hand it is easily seen that the observer o defined above suffices to take the two processes apart within the richer language.

- Must: $P \simeq^-_{must} P + \overline{x}y.\Omega$, where $P \equiv \overline{x}(w).\Omega + \overline{x}y.\mathbf{0}$. Indeed:

$$\begin{array}{lll} 1.\ P + \overline{x}y.\Omega & \sqsubseteq^-_{must}\ P & \text{(L3)} \\ 2.\ P & \simeq^-_{must}\ P + \overline{x}(w).\Omega & \text{(L1)} \\ & \sqsubseteq^-_{must}\ P + \overline{x}(w).\mathbf{0} + \overline{x}y.\Omega & \text{(Prop. 2.1.b)} \\ & \sqsubseteq^-_{must}\ P + \overline{x}y.\Omega & \text{(L3)} \end{array}$$

  It is easily seen that the observer $o' \equiv x(z).([z = y]\tau.\omega.\mathbf{0} + [z \neq y]\omega.\mathbf{0})$ suffices to take the two processes apart within the richer language.

- Testing: $P \simeq^- P + \overline{x}y.\Omega$.

  This follows from the fact that the two processes are both $\simeq^-_{may}$ and $\simeq^-_{must}$ equivalent.

These examples show that the omission of mismatch leads to identifications that do not correspond to any operational intuition. The $\simeq^-_{may}$ equivalence relates a process that may perform a bound output to one that may not. The $\simeq^-_{must}$ equivalence relates a process that after performing a free output $\overline{x}y$ always terminates to one that after the same action may diverge.

[1]All processes under consideration cannot perform invisible initial actions; and it is easy to see that when this condition holds, testing preorders are preserved by $+$-contexts.

2.4 Testing and Early and Late Instantiations

In this subsection the testing approach is compared with the bisimulation approach of [MPW89], especially for the differences induced by different, *early* and *late*, transitional semantics. The transitional semantics we have adopted is the late one proposed in [MPW89] that requires instantiating the formal parameter y of an input prefix $x(y)$. only when a communication is actually inferred (see `Inp` and `Com` rules). The original transitional semantics of the value passing version of CCS [Mil89] is instead based on an *early* instantiation schema, that requires instantiating the parameter when inferring the input action.

In [MPW89], two versions of strong bisimulation for the π-calculus are introduced, on the top of the late transitional semantics. They are called late and early strong ground bisimulations. Their definitions differ in the input clause only. *Late bisimulation* is the largest symmetric relation $\sim$ over processes satisfying: $P \sim Q$ and $P \stackrel{\alpha}{\longrightarrow} P'$ with $bn(\alpha) \cap n(Q) = \emptyset$ implies:

- if α is not an input action then $\exists Q'$: $Q \stackrel{\alpha}{\longrightarrow} Q'$ and $P' \sim Q'$;
- if $\alpha = x(y)$ then $\exists Q'$: $Q \stackrel{x(y)}{\longrightarrow} Q'$ and $\forall z : P'\{z/y\} \sim Q'\{z/y\}$.

The definition of *early bisimulation* is obtained from that above by replacing the second clause with the following more liberal one:

- if $\alpha = x(y)$ then $\forall z : \exists Q'$: $Q \stackrel{x(y)}{\longrightarrow} Q'$ and $P'\{z/y\} \sim Q'\{z/y\}$.

In [San93], another kind of bisimulation, called *open*, is defined and studied; it is based on a lazy name-instantiation strategy where input formal parameters are instantiated as late as possible (only when, and if, required). Open bisimulation is stronger than late bisimulation, that is in turn stronger than the early one. Over finite, Ω-free processes, the latter is stronger than all of our testing equivalences.

We would like to argue that this proliferation of equivalences, in the early, late and open forms, is specific of bisimulation; it arises from the interplay between quantification over names and processes in the input clauses of the definition. If only the original transitional semantics of [MPW89] is considered, within the testing approach it does not make any sense to talk about late or early equivalences.

The above claim holds even if one considers an early transitional semantics, that naturally leads to early bisimulation. Actually, such a transitional semantics for the π-calculus has been introduced in [MPW91]; there, besides the usual bound input transitions $P \stackrel{x(y)}{\longrightarrow} P'$, *free input* transitions are

introduced. These transitions are of the form $P \xrightarrow{xw} P'$, that essentially states that $P \xrightarrow{x(y)} P''$ for some y and P'' such that $P' \equiv_\alpha P''\{w/y\}$. The early transitional semantics is denoted by $\xrightarrow{\alpha}_e$ to distinguish it from the late one. It is very easy to see that the testing preorders over processes obtained from the early transition system ($\xrightarrow{\alpha}_e$) and those obtained from the late one ($\xrightarrow{\alpha}$) do coincide. [2]. The key for the proof is the fact that the τ-actions of the late and the early transition systems are the same, up to α-equivalence.

Lemma 2.12 *$P \xrightarrow{\tau}_e P'$ if and only if there exists some P'' such that $P \xrightarrow{\tau} P''$, with $P'' \equiv_\alpha P'$.*

Proof: See [MPW91]. □

Corollary 2.13 *The testing preorders over $\mathcal{P}$ defined on the top of $\xrightarrow{\alpha}_e$ coincide with the corresponding ones defined on the top of $\xrightarrow{\alpha}$.*

Proof: The previous lemma easily generalizes to sequences of τ-transitions (formally, this can be done by induction on the length of the sequences, by using Lemma 6.1 of Section 6). From this fact and from the definition of testing preorders, we easily obtain the thesis. □

3 A Proof System for Finite π-calculus

In this section, we provide an inequational theory for one of the testing preorders, namely $\sqsubseteq^+_\approx$. The extensions to $\sqsubseteq^+_{\approx must}$ and $\sqsubseteq_{\approx may}$ are only a matter of rephrasing the results. The axioms and the inference rules of the proof system, which we call $\mathcal{F}$, are shown in Table 2 and 3, respectively. We freely use "$P = Q$" as an abbreviation for "$P \sqsubseteq Q$ *and* $Q \sqsubseteq P$".

Most of the axioms and rules in Table 2 are taken from [MPW89] and [DH84]. In particular, the Restriction, Match and Expansion laws and the inference rules are from [MPW89]; the Omega law and the τ-laws N1, N2 and N4 are from [DH84]; the τ- law N3 is a modification of the corresponding law in [DH84] to take into account the fact that a process may have different options when prompted for an output. The Mismatch laws are new. In the Expansion law, an empty summation by convention denotes $\mathbf{0}$. By $\mathcal{F} \vdash P \sqsubseteq Q$ we mean that the relation $P \sqsubseteq Q$ is provable by applying the axioms and the inference rules of system $\mathcal{F}$. Sometimes, we abbreviate this simply as $P \sqsubseteq Q$.

[2] A similar result is proven in [Ing93] for a value-passing process algebra

Sum

$S0$ $\quad P + \mathbf{0} = P$

$S1$ $\quad P + P = P$

$S2$ $\quad P + Q = Q + P$

$S3$ $\quad P + (Q + R) = (P + Q) + R$

Restriction

$R0$ $\quad (x)P = P,\ x \notin fn(P)$

$R1$ $\quad (x)(y)P = (y)(x)P$

$R2$ $\quad (x)(P + Q) = (x)P + (x)Q$

$R3$ $\quad (x)\alpha.P = \alpha.(x)P,\ x \notin n(\alpha)$

$R4$ $\quad (x)\alpha.P = \mathbf{0},\ subj(\alpha) = x$

Match

$M0$ $\quad [x = y]P = \mathbf{0},\ x \neq y$

$M1$ $\quad [x = x]P = P$

Mismatch

$U0$ $\quad [x \neq x]P = \mathbf{0}$

$U1$ $\quad [x \neq y]P = P,\ x \neq y$

Expansion

Suppose P is $\sum_{i \in I} \alpha_i.P_i[+\Omega]$ and Q is $\sum_{j \in J} \beta_j.Q_i[+\Omega]$.

Suppose that no α_i (resp. β_j) binds a name free in Q (resp. P).

$$P \mid Q = \sum_{i \in I} \alpha_i.(P_i \mid Q) + \sum_{j \in J} \beta_j.(P \mid Q_j) + \sum_{\alpha_i \ \mathbf{comp} \ \beta_j} \tau.R_{ij}$$

$[+\Omega|\ \Omega$ is a summand either of P or of $Q]$

where α_i **comp** β_j and R_{ij} are defined as follows:

1. $\alpha_i = \overline{x_i}z$ and $\beta_j = y_j(y)$; then $R_{ij} = P_i \mid Q_j\{z/y\}$
2. $\alpha_i = \overline{x_i}(y)$ and $\beta_j = y_j(y)$; then $R_{ij} = (y)(P_i \mid Q_j)$
3. The "converse" of 1.
4. The "converse" of 2.

Omega

Ω $\quad \Omega \sqsubseteq P$

$\tau - laws$

$N1$ $\quad \alpha.P + \alpha.Q = \alpha.(\tau.P + \tau.Q)$

$N2$ $\quad P + \tau.Q \sqsubseteq \tau.(P + Q)$

$N3$ $\quad \alpha.P + \tau.(\beta.Q + R) = \tau.(\alpha.P + \beta.Q + R) \quad subj(\alpha) = subj(\beta)$ and $(\alpha, \beta \in IN$ or $\alpha, \beta \in OUT)$

$N4$ $\quad \tau.P \sqsubseteq P$

Table 2: Axioms of the proof System $\mathcal{F}$.

$$\alpha - conv \quad \frac{P \equiv_\alpha Q}{P = Q}$$

$$C0 \quad \frac{P \sqsubseteq Q}{P' \sqsubseteq Q'}$$

with $P' \sqsubseteq Q'$ either of: $\overline{\alpha}..P \sqsubseteq \alpha.Q\,(\alpha \notin IN), (x)P \sqsubseteq (x)Q,$

$P + R \sqsubseteq Q + R, [x = y]P \sqsubseteq [x = y]Q, [x \neq y]P \sqsubseteq [x \neq y]Q, P|R \sqsubseteq Q$

$$C1 \quad \frac{\forall z \in fn(P, Q, y).\ P\{z/y\} \sqsubseteq Q\{z/y\}}{x(y).P \sqsubseteq x(y).Q}$$

Table 3: Inference rules of the proof system $\mathcal{F}$; the usual rules for preorder reflexivity and transitivity have been omitted.

The next two propositions state soundness of the inference rules and of the axioms; the actual proofs (see Section 6) can be done along the lines of the corresponding ones for CCS. However, specific care is needed when interactions between processes and observers are analyzed, due to the exchange of names between them. Thus the proofs of the precongruence rules for parallel composition and restriction of C0 are significantly harder than those for CCS; rule C1 was not present in the proof system for CCS.

Proposition 3.1 (Soundness of the inference rules) *The inference rules of Table 3 are sound w.r.t.* $\sqsubseteq^+_\approx$.

Proposition 3.2 (Soundness of the axioms) *The axioms of Table 2 are sound w.r.t.* $\sqsubseteq^+_\approx$.

The above propositions give us the following:

Theorem 3.3 (Soundness of $\mathcal{F}$) $\mathcal{F} \vdash P \sqsubseteq Q$ *implies* $P \sqsubseteq^+_\approx Q$.

The completeness proof is heavily based on the existence of certain standard forms for processes called *head normal forms (hnf)*. Similar forms were already used in [DH84] (though there the completeness proof was based on normal forms). Head normal forms aim at presenting processes as an internal non-deterministic choice among a set, possibly a singleton, of *initial states* (each represented, in Definition 3.5, by a $P_A, A \in L$). Each initial states is essentially described by a set (A) of input or output *channels* through which the process can communicate with the environment,

and by the corresponding *initial actions* (α's) and their *α-derivatives* (P_α 's). If there are several initial states (this is the case of τ-hnf's), the non-deterministic choice among them is made by performing an internal (τ) action.

In the actual definition of hnf, a functional notation is used to make it explicit that each input or output channel a is associated with a unique process $g(a)$. In case a is an input channel, say x, process $g(a)$ has a unique initial input action with subject x, i.e. it is of the form $x(y).P$. In case a is an output channel, say $\overline{x}$, $g(a)$ is a *non-empty* summation of the form $\sum_{z\in F} \overline{x}z.P_z[+\overline{x}(w).P']$, where square brackets enclose an optional summand. Head normal forms will enable us to provide a syntactical characterization of the preorders that will be crucial when proving completeness (see Section 6.4). This syntactical relation relies on a notion of *set saturation*; differently from [DH84], this notion applies to set of input or output channels, not to set of actions.

Definition 3.4 (Saturation) *Let $IN = \mathcal{N}$, $OUT = \{\overline{x} | \, x \in \mathcal{N}\}$, $Ev = IN \cup OUT$; Ev is the set of all* events *and is ranged over by the metavariables a, b.*

- *Given a set of sets of events $L = \{A_1, ..., A_n\}$, define $Ev(L) = A_1 \cup \ldots \cup A_n$;*
- *A non-empty finite set of finite sets of events L is* saturated *if:*

$$\textit{for each } A \subseteq Ev\textit{: } (\, \exists\, B \in L \,:\, B \subseteq A \subseteq Ev(L) \,)\textit{ implies } A \in L.$$

Definition 3.5 (Head normal forms)

- *A partial function $g : Ev \longrightarrow \mathcal{P}$ is a* normal function *if for each $a \in domain(g)$ we have:*

  a. *$a \in IN$ implies $g(a)$ is of the form $a(y).P$, and*

  b. *$a \in OUT$ implies $g(a)$ is a non-empty summation of the form $\sum_{z\in F} az.P_z[+a(w).P']$, with $F \subseteq_{fin} \mathcal{N}$.*

- *Let $A \subseteq_{fin} EV$ and let L be a saturated set of subsets of EV. An agent P is in* head normal form (hnf) *if one of the following conditions holds:*

  i. *(α-hnf) $P \equiv \sum_{a\in A} g(a)$, where $g : A \longrightarrow \mathcal{P}$ is a normal function;*

  ii. *(τ-hnf) $P \equiv \sum_{A\in L} \tau.P_A$, where $P_A \equiv \sum_{a\in A} g(a)$ and $g : Ev(L) \longrightarrow \mathcal{P}$ is a normal function.*

  In both cases, function g is referred to as the associated function of P.

In the sequel, given a hnf P with associated function g, we will let $Ev(P)$ denote the domain of g and $Init(P)$ denote the set of initial visible actions of P, i.e. the set $\{\alpha \in \Lambda |\, P \stackrel{\alpha}{\Longrightarrow}\}$, *up to* α-equivalence. It is evident that if P is a hnf then $P \Downarrow$. From the above definition, it should be also evident that for each $\alpha \in Init(P)$, there exists a unique P_α such that $P \Longrightarrow \stackrel{\alpha}{\longrightarrow} P_\alpha$; this P_α is called α-*derivative* of P. The following property of τ-head normal forms follows directly from the functional notation used in the definition:

Property 3.6 *Let $P \equiv \sum_{A \in L} \tau.P_A$ be a τ-head normal form and let $A \in L$. If $Init(P_A)$ contains an output action with subject x, then $Init(P_A)$ contains also* all *output actions of $Init(P)$ with subject x.*

A few examples will clarify these points. Consider $P \equiv x(y).\mathbf{0} + x(z).\alpha.\mathbf{0}$, with $y \notin fn(\alpha)$; P is not a hnf because we cannot associate to the input channel x a unique term $g(x)$; in particular, P has not a unique $x(y)$-derivative, as $P \stackrel{x(y)}{\longrightarrow} \alpha\{y/z\}.\mathbf{0}$ and $P \stackrel{x(y)}{\longrightarrow} \mathbf{0}$. Next, consider $Q \equiv \tau.\overline{x}y.\mathbf{0} + \tau.\overline{x}z.\mathbf{0}$, with $y \neq z$; Q is not a τ-hnf, since Property 3.1 above does not hold for the output channel $\overline{x}$; indeed, it is not possible to define an associated normal function for Q. As an example of τ-hnf, consider

$$\tau.w(x).\mathbf{0} + \tau.(\overline{x}y.\mathbf{0} + \overline{x}z.\mathbf{0}) + \tau.(w(x) + \overline{x}y.\mathbf{0} + \overline{x}z.\mathbf{0})$$

with x, w, y, z distinct; the associated function g is given by: $g(w) = w(x).\mathbf{0}$ and $g(\overline{x}) = \overline{x}y.\mathbf{0} + \overline{x}z.\mathbf{0}$; notice that the set $L = \{\, \{w\}, \{\overline{x}\}, \{w, \overline{x}\}\}$ is saturated.

The counterpart of head normal form for divergent agents (i.e. those P's such that $P \Uparrow$ holds) is the notion of Ω-*Head Normal Form (Ω-hnf).*

Definition 3.7 (Ω-Head normal forms) *An agent P is in* Ω-Head Normal Form (Ω-hnf) *if $P \equiv (\sum_{\alpha \in A} \alpha.(P_\alpha + \Omega)) + \Omega$, for some $A \subseteq_{fin} \Lambda$, A not containing distinct α-equivalent actions.*

Notice the double occurrence of Ω in the above definition; it, somehow, indicates that once the process diverges at the top-level, it does so at each inner level. If P is an Ω-hnf we let $Init(P)$ and $Ev(P)$ have the same meaning as for hnf.

We establish now a few facts about head normal forms. Their proofs (see Section 6) follow the same lines as the corresponding ones for CCS in [DH84]; additional complications are again introduced by the need of applying rule (C1) and, in the proof of Proposition 3.8, by the need of reducicing P to a form satisfying Property 3.1.

Proposition 3.8 *If $P \Downarrow$ then there exists a head normal form $h(P)$ such that $\mathcal{F} \vdash P = h(P)$.*

Proposition 3.9 *If $P \Uparrow$, then there exists a Ω-head normal form $\Omega(P)$ such that $\mathcal{F} \vdash P = \Omega(P)$.*

For the proof of completeness, we need a special induction parameter, namely the largest number of communications that a process can perform with another agent. This parameter can be defined as the length, in terms of number of visible actions, of the maximal among the observers which are satisfied by the process. We remind the reader that here we have confined ourselves to finite processes, thus this number is finite.

Definition 3.10 (Depth of communication) *We define the depth of communication* dpc *as:*

$$dpc(P) = max\{k |\, P \textbf{ may } \alpha_1 \cdots \alpha_k.\omega.\mathbf{0},\ \textit{with the } \alpha_i,\ 0 \leq i \leq k,\ \textit{ranging over } \Lambda\}.$$

Theorem 3.11 (Completeness of $\mathcal{F}$) *$P \sqsubseteq^+_\approx Q$ implies $\mathcal{F} \vdash P \sqsubseteq Q$.*

Proof: (Sketch; See Section 6 for a detailed proof) The proof goes by induction on $dpc(P)$. It relies on the fact that, in virtue of the above propositions, we can suppose that both P and Q are either in hnf or in Ω-hnf and on the two following properties of α-derivatives of P and Q:

- if $\alpha \in OA$ then $P_\alpha \sqsubseteq^+_\approx Q_\alpha$;
- if $\alpha = x(y) \in IA$ then $P_\alpha\sigma \sqsubseteq^+_\approx Q_\alpha\sigma$ for any name z and any substitution $\sigma = \{z/y\}$.

By applying the induction hypothesis, the above relations still hold when substituting $\sqsubseteq^+_\approx$ by $\sqsubseteq$. Then, rules C0 and C1 and the τ-laws suffice to derive $P \sqsubseteq Q$. □

4 Denotational Semantics

In this section, we sketch a denotational semantics for the π-calculus and discuss its relationships with the operational semantics of Section 2. We assume familiarity with the standard algebraic semantics techniques as described, for example, in [Hen88] or [Gue81]. To avoid complications which would not provide any additional insight, we confine ourselves in searching for a model which is fully abstract w.r.t. the must preorder $\sqsubseteq^+_{\approx_{must}}$ (rather than $\sqsubseteq^+_\approx$). A complete proof system can be obtained for this preorder from the one presented in Section 3 by adding the following three laws:

- $\Omega + P = \Omega$
- $\tau.P + \tau.Q \sqsubseteq P$
- $\alpha.P + \beta.Q \sqsubseteq \alpha.P$ if $subj(\alpha) = subj(\beta)$ and $(\alpha, \beta \in IA$ or $\alpha, \beta \in OA)$.

We will indicate with $\sqsubseteq_{must}$ the corresponding proof-theoretic preorder. The proof of completeness for this preorder is very similar to that of $\sqsubseteq$ presented in Section 6 and will be omitted. In fact, due to the law $\Omega + P = \Omega$, that considers all divergent processes equivalent to Ω, the actual proof is simpler than the one discussed in full detail.

We shall give the denotational model under the form of a *natural interpretation*, a slight variant of the usual algebraic semantics, introduced in [HP80] to deal with languages with explicit value passing. A natural interpretation (for the finite π-calculus) is a 4-tuple $(\mathcal{D}, F_{op}, inp, out)$, where:

- $\mathcal{D}$ is a cpo;
- F_{op} is a family of continuous functions of suitable arities, one for each operator different from input and output prefixes;
- $inp : (\mathcal{N} \times (\mathcal{N} \longrightarrow \mathcal{D})) \longrightarrow \mathcal{D}$ is a function continuous in its second argument, where $(\mathcal{N} \longrightarrow \mathcal{D})$ inherits its ordering pointwise from $\mathcal{D}$;
- $out : (\mathcal{N} \times \mathcal{N} \times \mathcal{D}) \longrightarrow \mathcal{D}$ is a function continuous in its third argument.

Given a natural interpretation, we can define a semantic function: $[\![\,.\,]\!] : \mathcal{P} \longrightarrow \mathcal{D}$ as follows:

- $[\![op(P_1, ..., P_k)]\!] = f_{op}([\![P_1]\!], ..., [\![P_k]\!])$
  for each k-ary operator different from input and output, with $k = 0, 1$ or 2
- $[\![x(y).P]\!] = inp(x, \lambda z \in \mathcal{N}.[\![P\{z/y\}]\!])$
- $\overline{x}y.P = out(x, y, [\![P]\!])$.

Since the considered calculus does not permit describing infinite behaviours, one might ask whether cpo's and continuous functions are really needed to model it. To obtain a fully abstract model, the partial ordering of the domain we use must properly reflect the testing preorder $\sqsubseteq^{+}_{\approx_{must}}$. Now, although we are only dealing with finite agents, we still have elements which are approximable by infinite chains. Consider the process $P = x(y).\mathbf{0}$; P is (syntactically) finite but can be approximated by the infinite chain of processes

$$K = \{x(y).P_i \,|\, P_i \equiv \sum_{z \in F_i} [y = z]\mathbf{0} + [y \notin F_i]\Omega,\, i \geq 0\}$$

where $\{F_i \,|\, i \geq 0\}$ is a strictly increasing chain of finite set of names such that for each j, $y \notin F_j$. Indeed, we have that $x(y).P_0 \sqsubseteq^{+}_{\approx_{must}} x(y).P_1 \sqsubseteq^{+}_{\approx_{must}} ... \sqsubseteq^{+}_{\approx_{must}} x(y).\mathbf{0}$; these inequalities are clearly

strict. We have here a kind of *continuous breadth* approximation; thus, we can expect every fully abstract model be based on a cpo rather than a poset. In fact, the construction illustrated below naturally leads us to a cpo. It is worth noting that exactly the same cpo is suitable for modelling the full calculus with recursion (see Section 5).

The non-trivial task to face when constructing the model is defining function $inp(.,.)$. We will take advantage of process *strong normal forms* (formally described below), that recursively reproduce the top structure of head normal forms. We will first define a poset, PO, of equivalence classes of "finite breadth" strong normal; this poset will represent the compact element of our cpo. Over PO, monotonic functions can be easily defined in a syntax-driven way; moreover, the elements of PO can be used to 'approximate' all processes. Thus, to get a natural interpretation, it will be sufficient first to define monotonic functions f_{op}, out and $inp(x,-)$ over PO corresponding to the opeators of the calculus and then to take the ideal completion of PO the unique continous extensions of functions f_{op}, out and inp over it. It is not hard to show that the natural interpretation so obtained is fully abstract. Below, we sketch the actual construction of the model; it takes four steps.

1. *Strong Normal Forms for all processes are introduced.*

Strong normal functions and *Strong normal forms (snf)* are defined by mutual recursion as follows:

- A partial function $g : Ev \longrightarrow \mathcal{P}$, is a strong normal function if for each $a \in domain(g)$ we have:

  a. if $a = x \in IN$, $g(x)$ is of the form: $x(y).(\sum_{z\in F}[y = z]N_z + [y \notin F]N)$, where:
  1. $F \subseteq_{fin} \mathcal{N}$, $y \notin F$;
  2. for each $z \in F, y \notin fn(N_z)$;
  3. N and all N_z, $z \in F$, are snf.

  b. if $a = \overline{x} \in OUT$, $g(\overline{x})$ is a non-empty summation $\sum_{z\in F} \overline{x}z.N_z[+\overline{x}(w).N]$ where N and each N_z are snf.

- Let $A \subseteq_{fin} N$ and L be a saturated set of subsets of Ev. A process P is in strong normal form if one of the following conditions holds:

  i. P is Ω;

  ii. (α-snf) P is of the form $\sum_{\alpha\in A} g(a)$, where $g : A \longrightarrow \mathcal{P}$ is a strong normal function;

  iii. (τ-snf) P is of the form $\sum_{A\in L} \tau.\sum_{\alpha\in A} g(a)$, where $g : Ev(L) \longrightarrow \mathcal{P}$ is a strong normal function.

We call SNF the set of all strong normal forms. It can be shown, by extending the existence proof for head normal forms of Section 6, that each process P is provably equivalent to a snf. Differently from CCS, each process has in general syntactically different strong normal forms that cannot be reduced to each other by simply using associativity and commutativity of sum. As an example, consider, the pair of terms below, where y, w and z are distinct names and N is a snf:

$$x(y).([y = z]\mathbf{0} + [yz]N) \text{ and } x(y).([y = z]\mathbf{0} + [y = w]N\{w/y\} + [y \notin \{z, w\}]N).$$

In general, to reduce two normal forms to each other, besides the above mentioned laws for sum, we would certainly need the α-equivalence law and other laws to capture 'conditional renaming' like the one described above. However, our construction does not rely on uniqueness of normal forms; we work with their equivalence classes and simply assume existence of a function that associates an arbitrary, and provably must-equivalent, snf to each process.

The compact elements of our cpo will be equivalence classes of *finite strong normal forms (fsnf)*. These can be described informally as the 'finite-breadth' snfs, i.e., as those snfs that, for almost every name z, upon receipt of z become Ω. The formal definition is as follows.
A finite strong normal form is a snf N with associated function g, s.t.

a. $N \equiv \Omega$, or

b. $x \in domain(g)$ implies $g(x)$ is of the form: $x(y).(\sum_{z \in F}[y = z]N_z + [y \notin F]\Omega)$, where each N_z is a fsnf.

2. *The set of finite strong normal form is turned into a poset.*

To guarantee antisimmetry, PO is defined as the poset given by the set of equivalence classes of fsnf's w.r.t. the kernel equivalence $=_M$ of the preorder $\sqsubseteq_{must}$ (i.e. $=_M = \sqsubseteq_{must} \cap \sqsubseteq_{must}{}^{-1}$); the partial ordering $\preceq$ over PO is defined as follows: $[N] \preceq [M]$ iff $N \sqsubseteq_{must} M$; the bottom element is $[\Omega]$. A set of functions over PO, each corresponding to a different operator of the calculus, is defined:

- For functions corresponding to operators different from input and output prefixes, this is done via snf; for example, for a restriction (y), we define $f_{(y)} : PO \longrightarrow PO$ as:

$$f_{(y)}([N]) = [snf((y)N)].$$

- To define function *inp*, consider the poset of functions

$$FIN(\mathcal{N} \longrightarrow PO) = \{f : \mathcal{N} \longrightarrow PO | \text{ only for finitely many } x,\ f(x) \neq [\Omega]\}$$

with the partial ordering pointwise inherited from PO. Function $inp : \mathcal{N} \times FIN(\mathcal{N} \longrightarrow PO) \longrightarrow PO$ is then defined as follows:

$$inp(x, f) = [x(y).(\sum_{z \in F} [y = z]N_z + [y \notin F]\Omega)]$$

where $f(z) = [N_z]$, $F = \{x |\, f(x) \neq [\Omega]\}$ and y is fresh.

- Function $out : \mathcal{N} \times \mathcal{N} \times PO \longrightarrow PO$ is: $out(x, y, [N]) = [\overline{x}y.N]$.

It is easily checked that the above functions are well-defined and monotonic in PO (or $FIN(\mathcal{N} \longrightarrow PO)$ in the case of $inp(x, -)$).

3. *PO and $FIN(\mathcal{N} \longrightarrow PO)$ are extended to algebraic cpo's.*

Ideal completion [Hen88] is used to obtain the algebraic cpo's PO^∞ and $(FIN(\mathcal{N} \longrightarrow PO))^\infty$ and the unique continuous extensions of the above defined monotonic functions; for the sake of simplicity, we will use for the countinous extensions the same names as the monotonic functions they extend. The crucial point here is that, as easily seen, the algebraic cpo $(FIN(\mathcal{N} \longrightarrow PO))^\infty$ is isomorphic to $(\mathcal{N} \longrightarrow PO^\infty)$, thus they can be identified. The wanted domain is then $\mathcal{D} \stackrel{def}{=} PO^\infty$; this together with the extended functions give a natural interpretation of $\mathcal{P}$.

4. *Full abstraction of our interpretation with respect to $\sqsubseteq^+_{\approx_{must}}$ is proven.*

The proof relies on the existence, for each P, of a set of syntactical *Finite-Breadth Approximants FBA(P)* (see Section 6), such that (we denote with *sup* the least upper bound in $\mathcal{D}$):

a. if for each $N \in FBA(P)$ $N \sqsubseteq_{must} Q$ then $P \sqsubseteq_{must} Q$;

b. $[\![FBA(P)]\!] = \{[\![N]\!] |\, N \in FBA(P)\}$ is directed in $\mathcal{D}$ and $[\![P]\!] = sup\,[\![FBA(P)]\!]$.

It is easy to show that for each $N \in FSNF$, $[\![N]\!] = [N]$. By exploiting this fact, a. and b. above and the algebraicity of $\mathcal{D}$, it is not hard to show that for each P and Q: $[\![P]\!] \preceq [\![Q]\!]$ iff $P \sqsubseteq_{must} Q$, and, hence, iff $P \sqsubseteq^+_{\approx_{must}} Q$ (see detailed proofs in Section 6).

5 Dealing with Infinite Agents

In this section we sketch the extensions of the equational and denotational characterizations described in previous sections to the full calculus with recursive definitions. We will study the must preorder only; the may case is trivial and the treatment of the testing preorder can be obtained by "composing may and must. Most of what we are going to do is in fact standard algebraic semantics

in the style of [Hen88] and [DH84]. Below we give all new definitions needed for extending the testing framework:

- The syntax is extended by adding the following two clauses to the grammar of Section 2:

$$P ::= X \mid recX.P$$

  where X ranges over a countable set AV of *agents variables*. The language generated by this grammar will be referred to as $\mathcal{RP}$ (for *Recursive* $\mathcal{P}$). The operator $recX.$ is a binder for the agent variable X, thus notions of bound and free occurrence of an agent variable arise as expected. In the sequel we shall only consider *closed* agents, i.e. those agents without occurrences of free agent variables. Furthermore, we impose the constraint that for each term $recX.P$, no occurrence of X inside P is within the scope of a y-binder, for $y \in fn(P)$. This ensures that no free name of P is bound when "unfolding" the term $recX.P$ into $P[recX.P/X]$ ($[Y/X]$ denotes substitution of the agent variable X with the term Y).

- Transitional semantics is extended by adding the following SOS rule:

$$\texttt{Rec} : \frac{P[recX.P/X] \xrightarrow{\alpha} P'}{recX.P \xrightarrow{\alpha} P'} .$$

- Extending the processes language requires, in the testing framework, extending observers accordingly.

- The definition of predicate $\downarrow$ is extended with the clause:

$$P[recX.P/X]\downarrow \text{ implies } recX.P.\downarrow$$

  This amounts to saying that $P\downarrow$ if and only if in P there are no unguarded occurrences of Ω and agent variables.

- The definition of the convergence predicate ($\Downarrow$) is restated as follows:

$$P \Downarrow \text{ iff } (P \Longrightarrow P' \text{ implies } P'\downarrow) \text{ and } (\text{not } P \xrightarrow{\tau^\omega}) .$$

- An interactions is now either finite, as described in Definition 2.6, or an infinite sequence of τ-transitions.

- The definition of the must relation is extended to take into account the new infinite interactions. Thus P **must** o if and only if for all interactions (finite or infinite):

$$P|o = e_0 \xrightarrow{\tau} e_1 \xrightarrow{\tau} e_2 \xrightarrow{\tau} \cdots$$

$\exists j \geq 0$, such that: $o_j \xrightarrow{\omega}$ and $\forall i \geq 0$, $((e_i \uparrow$ implies $i \geq j)$.

Notice that now we have now another type of unsuccessful computations, namely the infinite sequence of τ-transitions that never reaches a successful state:

$$P|o = e_0 \xrightarrow{\tau} e_1 \xrightarrow{\tau} e_2 \xrightarrow{\tau} \cdots \text{ and } e_i \not\xrightarrow{\omega} \text{ for } i \geq 0\,.$$

In this new setting, we will first show how to extend the proof system presented at the beginning of Section 4 for the finite fragment to get a sound and complete (though infinitary) proof system for the full language. Then we will show that the denotational model of Section 4 is fully abstract for the full language as well.

All soundness proofs for the finite calculus easily extend to the infinite case (see Section 6). In order to get a complete proof systems for closed agents, we add to that for finite agents, introduced at the beginning of Section 4, a law and two inference rules to deal with recursion. The resulting system will be referred to as $\mathcal{A}$. In the sequel, we will let D range over finite processes:

$$REC1 \quad P[recX.P/X] =_M recX.P$$

$$REC2 \quad \frac{P \sqsubseteq_{must} Q}{recX.P \sqsubseteq_{must} recX.Q}$$

$$\omega - induction \quad \frac{\forall D \in FIN(P)\, D \sqsubseteq_{must} Q}{P \sqsubseteq_{must} Q}$$

where $FIN(P)$ are the finite syntactic approximants of P; we skip its actual definition, since it is a standard construction of algebraic semantics and can be found in, e.g., [Hen88]. We wish only to remind the reader that set $FIN(P)$ is exclusively concerned with the approximation *in depth* of agent P: an element of $FIN(P)$ is in fact obtained from P by performing a finite number of unfolding of its recursive definitions (subterms of the form $recX.P'$) and then replacing them with Ω. On the contrary, finite strong normal forms (FBA) provide a *breadth* approximation for finite agents. These two kinds of approximation can be dealt with separately. We have already considered the breadth one when considering finite agents; now, we indicate how to compose that treatment with the depth one.

Soundness of the above rules can be proved, again, along the same lines of [DH84]. Law $REC2$ is derivable from the other axioms in $\mathcal{A}$ (see [DH84]) and to prove soundness of $REC1$ it is sufficient to define $Der_\alpha(P) = \{P' | \, P \xrightarrow{\alpha} P'\}$, and to exploit the following fact:

$$\text{if for each } \alpha \; Der_\alpha(P) = Der_\alpha(Q) \text{ and } (P \uparrow \text{ iff } Q \uparrow) \text{ then } P \simeq Q.$$

The only non-trivial task is proving soundness of $\omega - induction$. Before proving it, it is useful to show *partial completeness* of $\mathcal{A} - \{\omega - induction, REC2\}$, i.e: for each finite D and for each Q, $D \sqsubseteq^+_{\sim must} Q$ implies $\mathcal{A} - \{\omega - induction, REC2\} \vdash D \sqsubseteq_{must} Q$.
To prove this, minor modifications of the completeness proof for finite closed agents of Section 6 are needed. We list them below.

Provably (within $\mathcal{A} - \{\omega - induction, REC2\}$) equivalent head normal forms for convergent agents still exist (see Proposition 6.14); however, this time the proof of the parallel composition case is done by induction on $mc(P)$, the maximal number of internal communications P can perform; this number is finite by a Koenigs lemma argument that relies on the fact that $P \Downarrow$. On the other side, since we are concerned with the must preorder only, divergent agents all collapse to Ω (due to $P + \Omega =_M \Omega$). These considerations allow us to extend the proof of Theorem 6.22 to prove partial completeness.

To prove soundness of $\omega - induction$ rule, it is also useful to have an alternative characterization of the must preorder which is based on a smaller set of observers.

Lemma 5.1 *The must testing preorder $<_M$ over $\mathcal{RP}$ obtained by considering the set of observers without restriction coincides with $\sqsubseteq_{\sim must}$.*

Proof: See Section 6. □

Now, soundness of $\omega - induction$ rule easily follows from the definition of $FIN(P)$ and from the following crucial proposition, which relies on the soundness of $\mathcal{A} - \{\omega - induction, REC2\}$.

Proposition 5.2 *For any process P and any restriction-free observer o, if P **must** o then there exists $D \in FIN(P)$ such that D **must** o.*

Proof: See Section 6. □

We now come to show completeness of $\mathcal{A}$. Suppose that for closed P and Q, $P \sqsubseteq^+_{\sim must} Q$; by partial completeness, for each $D \in FIN(P)$, $\mathcal{A} \vdash D \sqsubseteq_{must} Q$; now, apply $\omega - induction$ rule. Thus we can state:

Theorem 5.3 (Completeness of $\mathcal{A}$) *$P \sqsubseteq^{+}_{\approx_{must}} Q$ implies $\mathcal{A} \vdash P \sqsubseteq_{must} Q$.*

We are left with showing how to build a fully abstract denotational model for the must preorder. Indeed the definition of the model remains unchanged, but that of the semantic function is extended to deal with agent variables and with the recursion operator. Let $ENV = (AV \longrightarrow \mathcal{D})$ be the set of *environments* ranged over by the metavariable ρ; then the interpretation function

$$[\![\, . \,]\!] : \mathcal{RP} \longrightarrow (ENV \longrightarrow \mathcal{D})$$

is defined as as follows:

- $[\![X]\!]\rho = \rho(X)$
- $[\![op(P_1, ..., P_k)]\!]\rho = f_{op}([\![P_1]\!]\rho, ..., [\![P_k]\!]\rho)$, for each k-ary operator different from input and output, $k = 0, 1$ or 2
- $[\![x(y).P]\!]\rho = inp(x, \lambda z \in \mathcal{N}.[\![P\{^{z}/_{y}\}]\!]\rho)$
- $[\![\overline{x}y.P]\!]\rho = out(x, y, [\![P]\!]\rho)$
- $[\![recX.P]\!]\rho = fix(\lambda d \in D.[\![P]\!]\rho[d/X])$.

It can be proved [HP80] that the above definition makes sense and that many of the usual results of traditional algebraic semantics are still valid. The following fact will be essential:

$$\{[\![D]\!]\rho |\, D \in FIN(P)\} \text{ is directed in } \mathcal{D} \text{ and } [\![P]\!]\rho = sup\{[\![D]\!]\rho |\, D \in FIN(P)\}$$

It permits concluding that the denotations of infinite elements are uniquely determined by those of the finite ones. Therefore we state:

Theorem 5.4 (Full abstraction for the full calculus) *$[\![P]\!] \preceq [\![Q]\!]$ in $\mathcal{D}$ if and only if $P \sqsubseteq^{+}_{\approx_{must}} Q$.*

Proof: See Section 6. □

6 Detailed Proofs

6.1 Additional Notations

In addition to that fixed in Section 2, we need some more notation for our proofs:

- $Obj(P) = \{x \in fn(P) |\, x = obj(\alpha)$ for some action α in $P\}$.

- $fn(y_1,...,y_n,\alpha_1,...,\alpha_k,P_1,...,P_h) = \{y_1,...,y_n\} \cup fn(\alpha_1) \cup \ldots \cup fn(\alpha_k) \cup fn(P_1) \cup \ldots \cup fn(P_h)$; similar notations hold for the functions $bn(\,.\,), n(\,.\,)$ and $Obj(\,.\,)$.

- Sometimes, we will identify α-equivalent agents or actions. Formally, this means that α-conversion (application of rule α-conv) will be implicitly used whenever it suffices to make two agents identical.

- Given a hnf H with g as associated function, and $\overline{x} \in Ev(H)$, we write the generic $g(\overline{x})$ simply as a sum $\sum_{z \in N_x} \overline{x}z.H_{\overline{x}z}$, with the convention that $g(\overline{x})$ has a summand of the kind $\overline{x}(w).H_{\overline{x}(w)}$ (i.e. a *bound output summand*) if and only if a special item (w) belongs to N_x.

6.2 Basic Properties

In this subsection we establish some basic facts about the transition system of the π-calculus and about the testing preorders and fix some terminology that will be extensively used in later proofs. The following lemma is essentially borrowed from [MPW89]; only, here, $P, P', ...$ stand for either processes or observers, and actions α, β might be ω as well.

Lemma 6.1

1. *If* $P \xrightarrow{\alpha} P'$ *then* $fn(\alpha) \subseteq fn(P)$ *and* $fn(P') \subseteq fn(P) \cup bn(\alpha)$.

2. *If* $P \xrightarrow{a(y)} P'$, *with* $a = x$ *or* $a = \overline{x}$, *and* $z \notin n(P)$ *then* $P \xrightarrow{a(z)} P''$ *for some* $P'' \equiv_\alpha P'\{z/y\}$.

3. *If* $P \xrightarrow{\alpha} P'$, $bn(\alpha) \cap fn(P'\sigma) = \emptyset$ *and* σ *is injective when restricted to* $fn(P)$, *then for some* $P'' \equiv_\alpha P'\sigma$, $P\sigma \xrightarrow{\alpha\sigma} P''$.

4. *If* $P\{w/z\} \xrightarrow{\alpha} P'$, *with* $w \notin fn(P)$ *and* $bn(\alpha) \cap fn(P, w) = \emptyset$, *then for some* Q *and* β *with* $Q\{w/z\} \equiv_\alpha P'$ *and* $\beta\sigma = \alpha$, $P \xrightarrow{\beta} Q$.

5. *Suppose* $P \equiv_\alpha Q$. *Then:*

   i. *If* α *is a free action and* $P \xrightarrow{\alpha} P'$ *then for some* $Q' \equiv_\alpha P'$, $Q \xrightarrow{\alpha} Q'$.

   ii. *If* $P \xrightarrow{a(y)} P'$, *with* $a = x$ *or* $a = \overline{x}$, *then for every* $z \notin n(Q)$ *there exists a* Q' *with* $P'\{z/y\} \equiv_\alpha Q'$ *such that* $Q \xrightarrow{a(z)} Q'$.

PROOF: The proof of each item is an induction on the depth of the inference of $P \xrightarrow{\alpha} P'$ from the rules (action induction), and is obtained from the proof in [MPW89] by simply adding the case when the last applied rule is `Mismatch`. □

Item 3 above, which states a sort of 'monotonicity' of substitutions, contains an additional hypothesis on σ, when compared to the corresponding item of [MPW89]: σ is required to be injective on P. This restriction is due to the presence of the mismatch operator; in fact, if the two distinct names x and y of $[x \neq y]P$ are identified when applying a substitution, the resulting process is blocked (indeed, it is equivalent to **0**), and the property does not hold. Item 1 is very intuitive, it will be often used without mentioning.

We introduce now the technical notion of equivalent experiments, that will be extensively used in later proofs.

Definition 6.2 (Equivalent sequences of τ-actions) *Given any two sequences of τ-transitions s_1, s_2 of the same length ($k \geq 0$):*

$$s_1 = (e_1 \xrightarrow{\tau} ... \xrightarrow{\tau} e_k) \text{ and } s_2 = (f_1 \xrightarrow{\tau} ... \xrightarrow{\tau} f_k)$$

we say that s_1 and s_2 are equivalent *if and only if for all i, $0 \leq i \leq k$, $\nu \in \{\omega, \tau\}$ ($e_i \xrightarrow{\nu}$ if and only if $f_i \xrightarrow{\nu}$) and ($e_i \uparrow$ if and only if $f_i \uparrow$).*

Definition 6.3 (Equivalent experiments)

- *Given two experiments e_1, e_2, we set $e_1 \leq e_2$ if and only if for each invisible sequence s_1 from e_1 there exists an equivalent invisible sequence s_2 from e_2; $\leq$ is clearly a preorder.*
- *We say that e_1 is* equivalent *to e_2, if and only if $e_1 \leq e_2$ and $e_2 \leq e_1$.*

Since computations are particular sequences of τ-actions, we can use the notion of equivalent experiments to establish testing preorder relations between processes. Indeed, if $e_1 \equiv [P|o]$ and $e_2 \equiv [Q|o]$ are two experiments, we have that:

- $e_1 \leq e_2$ implies (P **may** o implies Q **may** o), and
- $e_1 \leq e_2$ implies (P **must** o implies Q **must** o).

Thus, a possible technique for proving that $P \sqsubseteq_{may} Q$, would be proving that $P|o \leq Q|o$ for each observer o. Analogously $P \sqsubseteq_{must} Q$ can be proved by showing that $Q|o \leq P|o$ for each observer o. It follows that if $P|o$ is equivalent to $Q|o$ for each o, we have $P \simeq Q$ (i.e. P and Q are testing equivalent). A useful result about sequences of invisible actions is reported below.

Lemma 6.4 *If $e_2 \equiv_\alpha e_1\{^{w'}/_{w}\}$, with $w = w'$ or $w' \notin fn(e_1)$, then $e_1 \leq e_2$.*

PROOF: An easy induction on the length of the sequences of invisible actions performable by the two experiments; it relies on items 3 and 5 of Lemma 6.1. □

The above lemma permits identifying α-equivalent experiments, since $P|o \equiv_\alpha P'|o'$ implies P **may** o if and only if P' **may** o' and P **must** o if and only if P' **must** o'. Next lemma states some basic properties of our preorders, that will be used in later proofs.

Lemma 6.5 *Given $P, Q \in \mathcal{P}$ and $\alpha \in \Lambda$:*

1. *($P \sqsubseteq_{may} Q$ and $P \stackrel{\alpha}{\Longrightarrow}$) imply $Q \stackrel{\alpha'}{\Longrightarrow}$, with $\alpha' \equiv_\alpha \alpha$;*
2. *($P \sqsubseteq^+_{must} Q$ and $P \Downarrow$ and $Q \stackrel{\alpha}{\Longrightarrow}$) imply ($Q \Downarrow$ and $P \stackrel{\alpha'}{\Longrightarrow}$, with $\alpha' \equiv_\alpha \alpha$);*
3. *($P \sqsubseteq^+_{must} Q$ and $P \Downarrow$ and $Q \stackrel{\tau}{\longrightarrow}$) imply $P \stackrel{\tau}{\longrightarrow}$.*

PROOF:

1. We first show that P may satisfy a certain observer o, whose form depends on the kind of the performed action α, and which is defined as follows:

- if $\alpha = \overline{x}y$ then $o \equiv x(z).[z = y]\omega.\mathbf{0}$, $z \neq y$;
- if $\alpha = x(y)$ then $o \equiv \overline{x}w.\omega.\mathbf{0}$, with w any name;
- if $\alpha = \overline{x}(y)$ then $o \equiv x(z).[z \notin Obj(P,Q)]\omega.\mathbf{0}$ (recall that $Obj(P,Q) = \{y \in fn(P,Q) |\, y$ is in object position in P or in $Q\}$) with $z \notin Obj(P,Q)$.

A simple case analysis on α shows that $P \stackrel{\alpha}{\Longrightarrow}$ implies P **may** o, which in turn implies Q **may** o. But the latter, as easily seen (by another simple case analysis on α), implies that $Q \stackrel{\alpha'}{\Longrightarrow}$, with $\alpha' \equiv_\alpha \alpha$. Notice that in case $\alpha = \overline{x}(y)$, we rely on the ability of P to send a fresh name to o.

2. First we show that $P \Downarrow$ implies $Q \Downarrow$: $P \Downarrow$ implies P **must** $\tau.\omega.\mathbf{0}$ implies Q **must** $\tau.\omega.\mathbf{0}$ implies $Q \Downarrow$. Next, suppose by contradiction that for no $\alpha' \equiv_\alpha \alpha$, $P \stackrel{\alpha}{\Longrightarrow}$. Then we can exhibit an observer o, depending on the kind of the performed α, such that P **must** o and Q ~~must~~ **must** o, that contradicts $P \sqsubseteq_{must} Q$. The observer o is defined as follows:

- $\alpha = \overline{x}y$ implies $o \equiv x(z).[z \neq y].\omega.\mathbf{0} + \tau.\omega.\mathbf{0}$, $z \neq y$;
- $\alpha = x(y)$ implies $o \equiv \overline{x}w.\mathbf{0} + \tau.\omega.\mathbf{0}$;
- $\alpha = \overline{x}(y)$ implies $o \equiv x(z).([z = y_1]\omega.\mathbf{0} + ... + [z = y_k]\omega.\mathbf{0}) + \tau.\omega.\mathbf{0}$, with $Obj(P,Q) = \{y_1, ..., y_k\}$, $k \geq 0$ (if $Obj(P,Q) = \emptyset$ then the above summation reduces to $\mathbf{0}$) and with $z \notin Obj(P,Q)$.

Then, since $P \Downarrow$ and $P \not\stackrel{\alpha'}{\Longrightarrow}$ for each $\alpha' \equiv_\alpha \alpha$, by a simple case analysis on α, we can show that P **must** o, but Q **must** o, which is impossible. Thus there must exist some $\alpha' \equiv_\alpha \alpha$ such that $P \stackrel{\alpha'}{\Longrightarrow}$. Notice that, this time in the proof of case $\alpha = \overline{x}(y)$, we rely on the ability of Q to send a fresh (hence not belonging to $Obj(P,Q)$) name to o and on the inability of P to do so.

3. Suppose $Q \stackrel{\tau}{\longrightarrow} Q'$ and $P \not\stackrel{\tau}{\longrightarrow}$. Take $x \notin fn(P,Q)$ and $o \equiv x(y).\omega.\mathbf{0}$. Then $P + \overline{x}y.\mathbf{0}$ **must** o: in fact $P \Downarrow$ and since $P \not\stackrel{\alpha}{\longrightarrow}$ with α **comp** $x(y)$ (in virtue of Lemma 6.1(1)) and $P \not\stackrel{\tau}{\longrightarrow}$, the only possible computation is $(P + \overline{x}y.\mathbf{0})|o \stackrel{c}{\longrightarrow} \mathbf{0}|\omega.\mathbf{0}$, that is successful. But $Q + \overline{x}y.\mathbf{0}$ **must** o, due to the unsuccessful computation starting as $(Q + \overline{x}y.\mathbf{0})|o \stackrel{\tau}{\longrightarrow} Q'|o$, (where we have exploited the fact that, by Lemma 6.1(1), for no α such that α **comp** $\overline{x}y$, $Q' \stackrel{\alpha}{\Longrightarrow}$, and thus $Q'|o \not\stackrel{c}{\Longrightarrow}$). But this contradicts the fact that $P \sqsubset^{+}_{\approx_{must}} Q$. □

6.3 Soundness Proofs

We need some preliminary lemmata. The following two will be used to show that the restriction operator preserves the testing preorder.

Lemma 6.6 *For all P, o and $y \notin fn(o)$: $(y)P|o$ and $P|o$ are equivalent as experiments.*

PROOF: Recall from Definition 6.3 that we have to show that for each P, o and $y \notin fn(o)$, (i) $P|o \leq (y)P|o$, and (ii) $(y)P|o \leq P|o$. Define $e \leq_k f$ iff for each sequence (of τ-actions) of length k from e there exists an equivalent sequence from f; clearly, $e \leq f$ iff for each $k \geq 0$ $e \leq_k f$. We now come to the actual proof of (i) and (ii).

(i) We show by induction on k that for each sequence $P|o \Longrightarrow [P_k|o_k]$ of length k there exists an equivalent sequence from $(y)P|o$, i.e $P|o \leq_k (y)P|o$. Suppose $k > 0$ and $P|o \stackrel{\tau}{\longrightarrow} [P_1|o_1] \Longrightarrow [P_k|o_k]$. It is easily seen that $(P|o) \downarrow$ iff $((y)P|o) \downarrow$ and $P|o \stackrel{\omega}{\longrightarrow}$ iff $(y)P|o \stackrel{\omega}{\longrightarrow}$. Thus it suffices to show, by exploiting the induction hypothesis, that $(y)P|o \stackrel{\tau}{\longrightarrow} e$, where e is an experiment such that $[P_1|o_1] \leq_{k-1} e$. This will be done by a case analysis on how transition $P|o \stackrel{\tau}{\longrightarrow} [P_1|o_1]$, which we call TR, is inferred.

1. The cases of the form:

$$\texttt{Par} : \frac{P \stackrel{\tau}{\longrightarrow} P'}{TR}, \qquad \texttt{Par} : \frac{o \stackrel{\tau}{\longrightarrow} o'}{TR}, \qquad \texttt{R} : \frac{P \stackrel{\alpha}{\longrightarrow} P' \quad o \stackrel{\beta}{\longrightarrow} o'}{TR}\ \alpha \textbf{ comp } \beta,$$

with $\texttt{R} = \texttt{Com}$ or $\texttt{Close}$ and $y \notin n(\alpha, \beta)$, are obvious. In fact it is easily seen that a transition $(y)P|o \stackrel{\tau}{\longrightarrow} [(y)P_1|o_1]$ always exists such that $y \notin fn(o_1)$, thus the thesis follows from the

inductive hypothesis [3].

2.

$$\texttt{Com}: \frac{P \xrightarrow{\overline{x}y} P' \quad o \xrightarrow{x(w)} o'}{P|o \xrightarrow{\tau} P'|o'\{y/w\} \equiv P_1|o_1}.$$

We can rename w into a fresh name $w' \notin n(y, P', o')$, thus:

$$\texttt{Lemma 6.1(2)}: \frac{o \xrightarrow{x(w)} o'}{o \xrightarrow{x(w')} o'' \equiv_\alpha o'\{w'/w\}}.$$

Now, consider the following derivation:

$$\texttt{Close}: \frac{\texttt{Open}: \dfrac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(w')} P'\{w'/y\}} \quad o \xrightarrow{x(w')} o''}{(y)P|o \xrightarrow{\tau} (w')(P'\{w'/y\}|o'') \equiv e}$$

Since $y \notin fn(o)$, from Lemma 6.1(1) it follows $w = y$ or $y \notin fn(o')$, hence we have:

$$(P_1|o_1)\{w'/y\} \equiv P'\{w'/y\}|o'\{y/w\}\{w'/y\} \equiv_\alpha P'\{w'/y\}|o'' \equiv e'$$

and from Lemma 6.4 we get $P_1|o_1 \leq e'$. But $e \equiv (w')e'$ is trivially equivalent as experiment to e' and thus $P_1|o_1 \leq e$, and the thesis follows.

3.

$$\texttt{Com}: \frac{P \xrightarrow{x(y)} P' \quad o \xrightarrow{\overline{x}w} o_1}{TR}.$$

Rename y into a fresh y' via Lemma 6.1(2), as done in the above case 1 with w.

4.

$$\texttt{Close}: \frac{P \xrightarrow{\overline{x}(y)} P' \quad o \xrightarrow{x(y)} o'}{TR}.$$

Rename y into a fresh y' via Lemma 6.1(2), as done in case 1 with w.

5. All remaining cases are such that:

$$\texttt{R}: \frac{P \xrightarrow{\alpha} P' \quad o \xrightarrow{\beta} o'}{P|o \xrightarrow{\tau} R_{\alpha\beta} \equiv [P_1|o_1] = TR\,\alpha\ \mathbf{comp}\ \beta}$$

[3] Given two equivalent experiments, prefixing by restrictions one or both of them does not affect their equivalence as experiments

where $\mathtt{R} = \mathtt{Com}$ or $\mathtt{R} = \mathtt{Close}$ and $y \in n(\beta)$. It must be then $y = bn(\beta)$: it cannot be $subj(\beta) = y$ due to the hypothesis $y \notin fn(o)$. The subcase $\beta = x(y)$ and $\alpha = \overline{x}w$, $w \neq y$, is trivial. Then the only remaining subcase is when $\beta = \overline{x}(y)$ and, hence, $\alpha = x(y)$. Therefore it is $\mathtt{R} = \mathtt{Close}$ and $R_{\alpha\beta} \equiv (y)(P'|o')$. Taken a fresh name z we have the following derivation:

$$\mathtt{Close}: \dfrac{\mathtt{Res}: \dfrac{\mathtt{Lemma\ 6.1(2)}: \dfrac{P \xrightarrow{\alpha} P'}{P \xrightarrow{x(z)} P'' \equiv_\alpha P'\{z/y\}}}{(y)P \xrightarrow{x(z)} (y)P''} \qquad \mathtt{Lemma\ 6.1}}{(y)P|o \xrightarrow{\tau} (z)((y)P''|o'') \equiv e}$$

Now, $(z)(P''|o'') \equiv_\alpha (y)(P'|o') \equiv [P_1|o_1]$ and $P''|o''$ is equivalent to $(z)(P''|o'')$ as experiment. Therefore, $P''|o''$ is equivalent to $[P_1|o_1]$. By applying the inductive hypothesis, we have $P''|o'' \leq_{k-1} (y)P''|o'' \equiv e'$, since $y \notin fn(o'')$. Thus, it follows that also $[P_1|o_1] \leq_{k-1} e'$; but $e \equiv (z)e'$ is trivially equivalent to e', thus $[P_1|o_1] \leq_{k-1} e$, and the thesis follows.

(ii) This half is done in the same style of part (i). The only remarkable difference is, in the inductive step, when the $\mathtt{Close}$ transition of $(y)P|o$ given by:

$$\mathtt{Close}: \dfrac{\mathtt{Open}: \dfrac{P \xrightarrow{\overline{x}y} P'}{(y)P \xrightarrow{\overline{x}(w)} P'\{w/y\}} \quad o \xrightarrow{x(w)} o'}{(y)P|o \xrightarrow{\tau} (w)(P'\{w/y\}|o') \equiv [P_1|o_1]}$$

with $w \notin fn((y)P')$, is emulated by $P|o$ by the $\mathtt{Com}$ transition given by:

$$\mathtt{Com}: \dfrac{P \xrightarrow{\overline{x}y} P' \quad o \xrightarrow{x(w)} o'}{P|o \xrightarrow{\tau} P'|o'\{y/w\} \equiv e}.$$

Now, since $w \notin fn((y)P')$, it is $e \equiv P'|o'\{y/w\} \equiv_\alpha (P_1|o_1)\{y/w\}$ and thus, from Lemma 6.4, $P_1|o_1 \leq e$. But, being $P_1|o_1$ trivially equivalent to $[P_1|o_1]$, it follows that $[P_1|o_1] \leq e$, and thus the claim follows. □

Of course, a statement symmetrical to that of the previous lemma holds for $P|(y)o$ and $P|o$, if $y \notin fn(P)$.

Lemma 6.7 *Let w, y be two distinct names; $P \sqsubseteq Q$ and $w \notin fn(P,Q), y \notin fn(o)$ imply:*

i. $P\{w/y\}$ **may** *o implies* $Q\{w/y\}$ **may** *o;*

ii. $P\{w/y\}$ **must** *o implies* $Q\{w/y\}$ **must** *o.*

PROOF:

i. We have that $(P\{^{w}/_{y}\}|o)\{^{y}/_{w}\} \equiv_\alpha (P|o\{^{y}/_{w}\})$, with $y \notin fn(P\{^{w}/_{y}\}|o)$. Then by Lemma 6.4 we have $P\{^{w}/_{y}\}|o \leq (P|o\{^{y}/_{w}\})$ and thus P **may** $o\{^{y}/_{w}\}$. This implies Q **may** $o\{^{y}/_{w}\}$. But $(Q|oyw)\{^{w}/_{y}\} \equiv_\alpha (Q\{^{w}/_{y}\}|o)$, with $w \notin fn(Q|oyw)$. Therefore from Lemma 6.4 we have $Q|o\{^{w}/_{y}\} \leq (Q\{^{w}/_{y}\}|o)$ and therefore the thesis follows.

ii. Suppose $Q\{^{w}/_{y}\}$ **must** o. Then by a reasoning dual to i. (substituting ' **may** ' with ' **must** ' and interchanging 'P' and 'Q' the above chain of implications still holds) we can show that it is $P\{^{w}/_{y}\}$ **must** o. □

A relevant consequence of the above lemma is the following proposition, which asserts that our preorder is preserved by substitutions under certain assumptions:

Proposition 6.8 *$P \sqsubseteq Q$ and $z \notin fn(P,Q)$ imply $P\{^{z}/_{y}\} \sqsubseteq Q\{^{z}/_{y}\}$.*

PROOF: Suppose $P\{^{z}/_{y}\}$ **may** o. If $y \notin fn(o)$, we apply the previous lemma, otherwise take y' fresh and suppose $z \neq y$. Consider then the following implications:

$P\{^{z}/_{y}\}$ **may** o	implies $P\{^{z}/_{y}\}$ **may** $o\{^{y'}/_{y}\}$	(Lemma 6.4, since $P\{^{z}/_{y}\}|o\{^{y'}/_{y}\} \equiv_\alpha (P\{^{z}/_{y}\}|o)\{^{y'}/_{y}\}$)
	implies $Q\{^{z}/_{y}\}$ **may** $o\{^{y'}/_{y}\}$	(Lemma 6.7)
	implies $Q\{^{z}/_{y}\}$ **may** $o\{^{y'}/_{y}\}\{^{y}/_{y'}\}$	(Lemma 6.4)
	implies $Q\{^{z}/_{y}\}$ **may** o	(Lemma 6.4) .

By a dual reasoning (interchanging 'P' and 'Q' and substituting ' **may** ' with ' **must** ' in the above chain of implications) we can show that $Q\{^{z}/_{y}\}$ **must** o implies $P\{^{z}/_{y}\}$ **must** o. □

The following lemma will be used to prove that our preorder is preserved by the parallel composition operator:

Lemma 6.9 *Let $P, R \in \mathcal{P}$, $o \in O$. Then:*

i. *P* **may** *$R|o$ iff $P|R$* **may** *o;*

ii. *P* **must** *$R|o$ iff $P|R$* **must** *o.*

PROOF: It suffices to show that $P|(R|o)$ and $(P|R)|o$ are equivalent as experiments. The structure of the proof is very similar to that of Lemma 6.6. We proof only that $(P|R)|o \leq P|(R|o)$ half, the other direction is perfectly symmetrical.

Suppose $(P|R)|o \xrightarrow{\tau} [[P_1|R_1]|o_1] \Longrightarrow [[P_k|R_k]|o_k]$, $k > 0$. It is obvious that $(P|R)|o \uparrow$ iff $P|(R|o) \uparrow$ and that $(P|R)|o \xrightarrow{\omega}$ iff $P|(R|o) \xrightarrow{\omega}$. Thus it suffices to show that $P|(R|o) \xrightarrow{\tau} e$, where e is an experiment such that $[[P_1|R_1]|o_1] \leq_{k-1} e$ (recall the definition of $\leq_k$ from the proof of Lemma 6.6). This will be done with a case analysis on the proof of transition $(P|R)|o \xrightarrow{\tau} [[P_1|r_1]|o_1]$; later we will refer to this transition by calling it TR.

1. The cases of the form

$$\texttt{Par} : \frac{\texttt{Par} : \dfrac{P \xrightarrow{\tau}}{P|R \xrightarrow{\tau}}}{TR}, \qquad \texttt{Par} : \frac{\texttt{Par} : \dfrac{R \xrightarrow{\tau}}{P|R \xrightarrow{\tau}}}{TR} \quad \text{and} \quad \texttt{Par} : \frac{o \xrightarrow{\tau}}{TR}$$

   are obvious, applying the inductive hypothesis.

2. $$\texttt{Par} : \frac{\texttt{Close} : \dfrac{P \xrightarrow{\alpha} P' \quad R \xrightarrow{\beta} R'}{P|R \xrightarrow{\tau} (w)(P'|R')}}{(P|R)|o \xrightarrow{\tau} (w)(P'|R')|o \equiv [[P_1|r_1]|o_1]}$$

   where α **comp** β and $bn(\alpha) = bn(\beta) = w$. We can suppose $w \notin fn(o)$ without loss of generality, since, via Lemma 6.1(2), we can always rename w into a fresh w' in the above derivation, obtaining an experiment equivalent to $[[P_1|R_1]|o_1]$ as a result. Thus we can consider the derivation:

$$\texttt{Close} : \frac{P \xrightarrow{\alpha} P' \qquad \texttt{Par} : \dfrac{R \xrightarrow{\beta} R'}{R|o \xrightarrow{\beta} R'|o}}{P|(r|o) \xrightarrow{\tau} (w)(P'|(R'|o)) \equiv e} .$$

   By applying the inductive hypothesis, we get $(P'|R')|o \leq_{k-1} P'|(R'|o) \equiv e'$; but e' is trivially equivalent to e, and $(P'|R')|o$ is equivalent to $(w)(P'|R')|o \equiv [[P_1|o_1]|R_1]$ in virtue of Lemma 6.6, since $w \notin fn(o)$, therefore we have $[[P_1|R_1]|o_1] \leq_{k-1} e$.

   All other cases are similar to, or easier than, case 2. The corresponding proofs are routine, and are omitted.

□

We come to the actual soundness proofs for inference rules and axioms.

Proposition 6.10 (Proposition 3.1) *The inference rules of Table 3 are sound w.r.t.* $\sqsubseteq^{+}_{\approx}$.

PROOF:

($\alpha - conv$). The soundness of $\alpha - conv$ is an obvious consequence of Lemma 6.4: in fact, for every observer o, $P \equiv_\alpha Q$ implies $P|o \equiv_\alpha Q|o$, and the latter are equivalent experiments.

($C0$). We omit the proofs for the free actions prefixes (τ and free output) and summation cases, which are much the same as for CCS [DH84].

- The match case is trivial if $x \neq y$, otherwise it is proven by showing that for each o, $[x = x]P|o$ and $P|o$ are equivalent as experiments, which is easy. The mismatch case is analogous. Let us see in more detail the next two cases, which are harder.

- Restriction. Suppose $P \sqsubseteq^+_\approx Q$. Given any observer o, take $y' \notin fn(P, Q, o)$ and $P' \equiv P\{y'/y\}$, $Q' \equiv Q\{y'/y\}$. Then we have the following chain of implications:

$$\begin{array}{lll} (y)P \textbf{ may } o & \text{implies } (y')P' \textbf{ may } o & (\alpha\text{-equivalence of experiments}) \\ & \text{implies } P' \textbf{ may } o & (\text{Lemma 6.6}) \\ & \text{implies } Q' \textbf{ may } o & (\text{Proposition 6.8}) \\ & \text{implies } (y')Q' \textbf{ may } o & (\text{Lemma 6.6}) \\ & \text{implies } (y)Q \textbf{ may } o & (\alpha\text{-equivalence})\,. \end{array}$$

  Thus $(y)P \sqsubseteq_{may} (y)Q$. Now, by a dual reasoning (substitute ' **may** ' with ' **must** ' and exchange 'P' and 'Q' in the above implications) we show that, for any o, $(y)Q$ **must** o implies $(y)P$ **must** o, that is $(y)P \sqsubseteq_{must} (y)Q$. Moreover, if $(y)P \Downarrow$ then also $P \Downarrow$; thus we have that: $(y)Q \xrightarrow{\tau}$ implies $Q \xrightarrow{\tau}$ implies $P \xrightarrow{\tau}$ (from the definition of $\sqsubseteq^+_{must}$) implies $(y)P \xrightarrow{\tau}$. We have showed that $(y)P \sqsubseteq^+_{must} (y)Q$.

- Parallel Composition. Suppose $P \sqsubseteq^+_\approx Q$. Then, for any observer o, we have the following chain of implications:

$$\begin{array}{lll} P|R \textbf{ may } o & \text{implies } P \textbf{ may } R|o & (\text{Lemma 6.9}) \\ & \text{implies } Q \textbf{ may } R|o & (P \sqsubseteq_{may} Q) \\ & \text{implies } Q|r \textbf{ may } o & (\text{Lemma 6.9}) \end{array}$$

  Thus $P|R \sqsubseteq_{may} Q|R$. By a similar reasoning, we show that $P|R \sqsubseteq_{must} Q|R$. Moreover, if $(P|R) \Downarrow$ then it must be also $P \Downarrow$ and $R \Downarrow$; therefore, if $Q|R \xrightarrow{\tau}$, a simple case analysis (on why $Q|R \xrightarrow{\tau}$) shows that also $P|R \xrightarrow{\tau}$. Thus $P|R \sqsubseteq^+_{must} Q|R$.

- Finally, the soundness of the congruence rule for bound output prefix is an easy consequence of the free output prefix and of restriction cases (recall that $\overline{x}(y).P$ stands for $(y)(\overline{x}y.P)$).

($C1$). Suppose $y(z).P$ **may** o. Besides the trivial case in which $o \stackrel{\omega}{\Longrightarrow}$, it must be $y(z).P|o \Longrightarrow y(z).P|o_k \stackrel{c}{\longrightarrow} [P_{k+1}|o_{k+1}]$, where P_{k+1} **may** o_{k+1} and $P_{k+1} \equiv P\sigma$, for a suitable $\sigma = \{x/z\}$. Now, if $x \in fn(z, P, Q)$, from the hypothesis we have also $Q\sigma$ **may** o_{k+1}, from which it easily follows that $y(z).Q$ **may** o. Otherwise $x \notin fn(z, P, Q)$, and then from Proposition 6.8 we have again that $Q\sigma$ **may** o_{k+1} and hence $y(z).Q$ **may** o.

Suppose $y(z).Q$ **must** o. Besides the trivial case in which $o \Longrightarrow o_k$ with $o_k \uparrow$ and $o_i \not\stackrel{\omega}{\longrightarrow}$ for $0 \leq i \leq k$, it must hold one of the following:

1. $y(z).Q|o \Longrightarrow y(z).Q|o_k \stackrel{c}{\longrightarrow} [Q\sigma|o_{k+1}]$, where $o_i \not\stackrel{\omega}{\longrightarrow}$ for $0 \leq i \leq k$ and $Q\sigma$ **must** o_{k+1}, for a suitable $\sigma = \{x/z\}$. Now, if $x \notin fn(z, P, Q)$, from Proposition 6.8 we have that it is also $P\sigma$ **must** o_{k+1}, and then it easily follows that $y(z).P$ **must** o; otherwise, this result follows directly from the hypotheses.

2. $y(z).Q|o \Longrightarrow y(z).Q|o_k \not\stackrel{\tau}{\longrightarrow}$, where $o_i \not\stackrel{\omega}{\longrightarrow}$ for $0 \leq i < k$. Then it is easily seen that an equivalent computation from $y(z).P|o$ exists, thus $y(z).P$ **must** o. Furthermore, $y(z).Q \not\stackrel{\tau}{\longrightarrow}$, so that it is $y(z).P \sqsubseteq^{+}_{\approx_{must}} y(z).Q$. □

Proposition 6.11 (Proposition 3.2) *The axioms of Table 2 are sound w.r.t.* $\sqsubseteq^{+}_{\approx}$.

Proof:

- Define for each P and α, $Der_\alpha(P) = \{P' \mid P \stackrel{\alpha}{\longrightarrow} P'\}$. The soundness of the laws in Sum, Match, Mismatch, of laws R2, R4 and of Expansion follows from the following (easy to prove) fact: if for each α, $Der_\alpha(P) = Der_\alpha(Q)$ and ($P \uparrow$ iff $Q \uparrow$) then $P \sqsubseteq^{+}_{\approx} Q$ and $Q \sqsubseteq^{+}_{\approx} P$.

- ($R0$). The soundness follows from the following fact: for each o and each P, $x \notin fn(P)$ implies $(x)P|o$ and $P|o$ are equivalent as experiments. If $x \notin fn(o)$, this fact follows from Lemma 6.6; otherwise, first rename x into a fresh x'.

- ($R1$). Follows from the following fact: for each o, $(x)(y)P|o$ and $(y)(x)P|o$ are equivalent as experiments. To prove this fact, consider $x', y' \notin n(P, o)$ and $P' \equiv P\{x'/x\}\{y'/y\}$. Then one has, for each o: $(x)(y)P|o \equiv_\alpha (x')(y')P'|o$, which in turn is equivalent, by using twice Lemma 6.6, to $P'|o$, which is equivalent, (Lemma 6.6 twice again, but in the opposite direction), to $(y')(x')P'|o \equiv_\alpha (y)(x)P|o$.

- ($R3$). Consider first the case in which α is not a bound output; in order to prove soundness of the law, we first show the following facts:
  a) for each $o, x \notin n(\alpha)$: $\alpha.(x)P|o \stackrel{c}{\longrightarrow} [P'|o']$ implies $(x)\alpha.P|o \stackrel{c}{\longrightarrow} [P''|o''] \equiv_\alpha [P'|o']$;

b) for each $o, x \notin n(\alpha)$: $(x)\alpha.P|o \xrightarrow{c} [P'|o']$ implies $\alpha.(x)P|o \xrightarrow{c} [P'|o']$.

a) is proven by a case analysis on why $\alpha.(x)P|o \xrightarrow{c} [P'|o']$. The only non trivial case is when $\alpha = z(y), z, y \neq x$, and

$$\texttt{Close}: \frac{\alpha.(x)P \xrightarrow{z(x)} (x')P\{x'/x\}\{x/y\} \quad o \xrightarrow{\overline{z}(x)} o'}{\alpha.(x)P|o \xrightarrow{c} (x)((x')P\{x'/x\}\{x/y\}|o')} \ , z \neq x$$

(notice that here we have made explicit the α-renaming of x into a fresh x' caused by the substitution $\{x/y\}$). Then, from Lemma 6.1(2), we have that $o \xrightarrow{\overline{z}(x'')} o'' \equiv_\alpha o'\{x''/x\}$, x'' fresh, so that we have the derivation:

$$\texttt{Close}: \frac{(x)\alpha.P \xrightarrow{z(x'')} (x)P\{x''/y\} \quad o \xrightarrow{\overline{z}(x'')} o''}{(x)a.P|o \xrightarrow{c} (x'')((x)P\{x''/y\}|o'') \equiv_\alpha (x)((x')P\{x'/x\}\{x/y\}|o')}$$

b) is trivial to prove. Statements a) and b) together are sufficient to prove that $x \notin n(\alpha)$ implies that $\alpha.(x)P|o$ and $(x)\alpha.P|o$ are equivalent as experiments when α is not a bound output. Finally, if α is a bound output apply first law R1 then law R3.

- (*Omega*). Trivial.

- (τ-laws). The proofs for these laws are very similar to those of the corresponding ones for CCS in [DH84]. As an example, we prove N3, confining ourselves to the 'must' part (the 'may' is trivial). We deal with the three cases ($\alpha = \beta = \tau$, $\alpha, \beta \in IA$ and $\alpha, \beta \in OA$) uniformely. In the sequel, 'P **moves-on** o' means '$P|o \stackrel{c}{\Longrightarrow}$ or $P \xrightarrow{\tau}$'. Let $T \equiv \alpha.P + \tau.(\beta.Q + R)$ and $U \equiv \tau.(\alpha.P + \beta.Q + R)$. Suppose P **must** o. Then it is either $\alpha.P$ **must** o or $\tau.(\beta.Q + R)$ **must** o. Let us exclude the trivial case in which $o \Longrightarrow o_k$, with $o_k \uparrow$ and $o_i \stackrel{\omega}{\not\longrightarrow}$ for $0 \leq i \leq k$. Suppose $\alpha.P$ **must** o; then it is easily seen that if $\alpha.P$ **moves-on** o also U **must** o, otherwise (i.e. not $\alpha.P$ **moves-on** o) it must be $\beta.Q + R$ **must** o and hence also U **must** o. In the second case, i.e. $\tau.(\beta.Q + R)$ **must** o, it is also $\beta.Q + R$ **must** o. Now, if $\beta.Q + R$ **moves-on** o, then it is easily seen that also U **must** o; otherwise, if not $\beta.Q + R$ **moves-on** o, then there exists a computation $(\beta.Q + R)|o \Longrightarrow (\beta.Q + R)|o_k \stackrel{\tau}{\not\longrightarrow}$. But then, since $subj(\alpha) = subj(\beta)$, a unsuccessful computation $U|o \Longrightarrow (\alpha.P + \beta.Q + R)|o_k \stackrel{\tau}{\not\longrightarrow}$ exists. Thus we have proven that $U \sqsubseteq_{must} T$. The converse (U **must** o) is similar.

□

$D1$	$\tau.P + \Omega = P + \Omega$	
$D2$	$\alpha.(\sum_{i\in I} \tau.P_i) = \sum_{i\in I} \alpha.P_i$	I finite non-empty set
$D3$	$\alpha.P + \Omega = \alpha.(P + \Omega) + \Omega$	
$D4$	$P + \tau.Q = \tau.(P + Q) + \tau.Q$	
$D5$	$\alpha.P + \alpha.Q = \alpha.P + \alpha.Q + \alpha.(P + Q)$	
$D6$	$\alpha.P + \alpha.(P + Q + R) = \alpha.P + \alpha.(P + Q) + \alpha.(P + Q + R)$	
$D7$	$\alpha.\tau.P = \alpha.P$	
$D8$	$P + \tau.(P + Q) = \tau.(P + Q)$	
$D9$	$\alpha.(P + \tau.Q) + \alpha.Q = \alpha.(P + \tau.Q)$	
$D10$	$\Omega + \alpha.(\sum_{i\in I} P_i + \Omega) = \Omega + \sum_{i\in I} \alpha.P_i$	I finite non-empty set

Table 4: Derived Laws

6.4 Completeness proofs

We list in Table 6.4 a few derived laws, borrowed from [DH84], that will be useful in later proofs.

Their derivation from the axioms and the inference rules of $\mathcal{F}$ is the same as in [DH84], except that each application of the action-prefix congruence rule has to be replaced by an application of either C0 - output or C1; this presents no problem. We have now to proof some preliminary lemmata about head normal forms. The first task is to show that each process has a provably equivalent hnf or Ω-hnf.

Lemma 6.12 *If P and Q are hnf's then there exists a hnf H s.t. $\mathcal{F} \vdash \tau.P + \tau.Q = H$.*

Proof: We have three different cases, according to the form of P and Q.

1. $P \equiv \sum_i \tau.P_i$ and $Q \equiv \sum_j \tau.Q_j$ (we omit the ranges of the indeces to simplify notation). The schema of the proof is like that of Lemma 4.2.6 in [DH84], but an extra step is needed to reduce the resulting term into a form such that Property 3.6 of hnf's holds; in this step the law N3 is used. We only do in detail this part of the proof. By applying the laws D2, D8, N3 and N1 exactly like in [DH84] (just taking care of reading 'a' as 'α'), we can re-write $\tau.P + \tau.Q$ into an equivalent agent S such that:

$$\tau.P + \tau.Q = S \equiv \sum_{A\in L} \tau.S_A$$

where L is a suitable, possibly not saturated, collection of events sets, and for each $A \in L$, $S_A \equiv \sum_{\alpha \in \Lambda(A)} \alpha S_\alpha$, with $\Lambda(A) \subseteq \Lambda$ a suitable set of actions, with the property that $A = \{a \in Ev |$ for some $y, a(y) \in \Lambda(A)$ or $ay \in \Lambda(A)\}$. Now, it is possible that Property 3.6 of hnf's does not hold for such a S, i.e. there exist S_A and S_B such that S_A has a summand $\alpha.S_\alpha$ and S_B has a summand $\beta.S_\beta$, with $\alpha, \beta \in OA$ and $subj(\alpha) = subj(\beta)$, but such that $\alpha.S_\alpha$ is not a summand of S_B. Then, for suitable terms T, S_1, S_2, we can write:

$$\begin{aligned} S &= \tau.S_A + \tau.S_B + T \\ &= \tau.(\alpha.S_\alpha + S_1) + \tau.(\beta.S_\beta + S_2) + T \\ &= \alpha.S_\alpha + \tau.(\alpha.S_\alpha + S_1) + \beta.S_\beta + \tau.(\beta.S_\beta + S_2) + T \quad (D8) \\ &= \tau.(\alpha.S_\alpha + \beta.S_\beta + S_1) + \tau.(\alpha.S_\alpha + \beta.S_\beta + S_2) + T \quad (N3). \end{aligned}$$

By applying systematically this re-writing for each S_A and S_B, we can re-write S into a term S' for which the Property3.6 does hold, i.e., for a suitable normal fuction $g : Ev(L) \longrightarrow \mathcal{P}$,

$$S = \sum_{A \in L} \tau. \sum_{a \in A} g(a) \equiv S'.$$

Now, it remains to see whether L is saturated. If it is not, we can apply to S' the saturation construction of [DH84], based on the laws D5 and D6. The construction apply exactly in our case (only take care of reading '$a.p_a$' as '$g(a)$').

Case 2. (P in α-hnf and Q in τ-hnf) and case 3. (P and Q in α-hnf) easily reduce to case 1. via law D2 (see [DH84]). □

Lemma 6.13 *Let $\{P_i | 0 \leq i \leq k\}$, $k \geq 0$, be a set of hnf's; then there exists a hnf H s.t. $\mathcal{F} \vdash \sum_{0 \leq i \leq k} P_i = H$.*

PROOF: Consider any couple of summands P_i, P_j of $\sum_{0 \leq i \leq k} P_i$; we can reduce $P_i + P_j$ to a hnf in one of the following ways:

- if P_i, P_j are both in τ-hnf, we can apply first law D2 (with $\alpha = \tau$) and then the above lemma to reduce $P_i + P_j$ to a hnf;
- if P_i is in α-hnf and P_j is in τ-hnf, apply laws D4 and D7 to get: $P_i + P_j = \tau.(P_i + P_{jA}) + \tau.P_j$, where P_{jA} is any summand of P_j; by applying law N1, if needed, $P_i + P_{jA}$ is reduced to an α-hnf; applying the previous lemma we get the wanted result;
- if both P_i and P_j are in α-hnf, apply law N1, if needed, to reduce $P_i + P_j$ to a α-hnf.

By repeatedly applying the above reductions, we can re-write $\sum_{0\leq i\leq k} P_i$ into a hnf . □

Proposition 6.14 (Proposition 3.8) *For every P s.t. $P \Downarrow$, there exists a process $h(P)$ in hnf s.t. $\mathcal{F} \vdash P = h(P)$.*

PROOF: (outline) The proof is by structural induction on P, and it is similar to the one for CCS in [DH84] (Proposition 4.2.16), only in the cases of Restriction and Parallel Composition there are some differences. In the former case ($P \equiv (y)P'$), when distributing the restriction over a summation $g(a)$, $a \in OUT$ (applying the law R2), an extra bound output summand may be generated; then we have to apply the law N1 in this way: $\overline{x}(y).P_\alpha + \overline{x}(y).P_\beta = \overline{x}(y).(\tau.P_\alpha + \tau.P_\beta)$, to eliminate the double occurrence of the bound output summands. The Parallel Composition case ($P \equiv P_1|P_2$) is in turn an induction on $dpc(P_1|P_2)$. Both the cases of Parallel Composition and the Summation ($P \equiv P_1 + P_2$) rely on the two previous lemmata. □

Proposition 6.15 (Proposition 3.9) *For every P s.t. $P \Uparrow$, there exists an Ω-hnf $\Omega(P)$ s.t. $\mathcal{F} \vdash P = \Omega(P)$.*

PROOF: An easy structural induction on P, which relies on the law D1, D3 and D10. □

The following lemmata will be used in the proof of the completeness theorem.

Lemma 6.16 *Let P, Q be in hnf and $P \sqsubseteq\!\!\!\!\sim\ Q$. Then:*

i. *for each $\alpha \in Init(P) \cap Init(Q) \cap OA$, $P_\alpha \sqsubseteq\!\!\!\!\sim\ Q_\alpha$, and*

ii. *for each $\alpha = x(y) \in Init(P) \cap Init(Q) \cap IA$ and $z \in fn(P_\alpha, Q_\alpha, y)$, $P_\alpha\{z/y\} \sqsubseteq\!\!\!\!\sim\ Q_\alpha\{z/y\}$.*

PROOF:

i. We first show the may part. Assume $\alpha = \overline{x}y$ and P_α **may** o. Then, taken a fresh variable $z \notin n(y, P, Q, o)$, let $o' \equiv x(z).[z = y]o$. Since $P|o' \stackrel{c}{\Longrightarrow} P_\alpha|[y = y]o \stackrel{\omega}{\Longrightarrow}$, it is P **may** o'. It follows Q **may** o'. But every successful computation for $Q|o'$ must begin like $Q|o \stackrel{c}{\Longrightarrow} Q_\alpha|[y = y]o$, from which it follows that also Q_α **may** o.

Suppose $\alpha = \overline{x}(w)$. We can suppose, by α-renaming, that $w \notin fn(P, Q, o)$. Now, let $o' \equiv x(w)[w \notin Obj(P, Q)]o$ and repeat the previous reasoning: we have that P **may** o', which implies Q **may** o'. But every successful computation from $Q|o'$ must be of the form (up to α-equivalence) $Q|o' \stackrel{c}{\Longrightarrow} (w)(Q_\alpha|[w \notin Obj(P, Q)]o) \stackrel{\omega}{\Longrightarrow}$, because each different computation, having an initial section of the form $Q|o' \stackrel{c}{\Longrightarrow} Q_\alpha|[y \notin Obj(P, Q)]o$, with $\beta \neq \alpha$ and $y \in Obj(Q)$, is unsuccessful. This implies Q_α **may** o. This suffices for the may part.

Now, let us consider the must part. Assume that $\alpha = \overline{x}y$ and P_α **must** o. Take $z \notin n(y, P, Q, o)$ and let $o' \equiv x(z).([z = y]o + [z \neq y]\omega.\mathbf{0}) + \tau.\omega.\mathbf{0}$. It is easily seen that P **must** o' and therefore Q **must** o'. Since $Q|o' \stackrel{c}{\Longrightarrow} Q_\alpha|([y = y]o + [y \neq y]\omega.\mathbf{0})$, it follows that Q_α **must** o. If $\alpha = \overline{x}(w)$ and P_α **must** o, the above reasoning can be repeated with the observer:

$$o' \equiv x(z).([z \notin Obj(P,Q)]o \quad + \sum_{y \in Obj(P,Q)} [z = y]\omega.\mathbf{0}) + \tau.\omega.\mathbf{0}, \ z \text{ fresh.}$$

Thus we have proven part i.

ii. Suppose $\alpha = x(y)$ and let $z \in fn(P_\alpha, Q_\alpha, y)$ and $\sigma = \{z/y\}$. Suppose $P_\alpha\sigma$ **may** o. We have to show that $Q_\alpha\sigma$ **may** o as well. Now, consider $o' \equiv \overline{x}z.o$. We have P **may** o', since $P|o' \stackrel{c}{\Longrightarrow} P_\alpha\sigma|o \stackrel{\omega}{\Longrightarrow}$. Therefore Q **may** o' and this implies in turn that $Q|o' \stackrel{c}{\Longrightarrow} Q_\alpha\sigma|o \stackrel{\omega}{\Longrightarrow}$, from which it follows $Q_\alpha\sigma$ **may** o.

Suppose $P_\alpha\sigma$ **must** o; we have to show that $Q_\alpha\sigma$ **must** o as well. Let $o' \equiv \overline{x}z.o + \tau.\omega.\mathbf{0}$. We have that P **must** o' which implies Q **must** o'. But, since $Q|o' \stackrel{c}{\Longrightarrow} Q_\alpha\sigma|o$, it follows $Q_\alpha\sigma$ **must** o. □

The following definition gives a syntactical relation for 'comparing' hnf's at their top levels. It is a *trait d'union* between the semantical relation $\sqsubseteq^+_\approx$ and the proof-theoretic one $\sqsubseteq$.

Definition 6.17 *Let P and Q be in hnf. Then $P \prec Q$ iff $Init(P) = Init(Q)$ and:*

i. *$P \equiv \sum_{a \in A} g_1(a)$ and $Q \equiv \sum_{a \in A} g_2(a)$, or*

ii. *$P \equiv \sum_{A \in L} \tau.P_A$ and $Q \equiv \sum_{A \in K} \tau.Q_A$, with $K \subseteq L$, or*

iii. *$P \equiv \sum_{A \in L} \tau.P_A$ and $Q \equiv \sum_{a \in A} g_2(a)$, with $A \in L$.*

The relationship between $\sqsubseteq^+_\approx$ and $\prec$ is given by the following:

Lemma 6.18 *Let P and Q be in hnf and $P \sqsubseteq^+_\approx Q$. Then $P \prec Q$.*

PROOF: Since $P \sqsubseteq_{\approx may} Q$, $P \Downarrow$ and $P \sqsubseteq_{\approx must} Q$, from Lemma 6.5 it easily follows that $Init(P) = Init(Q)$ (up to α-equivalence). Let us prove the second condition of Definition 6.17. There are three cases, according to the form of P and Q.

i. P and Q are in α-hnf. Then, since $Init(P) = Init(Q)$, condition i. of the definition holds.

ii. Suppose that P and Q are like in case ii. of the definition. We show that $K \subseteq L$. Now since $Init(P) = Init(Q)$, it is $Ev(K) = Ev(L)$. Since $L = \{A_1, ..., A_n\}$ is saturated, it suffices to show

that: for each $B \in K$ there exists $A_i \in L$ s.t. $A_i \subseteq B$ (this suffices because $A_i \subseteq B \subseteq Ev(K) = Ev(L)$ implies $B \in L$). By contradiction, suppose there exists $B \in K$ s.t. for each $A_i \in L$, $A_i \not\subseteq B$, i.e. for each i, $0 \leq i \leq n$, there exists an $a_i \in A_i - B$. Let $A = \{a_1, ..., a_n\}$ and define for each $a \in A$: $o(a) = \overline{x}y.\omega.\mathbf{0}$ if $a = x \in IN$, $o(a) = x(y).\omega.\mathbf{0}$ otherwise, where y is any name. Also, define $o \equiv \sum_{a \in A} o(a)$. Now, it is clear that P **must** o, but Q may**not** o, due to the unsuccessful computation $Q|o \xrightarrow{\tau} \sum_{a \in B} g(a)|o$. Thus we arrived at a contradiction. Therefore $K \subseteq L$, and the thesis follows.

iii. Suppose P and Q are like in case iii. of the definition. Suppose by contradiction that $A \notin L$. Then we can repeat the above construction with $B = A$. Finally, if P is in α-hnf, then $P \stackrel{\tau}{\not\longrightarrow}$. But being $P \sqsubseteq^{+}_{must} Q$, it is also $Q \stackrel{\tau}{\not\longrightarrow}$, thus Q as well needs to be in α-hnf and we fall in case i. □

Lemma 6.19 *Let P and Q be in hnf and $P \prec Q$. Then $P \sqsubseteq Q$ if for each $\alpha \in Init(P)$:*

i. *$\alpha \in OA$ implies $P_\alpha \sqsubseteq Q_\alpha$, and*

ii. *$\alpha = x(y) \in IA$ implies for each $z \in fn(P_\alpha, Q_\alpha, y)$, $P_\alpha\{z/y\} \sqsubseteq Q_\alpha\{z/y\}$.*

Proof: Since $Init(P) = Init(Q)$, it is $Ev(P) = Ev(Q)$. Let g_1 and g_2 be the functions associated with P and Q, respectively. By applying the rules C0 for $\alpha \in OUT$ and C1 for $\alpha \in IN$, it is straightforward to show that for each $a \in Ev(P) = Ev(Q)$, $g_1(a) \sqsubseteq g_2(a)$. Then we consider the term $R \equiv P[g_2(a)/g_1(a), a \in Ev(P)]$ (where $P[Y/X]$ denotes the agent obtained from P by replacing each (sub)term X in P with Y), and using the rules in C0, we show that $P \sqsubseteq R$. Now the proof that $R \sqsubseteq Q$ proceeds, using τ-laws, exactly like proof of Lemma 4.3.6.i) in [DH84] (just read '$a.q_a$' as 'g_2(a)'.) □

Lemma 6.20 *Suppose P is in Ω-hnf and Q is in hnf or Ω-hnf. If $P \sqsubseteq Q$ then for each $\alpha \in Init(P) \cap Init(Q)$:*

i. *$\alpha \in OA$ implies $P_\alpha + \Omega \sqsubseteq^{+} Q_\alpha + \Omega$, and*

ii. *$\alpha = x(y) \in IA$ implies for each $z \in fn(P_\alpha, Q_\alpha, y), P_\alpha\{z/y\} + \Omega \sqsubseteq^{+} Q_\alpha\{z/y\} + \Omega$.*

Proof: Similar to, but easier than, that of Lemma 6.16. □

Lemma 6.21 *Suppose P is in Ω-hnf and Q is in hnf or Ω-hnf. Then $P \sqsubseteq Q$ if $Init(P) \subseteq Init(Q)$ and for each $\alpha \in Init(P)$:*

i. *$\alpha \in OA$ implies $P_\alpha + \Omega \sqsubseteq Q_\alpha + \Omega$, and*

ii. $\alpha = x(y) \in IA$ *implies for each* $z \in fn(P_\alpha, Q_\alpha, y), P_\alpha\{z/y\} + \Omega \sqsubseteq Q_\alpha\{z/y\} + \Omega$.

PROOF: Similar to, but easier than, that of Lemma 6.19. □

We can finally give the completeness proof:

Theorem 6.22 (Theorem 3.11) $P \sqsubseteq^+_\approx Q$ *implies* $\mathcal{F} \vdash P \sqsubseteq Q$.

PROOF: The proof is by induction on $dpc(P)$.

Basis. $dpc(P) = 0$. If $P \Uparrow$ then $\Omega(P) \equiv \Omega$ exists (Proposition 3.9 and the thesis follows from law Omega. Otherwise, $P \Downarrow$ and therefore $hnf(P)$ exists (Proposition 3.8). Then it is also $Q \Downarrow$ and thus $hnf(Q)$ exists. It must be $hnf(P) \equiv \tau.\mathbf{0}$ or $hnf(P) \equiv \mathbf{0}$ and, since $Init(hnf(P)) = Init(hnf(Q))$, it is the same for $hnf(Q)$. Now, if $hnf(P) \equiv \tau.\mathbf{0}$ and $hnf(Q) \equiv \mathbf{0}$, the thesis follows from N4. Otherwise it must be $hnf(P) \equiv hnf(Q) \equiv \mathbf{0}$.

Inductive Step. $dpc(P) > 0$. If $P \Uparrow$, $\Omega(P)$ exists and Q has an (Ω-)hnf, say H. Since $\Omega(P) \sqsubseteq_{may} H$, from Lemma 6.5 we get $Init(\Omega(P)) \subseteq Init(H)$. By applying Lemma 6.20, we have that for each $\alpha \in Init(\Omega(P))$:

i. $\Omega(P)_\alpha + \Omega \sqsubseteq^+_\approx H_\alpha + \Omega$, for $\alpha \in OA$;

ii. $\Omega(P)_\alpha\sigma + \Omega \sqsubseteq^+_\approx H_\alpha\sigma + \Omega$, for each $\alpha = x(y) \in IA$ and each $\sigma = \{z/y\}$ with $z \in fn(\Omega(P)_\alpha, H_\alpha, y)$.

Since for each considered σ: $dpc(\Omega(P)_\alpha\sigma) < dpc(\Omega(P)) = dpc(P)$, we can apply the inductive hypothesis to get that the above relations i. and ii. still hold when replacing $\sqsubseteq^+_\approx$ with $\sqsubseteq$. By applying Lemma 6.21, we obtain the claim. If $P \Downarrow$ then also $Q \Downarrow$ and $hnf(P) \equiv H_1$ and $hnf(Q) \equiv H_2$ both exist. Moreover, by possibly applying law D7 ($\alpha.X = \alpha.\tau.X$), we can suppose H_1 is in a form s.t. for each $\alpha \in Init(H_1)$, $H_{1\alpha}$ is of the form $\tau.H'$, for some H'. From Lemma 6.18, we get $H_1 \prec H_2$. Moreover, from Lemma 6.16, we obtain that for each $\alpha \in Init(H_1) = Init(H_2)$:

a. $H_{1\alpha} \sqsubseteq^+_\approx H_{2\alpha}$, $\alpha \in OA$;

b. $H_{1\alpha}\sigma \sqsubseteq^+_\approx H_{2\alpha}\sigma$, for $\alpha = x(y) \in IA$ and for each $\sigma = \{z/y\}$ with $z \in fn(H_{1\alpha}, H_{2\alpha}, y)$.

(Recall that $H_{1\alpha} \xrightarrow{\tau}$) By applying the inductive hypothesis, we obtain that relations a. and b. still hold when replacing $\sqsubseteq^+_\approx$ with $\sqsubseteq$. By applying Lemma 6.19 we then obtain the thesis. □

We now prove a proposition about a variant of the testing theory not using the mismatch operator. The significance of this result is discussed in Subsection 2.3.

Proposition 6.23 (Proposition 2.1) *Let be x, y and w be any three names. Then:*

a. $\overline{x}(w).\mathbf{0} \sqsubseteq^{-}_{may} \overline{x}.y.\mathbf{0}$;

b. $\overline{x}(w).\Omega \sqsubseteq^{-}_{must} \overline{x}(w).\mathbf{0} + \overline{x}y.\Omega$.

PROOF: To prove this proposition we need a stronger version of Lemma 6.1. More precisely, we need to discard the hypothesis on the injectivity of σ in item 3 of the lemma. This is possible because, when mismatch is not considered, we can directly apply Lemma 3 of [MPW89]. Formally, we have:

a. Given a mismatch-free observer o, suppose P **may** o. We can assume, w.l.o.g., that $w \notin fn(o)$; then it must be, for some o', that $P|o \Longrightarrow P|o' \xrightarrow{c} (w)(\mathbf{0}|o_1)$, where $o_1 \overset{\omega}{\Longrightarrow}$ and $o \xrightarrow{x(w)} o_1$. On the other side, we have $Q|o \Longrightarrow Q|o' \xrightarrow{c} \mathbf{0}|(o_1\{y/w\})$. Then, by (repeatedly) applying Lemma 6.1.3 with the hypothesis σ *injective* discarded[4] it is easy to see that also $o_1\{y/w\} \overset{\omega}{\Longrightarrow}$ and thus also Q **may** o, and the thesis follows.

b. Let $P \equiv \overline{x}(w).\Omega$ and $Q \equiv \overline{x}(w).\mathbf{0} + \overline{x}y.\Omega$. Suppose, for any mismatch-free observer o, that Q **must** o. We can assume w.l.o.g. that $w \neq y$ and that $w \notin fn(o)$. We shall consider the possible cases of unsuccessful computation for $Q \mid o$ and in each case exhibit a unsuccesful computation for $P|o$ as well.

The non-trivial cases are those in which a communication between Q and the observer occurs before the observer itself reaches a divergent state, i.e. those computations having an initial segment of the kind:

$$Q|o \Longrightarrow Q \mid o_k \xrightarrow{c} [Q' \mid o_{k+1}]$$

where $o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \ldots \xrightarrow{\tau} o_k$ with $o_i \not\xrightarrow{\omega}$ for $0 \leq i \leq k+1$ and Q' **must** o_{k+1}. Let us analyze the possible cases for the communication $Q \mid o_k \xrightarrow{c} [Q' \mid o_{k+1}]$. There are two possibilities:

1. Q synchronizes via the bound output $\overline{x}(w)$. In this case, also $P|o$ has the failing computation $P|o \Longrightarrow P|o_k \xrightarrow{c} (w)(\Omega|o_{k+1})$.

2. Q synchronizes via the free output $\overline{x}y$. It is therefore $o_k \xrightarrow{x(z)} o'$ for some z and o' such that $o_{k+1} \equiv o'\{y/z\}$. Therefore, $P|o$ has the failing computation with the initial segment:

$$P|o \Longrightarrow P|o_k \xrightarrow{c} (w)(\Omega|o'\{w/z\})\,.$$

[4]This is essential, because we do not actually know whther $y \notin fn(o_1)$, i.e. whether σ is injective on o_1.

To prove that this yields a failing computation, it suffices to show that $o'\{w/z\} \stackrel{\omega}{\not\longrightarrow}$. Suppose the contrary, i.e. $o'\{w/z\} \stackrel{\omega}{\longrightarrow}$. Since $w \notin fn(o_{k+1})$ (this is proven by repeatdly applying Lemma 6.1.1), it is $o_{k+1} \equiv o'\{y/z\} \equiv_\alpha (o'\{w/z\})\{y/w\}$. In virtue of Lemma 6.1.3. with the hypothesis σ injective discarded, it would hence be also $o_{k+1} \stackrel{\omega}{\longrightarrow}$; but this contradicts Q' **must** o_{k+1}.

□

6.5 Denotational Semantics

Definition 6.24 (Finite-Breadth Approximants) *For each P, define the set $FBA(P) \subseteq FSNF$ inductively as follows:*

- $FBA(op(P_1, ..., P_k)) = \{snf(op(N_1, ..., N_k)) \mid N_i \in FBA(P_i),\ \textit{for } i = 1, ..., k\}$
  for each k-ary operator op different from input prefix, $k = 0, 1$ or 2.
- $FBA(x(y).P) = \{x(w).(\sum_{z \in F}[w = z]N_z + [w \notin F]\Omega) \mid F \subseteq_{fin} \mathcal{N},\ N_z \in FBA(P\{z/w\})$
  and w fresh $\}$.

Notice that, by construction, $N \in FBA(P)$ implies $N \sqsubseteq_{must} P$.

Proposition 6.25 *Let P and Q be processes. Then:*

a. *$N \in FBA(P)$ implies $[\![N]\!] = [N]$;*

b. *$[\![FBA(P)]\!]$ is directed in $\mathcal{D}$ and $[\![P]\!] = sup[\![FBA(P)]\!]$;.*

c. *$N \in FSNF$ and $N \sqsubseteq_{must} P$ implies there exists $M \in FBA(P)$ such that $N \sqsubseteq_{must} M$;*

d. *If for each $N \in FBA(P)$ $N \sqsubseteq_{must} Q$ then $P \sqsubseteq_{must} Q$.*

Proof: (outline) In the proof we use the notion of *finite-breadth term*, a term in which every input prefix $x(w).$ is followed by a subterm of the form $(\sum_{z \in F}[w = z]P_z + [w \notin F]\Omega)$, with $w \notin F$. Fsnf's are of course particular cases of finite-breadth terms.

a. Show, by structural induction, that for each finite-breadth term P, $snf(P) \in FSNF$ and $[\![P]\!] = [snf(P)]$.

b. The proof is a structural induction on P, which exploits the continuity of the functions f_{op}, out and inp. The only difficult case is when $P \equiv x(y).R$ for some process R. We consider only this case. Define for each z, $R_z = R\{z/y\}$; we have:

$$\begin{aligned}
[\![FBA(P)]\!] &= \{[x(w).(\textstyle\sum_{z\in F}[w=z]N_z + [y \notin F]\Omega)] \mid F \subseteq_{fin} \mathcal{N},\ N_z \in FBA(R_z), w \text{ fresh } \} \\
&\quad \text{(Definition of } FBA \text{ and part a.)} \\
&= \{inp(x,f) \mid f \in K\} \ \text{(Definition of } inp\text{)}
\end{aligned}$$

where K is the set of functions $\{f : \mathcal{N} \longrightarrow PO \mid f$ is of the kind: $\lambda z.\ if\ z \in F\ then\ [N_z]\ else\ [\Omega]$, with $F \subseteq_{fin} \mathcal{N}$ and $N_z \in FBA(R_z)\}$. Now it is easy to check that K is directed in $(\mathcal{N} \longrightarrow \mathcal{D})$: this is an easy consequence of the fact that for each z, by inductive hypothesis $[\![FBA(R_z)]\!]$ is directed in $\mathcal{D}$. Therefore, due to continuity of $inp(x,-)$, we have:

$$\begin{aligned}
sup[\![FBA(P)]\!] &= sup\{inp(x,f) \mid f \in K\} \\
&= inp(x, sup\, K).
\end{aligned}$$

On the other side:

$$\begin{aligned}
[\![P]\!] &= inp(x, \lambda z.[\![R_z]\!]) \\
&= inp(x, \lambda z.sup\,[\![FBA(R_z)]\!]) \quad \text{(inductive hypothesis)} \\
&= inp(x,\ sup\, H)
\end{aligned}$$

where $H = \{f \mid f$ is a function of the form: $\lambda z.[N_z]$ with $N_z \in FBA(R_z)\}$. We have $K \subseteq H$ and thus $sup\, K \preceq sup\, H$. On the other side, it is easily seen that each function of H is approximable by a chain of functions in K. In fact, given any $f = \lambda z.[N_z]$ in H, consider the chain $C_f = \{f_i \mid i \geq 0\} \subseteq K$, with the functions f_i, $i \geq 0$, defined in this way: fixed any infinite chain of finite sets $F_0 \subseteq F_1 \subseteq F_2 \subseteq ...$ s.t. $\cup_{i\geq 0} F_i = \mathcal{N}$, define $f_i = \lambda z.if\ \ z \in F_i\ then\ [N_z]\ else\ [\Omega]$; it is easy to check that $sup\, C_f = f$. Now apply the following fact from cpo theory: given $X, Y \subseteq D$ cpo, with sup's s_x, s_y respectively, if for each $y \in Y$ there exists $A_y \subseteq X$ s.t. $sup\, A_y = y$, then $s_y \preceq s_x$. We obtain that $sup\, H \preceq sup\, K$, and therefore $sup\, H = sup\, K$; therefore $inp(x, sup\, K) = inp(x, sup\, H)$ and the thesis follows.

c. It is sufficient to prove that for any finite-breadth term P and any process Q it holds that $P \sqsubseteq_{must} Q$ implies there exists $M \in FBA(P)$ such that $P \sqsubseteq_{must} M$. This is proved by induction on the depth of the formal proof of $P \sqsubseteq_{must} Q$ within the proof system for $\sqsubseteq_{must}$. The relevant case is when C1 is the last applied rule. In that case, techniques similar to the ones used in the above item b. can be employed.

d. The claim easily follows from Lemma 5.1 and from the following assertion: for any process P and any restriction-free observer o, if P **must** o then there exists $N \in FBA(P)$ such that N **must** o. Thus, it is sufficient to prove the latter assertion. Indeed, it holds that for any process P and any restriction-free observer o, if P **must** o then there exists $M \in FSNF$ such that M **must** o (this can be proved by specializing the proof for the infinite case of Proposition 6.29; only notice that, for this finite case, the proof of the latter can be re-written without mentioning any recursion law). Now, since $M \sqsubseteq_{must} P$, from part c. of this proposition we get that there exists $N \in FBA(P)$, $M \sqsubseteq_{must} P$, s.t. $M \sqsubseteq_{must} N$; therefore N **must** o, and the assertion is proven.

□

Now we can prove:

Theorem 6.26 (Full abstraction for finite processes) *For each P and Q, $P \sqsubseteq_{must} Q$ iff $[\![P]\!] \preceq [\![Q]\!]$.*

Proof:

(If) We first prove that given any $N \in FSNF$, $[\![N]\!] \preceq [\![Q]\!]$ implies $N \sqsubseteq_{must} Q$. In fact, $[\![FBA(Q)]\!]$ is directed in $\mathcal{D}$ and $[\![N]\!]$ is compact in $\mathcal{D}$; hence, due to Proposition 6.25, part b., there exists $N' \in FBA(Q)$ s.t. $[\![N]\!] \preceq [\![N']\!]$, i.e. $N \sqsubseteq_{must} N'$ for part a. of Proposition 6.25. But $N' \sqsubseteq_{must} Q$, and thus $N \sqsubseteq_{must} Q$. Now, $[\![P]\!] \preceq [\![Q]\!]$ implies, for the previous proposition, part b., that for each $N \in FBA(P)$ it is $[\![N]\!] \preceq [\![Q]\!]$ and therefore $N \sqsubseteq_{must} Q$. Applying part d. of Proposition 6.25, we obtain then $P \sqsubseteq_{must} Q$.

(Only If) In virtue of Proposition 6.25, part c., for each $N \in FBA(P)$, there exists a $N' \in FBA(Q)$ s.t. $N \sqsubseteq_{must} N'$, i.e. $[\![N]\!] \preceq [\![N']\!]$ in $\mathcal{D}$, for part a. of the same proposition. From this fact it follows that $sup\,[\![FBA(P)]\!] \preceq sup\,[\![FBA(Q)]\!]$, i.e. $[\![P]\!] \preceq [\![Q]\!]$ in $\mathcal{D}$, for part b. of Proposition 6.7. □

6.6 Dealing with Infinite Agents

All proofs given in the soundness subsection for the finite calculus easily extend to arbitrary agents via the following lemma, that generalizes the assertions after Definition 6.3 to the infinite agents. Essentially, it says that, even in the case of infinite agents, if two processes always generate *finite* equivalent sequences of silent moves, when run in parallel with the same observer, then they are must-equivalent.

Lemma 6.27 *Given P, P' processes and o, o' observers, suppose $P|o \leq P'|o'$. Then P* **must** *o implies P'* **must** *o'.*

PROOF: Consider the possible kinds of failing computation for $P|o$. By definition of $\leq$ between experiments, if a finite failing computation exists for $P|o$, then it must also exist for $P'|o'$.

The only remaining case is therefore $P|o = e_0 \xrightarrow{\tau} e_1 \xrightarrow{\tau} e_2 \xrightarrow{\tau} \cdots$ and $e_i \not\xrightarrow{\omega}$ for $i \geq 0$. Suppose by contradiction that P' **must** o'. Consider the *computation tree* from $P'|o'$ (with τ-actions only) where α-equivalent direct descendants of each node are identified, and with branches pruned to obtain a tree whose leaves are all those nodes that can give rise to $\xrightarrow{\omega}$ transitions; call this tree $\mathcal{T}_{\equiv_\alpha \omega}$. Since P' **must** o', $\mathcal{T}$ has the property that for each non-leaf node e, $e \downarrow$. It is not difficult to show that if $e \downarrow$ then e has a finitely many (up to α-equivalence) τ-derivatives; therefore $\mathcal{T}$ is finitary. By definition of $\leq$ between experiments, for each $k \geq 0$, there exists in $\mathcal{T}$ a path $(P'|o' = f_0, f_1, ..., f_k)$ with the property that $(e_0 \xrightarrow{\tau} ... \xrightarrow{\tau} e_k)$ and $(f_0 \xrightarrow{\tau} ... \xrightarrow{\tau} f_k)$ are equivalent sequences of τ-transitions. By applying Koenig's lemma, there exists in $\mathcal{T}$ an infinite path; but this contradicts P' **must** o'. □

The following lemma, that provides a different characterization of the must preorder, is necessary for proving the next proposition.

Lemma 6.28 (Lemma 5.1) *The must testing preorder $<_M$ over $\mathcal{RP}$ obtained by taking the set of observers without restriction coincides with $\sqsubseteq_{must}$.*

PROOF: Obviously $P \sqsubseteq_{must} Q$ implies $P <_M Q$.

To show the reverse implication, suppose that $P <_M Q$ and P **must** o, where o is any observer.

By possibly α-renaming o, we can suppose w.l.o.g. that each restriction (y) in o is such that $y \notin fn(P, Q, o)$. Fixed a restriction (y) of o, consider now the observer $r(o, y)$ where, intuitively, all restrictions (y) have been discarded; more precisely $r(o, y)$ is defined inductively in this way: $r(op(P_1, ..., P_k), y) = op(r(P_1, y), ..., r(P_k, y))$, for any k-ary operator op, $k = 0, 1$ or 2, different from (y), $r((y)o', y) = r(o', y)$. It can be easily proved, by structural induction on o and by exploiting Lemma 6.4 in case $o = (y)o'$, that $R|o$ and $R|r(o, y)$ are equivalent as experiments, where R is P or Q. By repeatedly applying this result, we can eventually get rid of all the restrictions of o, i.e. obtain a restriction-free observer o', such that $R|o$ and $R|o'$ are equivalent as experiments, where R is P or Q.

Thus P **must** o'; but being $P <_M Q$, we have Q **must** o', and therefore, since $Q|o'$ is equivalent to $Q|o$, also Q **must** o. □

Proposition 6.29 (Proposition 5.2) *For any process P and for any restriction-free observer o, if P* **must** *o then there exists $D \in FIN(P)$ such that D* **must** *o.*

PROOF: In this proof "$X \sqsubseteq_{must} Y$" will be used as a shorthand for "$\mathcal{A} - \{\omega - induction, REC2\} \vdash X \sqsubseteq_{must} Y$". It is a routine task of algebraic semantics (see [Hen88]) to show that, for any finite D and any P, if $D \sqsubseteq_{must} P$, then there exists $D' \in FIN(P)$ such that $D \sqsubseteq_{must} D'$. Thus, to prove our claim, it suffices to show there exists a finite D s.t. $D \sqsubseteq_{must} P$ and D **must** o. Actually, we will show a little stronger result, i.e.: if P **must** o then there exists a finite strong normal form N s.t. $N \sqsubseteq_{must} P$ and N **must** o. Note that, by hypothesis, o cannot perform any bound output action.

Suppose now that P **must** o and consider the computation tree $\mathcal{T}$ from $P|o$, defined like in the proof of Lemma 6.27. Since P **must** o and $\mathcal{T}$ is finitary, it follows from Koenig's lemma that $\mathcal{T}$ is finite. The proof is by induction on the *depth of communication of* $\mathcal{T}$, i.e. the maximal number of communications between the process and the observer in a path from the root to a leaf in $\mathcal{T}_{\equiv_\alpha \omega}$. It is not hard to show that this number does not change if we consider the computation tree generated by a $P'|o$, for any P' must-equivalent to P.

If $P \Uparrow$, then it must be $o \xrightarrow{\omega}$, and we can take $N \equiv \Omega$. If $P \Downarrow$, then we can suppose that P is in hnf. There are two cases.

1. $P \equiv \sum_{\alpha \in A} \alpha.P_\alpha$. Agent N is built up out of the finite agents N_α, $\alpha \in A$, defined as follows:

   - $\alpha = x(y)$. Consider the set:

     $$Pairs(\alpha) = \{(z, o') |\, o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} o_k \xrightarrow{\overline{x}z} o',\, o_i \not\xrightarrow{\omega} \text{ for } 0 \leq i \leq k\}\,.$$

     For each $(z, o') \in Pairs(\alpha)$, we have $P|o \stackrel{c}{\Longrightarrow} P_\alpha\{z/y\}|o'$, thus $P_\alpha\{z/y\}$ **must** o'; therefore, for each z such that some pair (z, o') is in $Pairs(\alpha)$, $P_\alpha\{z/y\}$ **must** $\sum_{(z,o') \in Pairs(\alpha)} \tau.o'$ and, by induction hypothesis, there exists a finite strong normal form $N(\alpha, z) \sqsubseteq_{must} P_\alpha\{z/y\}$ such that $N(\alpha, z)$ **must** $\sum_{(z,o') \in Pairs(\alpha)} \tau.o'$, i.e $N(\alpha, z)$ **must** o', for each $(z, o') \in Pairs(\alpha)$. Note that $N(\alpha, z)$ does not depend on y, thus, by possibly α-renaming y in P, we can suppose $y \notin \Pi_1(Pairs(\alpha))$ and $y \notin fn(N(\alpha, z))$ (Π_1 is the projection on the first coordinate). Finally, define N_α to be

     $$\sum_{z \in \Pi_1(Pairs(\alpha))} [y = z]N(\alpha, z) + [y \notin \Pi_1(Pairs(\alpha))]\Omega\,.$$

     By applying Match and Mismatch laws it is easy to check that for each name w, $N_\alpha\{w/y\} \sqsubseteq_{must} P_\alpha\{w/y\}$.

- $\alpha = \overline{x}y$. If the set

$$S = \{o' |\, o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} o_k \text{ and } P|o_k \xrightarrow{c} P_\alpha|o',\, o_i \not\xrightarrow{\omega} \text{ for } 0 \leq i \leq k\}$$

  is not empty, then P_α **must** $\sum_{o' \in S} \tau.o'$; by induction hypothesis, there exists a fsnf $N' \sqsubseteq_{must} P_\alpha$ with the property that for each $o' \in S$, N'_α **must** o'; take N_α to be N'_α. If $S = \emptyset$, take N_α to be Ω.

- $\alpha = \overline{x}(y)$. We can suppose, by possibly α-renaming, that $y \notin fn(o)$. If the set

$$S = \{o' |\, o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} o_k \text{ and } P|o_k \xrightarrow{c} (y)(P_\alpha|o'),\, o_i \not\xrightarrow{\omega} \text{ for } 0 \leq i \leq k\}$$

  is not empty, then P_α **must** $\sum_{o' \in S} \tau.o'$, and, by induction hypothesis, there exists a fsnf $N' \sqsubseteq_{must} P_\alpha$ with the property that for each $o' \in S$, N'_α **must** o'; take N_α to be N'_α. If $S = \emptyset$, take N_α to be Ω.

Now define N to be the fsnf $\sum_{\alpha \in A} \alpha.N_\alpha$. By using the rules C0 and C1, we see that $N \sqsubseteq_{must} P$ and furthermore by construction N **must** o.

2. $P \equiv \sum_{A \in L} \tau.\sum_{\alpha \in A} \alpha.P_\alpha$. For each $A \in L$, $\sum_{\alpha \in A} \alpha.P_\alpha$ **must** o; repeat the above construction, thus there exists a fsnf N_A s.t. $N_A \sqsubseteq_{must} \sum_{\alpha \in A} \alpha.P_\alpha$ and N_A **must** o; then take N to be $\sum_{A \in L} \tau.N_A$.

□

Theorem 6.30 (Theorem 5.1) $[\![P]\!] \preceq [\![Q]\!]$ *in* $\mathcal{D}$ *if and only if* $P \sqsubseteq^+_{must} Q$.

PROOF: Recall from [HP80] that:

$$\{[\![D]\!]\rho |\, D \in FIN(P)\} \text{ is directed in } \mathcal{D} \text{ and } [\![P]\!]\rho = sup\,\{[\![D]\!]\rho |\, D \in FIN(P)\}\,. \tag{1}$$

Define for each closed non-finite P, $FBA(P) = \{N |\, N \in FBA(D) \text{ for some } D \in FIN(P)\}$. From Proposition 6.25, Theorem 6.26 and 1 above, it easily follows that:

$$[\![FBA(P)]\!] = \{[\![N]\!] | N \in FBA(P)\} \text{ is directed in } \mathcal{D} \text{ and } [\![P]\!] = sup\,\{[\![N]\!] | N \in FBA(P)\}\,. \tag{2}$$

On the other side, it is easy to check that for each closed P and Q, we have $P \sqsubseteq_{must} Q$ if and only if for each $D \in FIN(P)$ there exists $E \in FIN(Q)$ such that $D \sqsubseteq_{must} E$ (to prove the 'if' part, exploit $\omega - induction$ rule; for the 'only if' part, see the proof of Proposition 6.29). From Proposition 6.25, it therefore follows that:

$$P \sqsubseteq_{must} Q \text{ iff for each } N \in FBA(P) \text{ there exists a } M \in FBA(Q) \text{ s.t. } N \sqsubseteq_{must} M\,. \tag{3}$$

Recalling that $[\![FBA(P)]\!]$ is a set of compact elements in $\mathcal{D}$, it follows from (2): $[\![P]\!] \preceq Q$ in $\mathcal{D}$ if and only if for each $N \in FBA(P)$ there exists an $M \in FBA(Q)$ such that $[\![N]\!] \preceq [\![M]\!]$ in $\mathcal{D}$. Since it has already been proved for finite agents N and M that $[\![N]\!] \preceq [\![M]\!]$ in $\mathcal{D}$ if and only if $N \sqsubseteq_{must} M$, we have from (3) that $[\![P]\!] \preceq [\![Q]\!]$ in $\mathcal{D}$ iff $P \sqsubseteq_{must} Q$, and the thesis follows from the soundness and completeness of $\sqsubseteq_{must}$ w.r.t. $\sqsubseteq^{+}_{\approx_{must}}$. □

7 Conclusions

We have equipped the π-calculus with a complete proof system and a fully abstract denotational model w.r.t. testing-based behavioural relations. Besides this, we have assessed the importance of the mismatch operator from an observational point of view and the stability of the testing approach w.r.t. different (early, late) underlying transition systems. The paper has been concerned with a particular presentation of the π-calculus, that of [MPW89]; however, we think that the constructions and the results presented here easily generalize to other variants of -calculus, such as its polyadic version (that permits exchanging tuples of names and describes infinite behaviours by means of a replicator operator) [Mil91].

It must be said that the construction of our denotational model is somehow syntax-dependent and thus that it is not completely satisfactory. We would have preferred a more abstract presentation of the cpo, for example in terms of a domain equation; an attempt has been made of using a variant of *generalized acceptance trees* as in [HI91], but we could not directly describe the restriction operator as a binder in such a model.

Besides [MPW89], this paper relates mainly to [Hen91] and, in some respects, to [PS93]. In [Hen91], independently, a testing-based axiomatization and a fully abstract denotational model are worked out for a language slightly different from ours: Internal moves are replaced by a binary internal choice operator and the match- mismatch pair is replaced by an *if-then-else* construct. The actual completeness proof relies on the existence of head normal forms similar to ours. The different selection of operators does not permit any assessment of the widely studied originalπ-calculus with respect to the testing approach. It also prevents a precise comparison between the work described in this paper and Hennessy's. It can however be said that the main differences between the two axiomatizations are essentially the same as those between the axiomatization of testing for CCS [DH84] and that for TCCS [DH87]. The comparison becomes more significative when considering the two denotational models: in [Hen91] no notion of normal form is introduced and the compact

elements of the domain are defined as (the equivalence classes of) those finite terms which have a "finite behaviour" on input actions. As a consequence a term-model is obtained.

In [PS93], four strong bisimulation-based equivalences over the π-calculus are axiomatized, namely early and late ground bisimulations (which we considered in Section 2.4) and the congruences they induce. The latter can be defined as the closures under substitutions of the former. The axiomatizations of the ground equivalences contain an input prefix inference rule similar to our C1 and to that used in Hennessy's work. In the proof systems for congruences, the input prefix rule is replaced by a simpler one that exploits the fact that congruences are fully preserved by substitutions. All axiomatizations of [PS93], but that of late ground equivalence, heavily rely on the presence of the mismatch operator. The authors pose the interesting problem of formally establishing the necessity of this operator for axiomatizing bisimulation-based equivalences. In this paper we have seen that mismatch has a fundamental role in the definition of testing preorders; its omission would lead to a theory that does not correspond to any operational intuition.

We have not tackled the problem of the input prefix congruence rule. The presence of this rule in a proof system leads in general to inefficient derivations, in that multiple checks are required whenever a (sub-)proof of the form $a(x).P \sqsubseteq a(x).Q$ is needed. A possible solution to this problem would consist in devising a proof system with *guarded* statements of the form $\phi \vdash a(x).P \sqsubseteq a(x).Q$, where ϕ is a boolean condition over names; this would permit making assumptions over names explicit via ϕ and avoiding case analysis on input by inspecting ϕ. For bisimulation-based equivalences, a similar approach has been developed in [HL93] for a static value-passing language and in [BD94] for the π-calculus; it would be interesting to extend the approach to the theory of testing.

References

[BD94] M. Boreale and R. De Nicola. A symbolic semantics for the π-calculus. Technical report, Dipartimento di Scienze dell'Informazione Università "La Sapienza", Roma, 1994. Extended Abstract appeared in *Proceedings of Concur '94*, LCNS 836, Jonsson, B. and Parrow, J. eds.

[BK89] J.A. Bergstra and J.-W. Klop. Process theory based on bisimulation semantics. In *Linear Time, Branching Time and Partial Orders in Logic and Models for Concurrency, LNCS, 354*, pages 50–122. Springer-Verlag, Berlin, 1989.

[De 87] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

[DH84] R. De Nicola and M. Hennessy. Testing equivalence for processes. *Theoret. Comput. Sci.*, 34:83–133, 1984.

[DH87] R. De Nicola and M. Hennessy. Ccs without τ's. In *Proceedings of Tapsoft 87, LNCS 249*, pages 138–152, 1987.

[DIN90] R. De Nicola, P. Inverardi, and M. Nesi. Using axiomatic presentation of behavioural equivalences for manipulating ccs specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Machines, LNCS 407*, pages 54–67, 1990.

[EN86] U. Engberg and M. Nielsen. A calculus of communicating systems with label passing. Technical Report DAIMI-PB 208, Computer Science Dept., Aarhus University, 1986.

[Gue81] I. Guessarian. *Algebraic Semantics*. LNCS, 99. Springer-Verlag, Berlin, 1981.

[Hen88] M. Hennessy. *An Algebraic Theory of Processes*. MIT Press, Cambridge, 1988.

[Hen91] M. Hennessy. A model for the π-calculus. Technical report, University of Sussex, 1991.

[HI91] M. Hennessy and A. Ingolfsdottir. A theory of testing equivalence with value-passing. *Information and Computation*, 1991. (to appear).

[HL93] M. Hennessy and H. Lin. Proof systems for message-passing process algebras. In E. Best, editor, *Proceedings of CONCUR '93, LNCS 715*. Springer-Verlag, Berlin, 1993.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall Int., London, 1985.

[HP80] M. Hennessy and G. Plotkin. A term model for ccs. *LNCS*, 88, 1980.

[Hui92] L. Huimin. Pam: A process algebra manipulator. In K.G. Larsen and A. Skou, editors, *Computer Aided Verification, LNCS 575*, 1992.

[Ing93] A. Ingolfsdottir. Late and early semantics coincide for testing. Technical Report IR93-2008, Departement of Mathematics and Computer Science , University of Aalborg, 1993.

[Mil80] R. Milner. *A Calculus of Communicating Systems*. LNCS, 92. Springer-Verlag, Berlin, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[Mil91] R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, LFCS, Dept. of Comp. Sci., Edinburgh Univ., 1991.

[MPW89] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part i and ii. Technical report, LFCS, Department of Computer Science, University of Edinburgh, 1989. Published in Information and Computation, Vol. 100, pp. 1-41 and 42-78, 1992.

[MPW91] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. In *Proc. Concur '91, LNCS 527.* Springer-Verlag, 1991.

[Plo81] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[PS93] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. Technical report, University of Edinburgh, 1993. To appear in *Information and Computation.*

[San93] D. Sangiorgi. A theory of bisimulation for the π-calculus. In E. Best, editor, *Proceedings of CONCUR '93, LNCS 715.* Springer-Verlag, Berlin, 1993.

[Tho89] B Thomsen. A calculus of higher order processes. In *Proc. of POPL 89, ACM*, 1989.