A Fully Abstract Model for Higher-Order Mobile Ambients

M. Coppo* M. Dezani-Ciancaglini**

Dipartimento di Informatica – Università di Torino
{coppo,dezani}@di.unito.it

Abstract. Aim of this paper is to develop a filter model for a calculus with mobility and higher-order value passing. We will define it for an extension of the Ambient Calculus in which processes can be passed as values. This model turns out to be fully abstract with respect to the notion of contextual equivalence where the observables are ambients at top level.

1 Introduction

The *Ambient Calculus* [8] is a calculus of mobile computation that allows active processes to move between sites and interact with them. Owing to its interest a number of studies on various foundational aspects of this or derived systems have been recently developed. The subject of these investigations have been mainly type systems (finalized to the proof of various properties like safe communications [9] or security [7, 17, 6]), proof systems [10], abstract interpretations [19] and flow analysis [13]. In [11] a denotational model has been proposed for a very basic subset of the language, in which only mobility primitives where present. Aim of this paper is to extend this model to a language in which processes can be exchange as values inside ambients. Our approach is similar to that of [3], but in our language we do not have a separate set of expressions (including an explicit λ operator) which can be communicated. In the resulting language, however, the λ-calculus can be directly simulated.

One main difficulty in defining models of the Ambient Calculus is that of finding an abstract counterpart to the notion of mobility. A promising tool for overcoming this difficulty seems the notion of "logical" semantics in which domains are described by abstract filters of logical formulas expressing properties of the terms of the calculus. In filter models, moreover, terms are interpreted as the sets of their computational properties (types). This makes them an interesting basis for the study and development of analysis tools. Filter models have been successfully applied, for instance, to the study of normalization properties of lambda terms [16].

In this paper we define, in particular, a model for a variant of the Ambient Calculus with synchronous higher-order value passing and a new "selfopen" primitive so. The so action is strongly related to the *acid* operation of [8] and allows the simulation of

* Partially supported by MURST Cofin'00 AITCFA Project, and IST-2001-33477 DART Project.
** Partially supported by MURST Cofin'01 COMETA Project, and IST-2001-322222 MIKADO Project.

objective moves [8]. Our model turns out to be fully abstract with respect to the notion of contextual equivalence defined in [18]. The same model is also adequate with respect to the calculus obtained by eliminating the so action, but in this case it is not fully abstract.

The "logical" approach to denotational semantics goes back to [25], and has been advocated in [1] as a general paradigm unifying, among other things, type assignment, logic of programs and logical characterizations of behaviors of concurrent processes such as Hennessy-Milner logic. A main advantage of this approach is that it produces a denotational model in the sense that the denotation of a term is given in a compositional way. This has been used in [4] and [2] for λ-calculus. In the same line are the studies concerning extensions of λ-calculus by means of operators with concurrent features like [23, 5, 14, 15]. In [5], in particular, the intersection type operator is seen as the basic tool to represent nondeterministic choice in the "may" perspective.

In [21] Hennessy presents the first denotational model of higher-order concurrent processes based on a compromise between type systems and modal logic. The resulting filter model turns out to be fully abstract with respect to an operational semantics based on a notion of testing and "may" convergency. A similar result has been proved in [20] for a kernel of the language FACILE. A filter model for higher-order processes which is adequate but not complete with respect to the "must" testing as been proposed in [22]. The same approach is used in [12] to build a filter model of the π-calculus which is fully abstract for "may" convergence.

The type system used for the definition of the model can also be seen as a proof system to express ambient and process properties. Proof systems with these aims have also proposed by Cardelli and Gordon (see e.g. [10]). Sangiorgi's paper [24] provides a careful study of the equivalence on mobile ambients induced by the *ambient logic* of [10]. The logic turns out to be strongly intensional, equating mainly structurally equivalent processes: for example it distinguishes between the processes $in\, a.in\, a.\mathbf{0}$ and $in\, a.\mathbf{0} \mid in\, a.\mathbf{0}$, which are observationally equivalent. Therefore the ambient logic seems not suitable to be taken as a basis for the construction of a model of contextual semantics, where properties need to have an extensional meaning.

2 The Language

The calculus of Selfopening Mobile Ambients, introduced in [11], is an extension of the calculus of Mobile Ambients [8] obtained by adding a new primitive action so. In the Ambient Calculus a process has no way of opening the ambient a in which he is running raising itself one level up. This kind of action is allowed by the so primitive. This primitive is quite similar to the *acid* primitive discussed by Cardelli and Gordon in [8]. The main difference is that *acid* does not mention the ambient dissolved while so does. This can be crucial in defining a type system for avoiding unwanted uses of so. Processes can be exchanged as values with the standard input and output primitives of the synchronous Ambient Calculus. We leave out here restriction: its introduction could be investigated following the lines of [12].

Ambients and Processes

Let $\mathcal{V}$ be a set of *process variables*, ranged over by $X,\ Y, \ldots$. Let $\mathcal{L}$ be a set of *ambient names* ranged over by $a, b, c \ldots$ and $\mathcal{M}$ be the set of *actions*, ranged over by $\mathsf{m}, \mathsf{n}, \ldots$, containing $in\, a$, $out\, a$, $opena$, $so\, a$ for all ambients $a \in \mathcal{L}$, and (X), $\langle P \rangle$ for all process variables X and processes P. The set $\mathcal{P}$ of *processes* (ranged over by $P,\ Q,\ R, \ldots$) is defined by

$$\mathcal{P} \ ::= \ \mathbf{0} \ \mid \ \mathcal{V} \ \mid \ \mathcal{M}.\mathcal{P} \ \mid \ \mathcal{L}[\mathcal{P}] \ \mid \ \mathcal{P} \,\big|\, \mathcal{P} \ \mid \ !\mathcal{P}.$$

$\mathcal{M}^{\updownarrow} \subseteq \mathcal{M}$ will denote the set of *mobility actions* containing only $in\, a$, $out\, a$, $open\, a$, $so\, a$ for all ambients $a \in \mathcal{L}$.
We assume that "." takes precedence over "|". So $\mathsf{m}\,.\alpha \,\big|\, \beta$ is read $(\mathsf{m}\,.\alpha)|\beta$.

As customary, the relation of structural congruence $\equiv$ is defined as the minimal reflexive, transitive and symmetric relation which is a congruence and moreover:

- satisfies $!P \equiv !P \,\big|\, P$;
- includes α-conversion;
- makes the operator $\big|$ commutative, associative, with $\mathbf{0}$ as neutral element.

The behavior of processes is then represented by the reduction relation defined in Figure 1. Note that the so action allows a process to open its enclosing ambient. This action is orthogonal to the other mobility actions (in , out , $open$) and cannot be internally simulated in the standard Ambient Calculus.

$(red\ in)$	$a[in\, b.P \,\big|\, Q] \,\big|\, b[R]]$	$\rightarrow b[a[P \,\big|\, Q] \,\big|\, R]$
$(red\ out)$	$a[b[out\, a.P \,\big|\, Q] \,\big|\, R]$	$\rightarrow a[R] \,\big|\, b[P \,\big|\, Q]$
(*red selfopen*)	$a[so\, a.P \,\big|\, Q]$	$\rightarrow P \,\big|\, Q$
$(red\ open)$	$open\, a.P \,\big|\, a[Q]$	$\rightarrow P \,\big|\, Q$
$(comm)$	$(X).P \,\big|\, \langle Q \rangle.R$	$\rightarrow P[X := Q] \,\big|\, R$
$(R - par)$	$P \rightarrow Q$	$\Rightarrow P \,\big|\, R \rightarrow Q \,\big|\, R$
$(R - amb)$	$P \rightarrow Q$	$\Rightarrow n[P] \rightarrow n[Q]$
$(R- \equiv)$	$P' \equiv P' \ \ P \rightarrow Q \ \ Q \equiv Q'$	$\Rightarrow P' \rightarrow Q'$

Fig. 1. Reduction

Observational Equivalence

In the ambient calculus the natural candidates to represent observables are the ambients. The following definition of observational preorder takes the notion of observable proposed in the original system [18].

Definition 1. *(i) We say that process* P exhibits *an ambient* a*, notation* $P \Downarrow a$ *if* $P \rightarrow^* a[Q] \,\big|\, R$ *for some processes* $Q,\ R$*.*
(ii) $P \sqsubseteq Q$ *if for all closing context* $\mathcal{C}[\,]$ *and ambients* a*:* $\mathcal{C}[P] \Downarrow a \ \Rightarrow\ \mathcal{C}[Q] \Downarrow a$*.*
(iii) $P \cong Q$ *if* $P \sqsubseteq Q$ *and* $Q \sqsubseteq P$*.*

Remark 1. Note that $P \rightarrow Q$ implies $Q \sqsubseteq P$, but in general $P \not\cong Q$. For instance let $P_1 = open\, a.\mathbf{0} \,\big|\, a[b[\mathbf{0}]]$ and $P_2 = b[\mathbf{0}]$. Then $P_1 \rightarrow P_2$ but $P_1 \not\cong P_2$ (take $\mathcal{C}[\,]$ as $[-]$).

3 Types

Like in type assignment systems for polymorphic λ-calculus, types are seen as properties of type-free objects rather than domains in which objects live. Types are intended to provide partial information about the processes they are associated with. Our language of types must be expressive enough to completely characterize process behaviors. We need, thus, to consider both the ambient, mobility actions and parallel composition as type constructors. The (binary) type constructor for input actions is $(-).-$, which correspond to the $\rightarrow$ function type constructor. We use here this notation for uniformity with the other type constructors. Similarly $\langle - \rangle.-$ is the type constructor for output actions. A type $\langle \sigma \rangle.\alpha$ represents processes which can produce an output of type σ and then leave a continuation of type α. For technical reasons (see Remark 2 of Section 5) we need to restrict the set of types allowed in the input-output fields of types.

The conjunction type constructor is added to represent nondeterminism. Type ω represents a property that is true of all processes. The set $\mathcal{T}$ of *types* (ranged over by $\alpha,\ \beta,\ \gamma, \ldots$) is then defined by

$$\mathcal{T} \ ::= \ \omega \ \mid \ \mathcal{M}^{\updownarrow}.\mathcal{T} \ \mid \ \langle \mathcal{T}^- \rangle.\mathcal{T} \ \mid \ (\mathcal{T}^-).\mathcal{T} \ \mid \ \mathcal{L}[\mathcal{T}] \ \mid \ \mathcal{T} \,\big|\, \mathcal{T} \ \mid \ \mathcal{T} \wedge \mathcal{T},$$

where $\mathcal{T}^- \subseteq \mathcal{T}$ is the subset of *simple* types, ranged over by $\sigma,\ \tau, \ldots$, containing all types without occurrences of the $\wedge$ operator. Intersection represent "may" nondeterminism. A process has type $\alpha \wedge \beta$ if it can possibly exhibit, although in different reduction paths, both property α and property β. We make the convention that $\wedge$ has the lowest precedence.

In connecting types to processes we must consider two distinct formal systems. One is to represent the logical structure of types, determined by their entailment relation (denoted $\leq$), and one to assign types to processes.

The logical structure of types is formalized as a partial order relation representing entailment. We write $\alpha \leq \beta$ to mean that property α entails property β. We write $\alpha \simeq \beta$ if $\alpha \leq \beta \leq \alpha$. Let $\simeq$ be the equivalence relation induced by $\leq$.

The formal rules for type entailment are represented in Figure 2. We say that two actions m and n *match* if either $\mathsf{m} \equiv (\sigma)$, $\mathsf{n} \equiv \langle \tau \rangle$ or $\mathsf{n} \equiv (\sigma)$, $\mathsf{m} \equiv \langle \tau \rangle$, and in both cases $\tau \leq \sigma$.

As pointed out in Remark 1 of Section 2, the execution of an action corresponds to a loss of capabilities. This is formalized by the axioms of the group "Reduction". Rule $(out - in)$ takes into account the fact that, in rule $(red\ in)$, after the consumption of the $in\, a$ action, the process inside a is always able to perform a sequence $out\, a,\ in\, a$ of

– Commutativity and distributivity of $\mid$

$$(\mid 1) \quad \alpha \mid \beta \simeq \beta \mid \alpha \qquad\qquad (\mid 2) \quad (\alpha \mid \beta) \mid \gamma \simeq \alpha \mid (\beta \mid \gamma)$$

– Axioms for ω:

$$(\omega 1) \quad \alpha \leq \omega \qquad\qquad (\omega 2) \quad \alpha \simeq \alpha \mid \omega$$

– Distributivity of $\wedge$

$$\begin{array}{ll} ([\,]\wedge) & a[\alpha \wedge \beta] \simeq a[\alpha] \wedge a[\beta] \\ (\mid \wedge) & \alpha \mid (\beta \wedge \gamma) \simeq (\alpha \mid \gamma) \wedge (\beta \mid \gamma) \\ (.\wedge) & \mathsf{m}.(\alpha \wedge \beta) \simeq \mathsf{m}.\alpha \wedge \mathsf{m}.\beta \end{array}$$

– Sequentialization

$$\begin{array}{ll} (. \mid_1) & \mathsf{m}.\alpha \mid \beta \leq \mathsf{m}.(\alpha \mid \beta) \\ (. \mid_2) & \mathsf{m}.\alpha \mid \mathsf{n}.\beta \simeq \mathsf{m}.(\alpha \mid \mathsf{n}.\beta) \wedge \mathsf{n}.(\mathsf{m}.\alpha \mid \beta) \quad \text{if } \mathsf{m} \text{ and } \mathsf{n} \text{ do not match} \\ (comm) & (\sigma).\beta \mid \langle\tau\rangle.\gamma \simeq \beta \mid \gamma \wedge (\sigma).(\beta \mid \langle\tau\rangle.\gamma) \wedge \langle\tau\rangle.((\alpha).\beta \mid \gamma) \quad \text{if } \tau \leq \sigma \end{array}$$

– Reduction

$$\begin{array}{ll} (in) \quad a[in\, b.\alpha \mid \beta] \mid b[\gamma] \leq b[a[\alpha \mid \beta] \mid \gamma] & (out) \quad a[b[out\, a.\alpha \mid \beta] \mid \gamma] \leq a[\gamma] \mid b[\alpha \mid \beta] \\ (\textit{selfopen}) \quad a[so\, a.\, \alpha \mid \beta] \leq \alpha \mid \beta & (open) \quad open\, a.\alpha \mid a[\beta] \leq \alpha \mid \beta \\ (out{-}in) \quad in\, a.out\, a.in\, a.\alpha \leq in\, a.\alpha & (in{-}out) \quad out\, a.in\, a.out\, a.\alpha \leq out\, a.\alpha \end{array}$$

– Congruence

$$(cg-[\,]) \quad \frac{\alpha \leq \beta}{a[\alpha] \leq a[\beta]} \qquad (cg-action) \quad \frac{\alpha \leq \beta}{\mathsf{m}.\alpha \leq \mathsf{m}.\beta}$$

$$(cg-()) \quad \frac{\sigma' \leq \sigma}{(\sigma).\alpha \leq (\sigma').\alpha} \qquad (cg-\langle\rangle) \quad \frac{\sigma \leq \sigma'}{\langle\sigma\rangle.\alpha \leq \langle\sigma'\rangle.\alpha}$$

$$(cg-\mid) \quad \frac{\alpha \leq \gamma \quad \beta \leq \delta}{\alpha \mid \beta \leq \gamma \mid \delta}$$

– Transitivity

$$(trans) \quad \frac{\alpha \leq \beta \quad \beta \leq \gamma}{\alpha \leq \gamma}$$

– Logical

$$\begin{array}{ll} (\wedge - r) \quad \alpha \wedge \beta \leq \beta & (\wedge - l) \quad \alpha \wedge \beta \leq \alpha \\ (\wedge - id) \quad \alpha \leq \alpha \wedge \alpha & (\wedge - \leq) \quad \dfrac{\alpha \leq \alpha' \quad \beta \leq \beta'}{\alpha \wedge \beta \leq \alpha' \wedge \beta'} \end{array}$$

Fig. 2. Type Entailment Rules

actions. A similar motivation holds for rule $(in - out)$. Sangiorgi [24] calls *stuttering* this phenomenum.

Axiom $(. \mid_2)$ represents the fact that two parallel processes can perform two distinct actions in any order. Axiom $(. \mid_1)$ is a consequence of this. If the two processes can communicate (axiom $(comm)$) we must take into account also this possibility. Axioms $(\omega 1)$, $(\omega 2)$ and rule $(cg- \mid)$ imply that $\alpha \mid \beta \leq \alpha$, i.e. parallel composition corresponds to increase of capabilities. Note also that, using $(\wedge - id)$, $(\omega 1)$, $(\mid 1)$, $(cg- \mid)$, $(\wedge - \leq)$, $(\omega 2)$, we get $\alpha \mid \beta \leq \alpha \wedge \beta$. As usual output is covariant (rule $(cg - \langle\rangle)$), while input is contravariant (rule $(cg - ())$).

Types will be always considered modulo $\simeq$. Note that $\simeq$ is preserved by both intersection and parallel composition with ω. The operators $\mid$ and $\wedge$ are associative so, for instance, we can write unambiguously $\alpha \mid \beta \mid \gamma$. Parallel composition of types are also considered modulo permutations, and intersection of types are considered modulo permutations and repetitions (rules $(\wedge - id)$, $(\wedge - l)$, $(\wedge - r)$).

A parallel composition $\alpha_1 \mid \ldots \mid \alpha_n$ will sometimes be denoted by $\overrightarrow{\alpha}$ in *vector* notation. An intersection of types $\alpha_1 \wedge \ldots \wedge \alpha_n$ will be denoted by $\bigwedge_{i\in[1\ldots n]} \alpha_i$. In this case $\beta \propto \bigwedge_{i\in[1\ldots n]} \alpha_i$ denote that $\beta \equiv \alpha_i$ for some $(1 \leq i \leq n)$.

A crucial technical notion is that of normal type.

Definition 2. *(i) The set $\mathcal{N} \subset \mathcal{T}$ of* normal *types is defined inductively in the following way:*

1. $\omega \in \mathcal{N}$.
2. $\omega \mid \phi \in \mathcal{N}$ *where* $\phi \in \mathcal{N}$.
3. $\mathsf{m}.\phi \in \mathcal{N}$ *where* $\mathsf{m} \in \mathcal{M}$ *and* $\phi \in \mathcal{N}$.
4. $a[\phi] \in \mathcal{N}$ *where* $\phi \in \mathcal{N}$.
5. $\phi \mid a[\psi] \in \mathcal{N}$ *where* $\phi, \psi \in \mathcal{N}$.

(ii) A normal type is easy *if it is normal and is of the form 3.*

Let $\phi, \psi, \xi, \chi ...$ range over normal types. In general a normal type different from ω has the form $\phi \mid a_1[\psi_1] \mid \ldots \mid a_n[\psi_n]$ (or $\phi \mid \overrightarrow{a[\psi]}$ in vector notation) where ϕ can be missing or is easy or ω and $n \geq 0$.

Note that normal types do not contain intersections. A normal type represents a process in which, in each ambient, there is at most one possible action that can be performed. Nondeterminism is left, however, since in the same normal type different actions can be enabled in different ambients.

Definition 3. *Let $\simeq_0$ be the equivalence relation defined by the rules obtained by replacing $\leq$ by $\simeq_0$ in the rules $(\mid 1)$, $(\mid 2)$, $(\omega 2)$, $([\,]\wedge)$, $(\mid \wedge)$, $(.\wedge)$, $(. \mid_2)$, $(comm)$ and $(trans)$ of Figure 2.*

We can show by structural induction on types that each type has a unique normal form modulo permutations and parallel composition with ω.

Lemma 1. *For all $\alpha \in \mathcal{T}$ there is a unique type $\bigwedge_{i\in I} \phi_i$, where ϕ_i $(i \in I)$ are normal types such that $\alpha \simeq_0 \bigwedge_{i\in I} \phi_i$. We call it the* normal form *of α, denoted $nf(\alpha)$.*

Ambients are inactive with respect to normal forms in the following sense.

Lemma 2. *Let $\phi, \overrightarrow{a[\psi]}$ be normal types. Then $\phi \propto nf(\alpha)$ iff $\phi \mid \overrightarrow{a[\psi]} \propto nf(\alpha \mid \overrightarrow{a[\psi]})$.*

Lemma 3. *Let $nf(\alpha) = \bigwedge_{i\in I} \phi_i$. Then*

1. $nf(a[\alpha])) = \bigwedge_{i\in I} a[\phi_i]$;
2. $nf(\mathsf{m}.\alpha)) = \bigwedge_{i\in I} \mathsf{m}.\phi_i$;
3. *Let* $nf(\beta)) = \bigwedge_{j\in J} \psi_j$. *Then* $nf(\alpha \mid \beta) = \bigwedge_{i\in I, j\in J} nf(\phi_i \mid \psi_j)$.

The entailment relation can be specialized to normal types. Let $\leq_N \subset \mathcal{N}\times\mathcal{N}$ denote this relation, defined by the rules of Figure 3.

In the rules for $\leq_N$ the r.h.s. is naturally a normal type whenever the l.h.s. is normal, except for rules $(open^N)$ and (*selfopen*^{N}), since the parallel composition of two normal types is not normal, in general.

The following lemma is crucial for representing the entailment properties of normal types.

Lemma 4. *Let ϕ, ψ, ξ, χ be normal types such that $\phi \leq_N \psi$ and $\xi \leq_N \chi$. Then for all $\nu \propto nf(\psi \mid \chi)$ there is $\mu \propto nf(\phi \mid \xi)$ such that $\mu \leq_N \nu$.*

Proof sketch. It is enough to prove the lemma assuming $\phi \leq_N \psi$ and $\xi \equiv \chi$. The general property can be easily obtained by transitivity. The proof is then by induction on the proof of $\phi \leq_N \psi$. The most difficult cases are when $\leq_N$ has been obtained by rules $(open^N)$ and (*selfopen*^{N}). The proof of these cases requires a careful analysis of the shape of the normal forms and is not given here.

The main lemma of this section relates normal forms and $\leq_N$ to $\leq$.

Lemma 5. *Let $\alpha \leq \beta$. Then for all $\psi \propto nf(\beta)$ there exists $\phi \propto nf(\alpha)$ such that $\phi \leq_N \psi$.*

Proof. By induction on the proof of $\leq$. The most difficult case is that of rule $(cg - \mid)$ which is handled using Lemmas 4 and 3(3).

Corollary 1. *Let ϕ, ψ be normal types. Then $\phi \leq \psi$ iff $\phi \leq_N \psi$.*

4 Type Inference

It is rather natural to devise type assignment rules for ambients and processes. They are represented in Figure 4, where $\Gamma : \mathcal{V} \rightharpoonup \mathcal{T}^-$ is a mapping from process variables to simple types. Let $\Gamma[X := \sigma]$ denote the mapping equal to Γ except that its value at X is σ. As usual processes are considered modulo renaming of bound variables.

The system $\vdash$ has only introduction rules for the various constructors: elimination rules are replaced by rule $(\leq)$.

We can prove by a simple induction on deductions a generation lemma.

Lemma 6 (Generation Lemma).

1. $\Gamma \vdash \mathbf{0} : \alpha$ *iff* $\alpha \simeq \omega$;

– Commutativity and distributivity of $\mid$

$$(\mid 1^N) \quad \phi \mid \psi \simeq_N \psi \mid \phi \text{ provided } \phi \mid \psi \text{ is normal}$$
$$(\mid 2^N) \quad (\phi \mid \psi) \mid \xi \simeq_N \phi \mid (\psi \mid \xi) \text{ provided } (\phi \mid \psi) \mid \xi \text{ is normal}$$

– Sequentialization

$$(. \mid_1^N) \quad \mathsf{m}.\phi \mid \overrightarrow{a[\psi]} \leq \mathsf{m}.(\phi \mid \overrightarrow{a[\psi]})$$

– Axioms for ω:

$$(\omega 1^N) \quad \phi \leq_N \omega \qquad\qquad (\omega 2^N) \quad \phi \mid \omega \simeq_N \phi$$

– Reduction

$$(in^N) \quad a[in\, b.\phi \mid \overrightarrow{c[\psi]}] \mid b[\xi] \leq_N b[a[\phi \mid \overrightarrow{c[\psi]}] \mid \xi]$$
$$(out^N) \quad a[b[out\, a.\phi \mid \overrightarrow{c[\psi]}] \mid \xi] \leq_N a[\xi] \mid b[\phi \mid \overrightarrow{c[\psi]}]$$
$$(\mathit{selfopen}^N) \quad a[so\, a.\phi \mid \overrightarrow{c[\chi]}] \mid \psi \leq_N \xi \mid \overrightarrow{c[\chi]} \quad \text{for all } \xi \propto nf(\phi \mid \psi)$$
$$(open^N) \quad open\, a.\phi \mid a[\psi] \leq_N \xi \quad \text{for all } \xi \propto nf(\phi \mid \psi)$$
$$(out{-}in^N) \quad in\, a.out\, a.in\, a.\phi \leq in\, a.\phi$$
$$(in{-}out^N) \quad out\, a.in\, a.out\, a.\phi \leq out\, a.\phi$$

– Congruence

$$(cg{-}[\,]^N) \quad \frac{\phi \leq_N \psi}{a[\phi] \leq_N a[\psi]} \qquad (cg{-}action^N) \quad \frac{\phi \leq_N \psi}{\mathsf{m}.\phi \leq_N \mathsf{m}.\psi}$$
$$(cg-()^N) \quad \frac{\sigma' \leq \sigma}{(\sigma).\phi \leq_N (\sigma').\phi} \qquad (cg-\langle\rangle^N) \quad \frac{\sigma \leq \sigma'}{\langle\sigma\rangle.\phi \leq_N \langle\sigma'\rangle.\phi}$$
$$(cg{-}\mid^N) \quad \frac{\phi \leq_N \phi' \quad \psi \leq_N \psi'}{\phi \mid a[\psi] \leq_N \phi' \mid a[\psi']}$$

– Transitivity

$$(trans^N) \quad \frac{\phi \leq_N \psi \quad \psi \leq_N \xi}{\phi \leq_N \xi}$$

Fig. 3. Entailment Rules for Normal Types

$$(\omega) \quad \Gamma \vdash P : \omega \qquad (\mathsf{m}^{\updownarrow}) \quad \frac{\Gamma \vdash P : \alpha \quad \mathsf{m} \in \mathcal{M}^{\updownarrow}}{\Gamma \vdash \mathsf{m}.P : \mathsf{m}.\alpha}$$

$$(input) \quad \frac{\Gamma[X := \sigma] \vdash P : \alpha}{\Gamma \vdash (X).P : (\sigma).\alpha} \qquad (output) \quad \frac{\Gamma \vdash P : \sigma \quad \Gamma \vdash Q : \alpha}{\Gamma \vdash \langle P \rangle.Q : \langle \sigma \rangle.\alpha}$$

$$(amb) \quad \frac{\Gamma \vdash P : \alpha}{\Gamma \vdash a[P] : a[\alpha]} \qquad (\,|\,) \quad \frac{\Gamma \vdash P : \alpha \quad \Gamma \vdash Q : \beta}{\Gamma \vdash P \,|\, Q : \alpha \,|\, \beta}$$

$$(!) \quad \frac{\Gamma \vdash P : \alpha \quad \Gamma \vdash !P : \beta}{\Gamma \vdash !P : \alpha \,|\, \beta} \qquad (\leq) \quad \frac{\Gamma \vdash P : \alpha \quad \alpha \leq \beta}{\Gamma \vdash P : \beta}$$

$$(\wedge I) \quad \frac{\Gamma \vdash P : \alpha \quad \Gamma \vdash P : \beta}{\Gamma \vdash P : \alpha \wedge \beta}$$

Fig. 4. Type Inference Rules

2. $\Gamma \vdash \mathsf{m}.P : \alpha$ *and* $\mathsf{m} \in \mathcal{M}^{\updownarrow}$ *iff* $\Gamma \vdash P : \beta$ *and* $\mathsf{m}.\beta \leq \alpha$ *for some* β*;*
3. $\Gamma \vdash (X).P : \alpha$ *iff* $\Gamma[X := \sigma_i] \vdash P : \beta_i$ *for* $(1 \leq i \leq n)$*, and* $(\sigma_1).\beta_1 \wedge \ldots \wedge (\sigma_n).\beta_n \leq \alpha$ *for some* $\beta_1, \ldots, \beta_n, \sigma_1, \ldots, \sigma_n$*;*
4. $\Gamma \vdash \langle P \rangle.Q : \alpha$ *iff* $\Gamma \vdash P : \sigma$*,* $\Gamma \vdash Q : \beta$ *and* $\langle \sigma \rangle.\beta \leq \alpha$ *for some* σ, β*;*
5. $\Gamma \vdash a[P] : \alpha$ *iff* $\Gamma \vdash P : \beta$ *and* $a[\beta] \leq \alpha$ *for some* β*;*
6. $\Gamma \vdash P \,|\, Q : \alpha$ *iff* $\Gamma \vdash P : \beta$*,* $\Gamma \vdash Q : \gamma$ *and* $\beta \,|\, \gamma \leq \alpha$ *for some* β, γ*;*
7. $\Gamma \vdash !P : \alpha$ *iff* $\Gamma \vdash P : \beta_i$ *for* $(1 \leq i \leq n)$ *and* $\beta_1 \,|\, \ldots \,|\, \beta_n \leq \alpha$ *for some* $\beta_1, \ldots, \beta_n$*.*

Lemma 6 says that the types of a term can be obtained in an uniform way from the types of its subterms, and this will guarantee the compositionality of the filter model we will build in the next section.

Since we are in a "may" perspective, it is natural that a process P offers all the capabilities offered by one of its reducts Q (may be more). At the type assignment level this means that types are preserved under subject expansion. Of course subject reduction should not hold; for example, the reduction of a process P with the rule $(red\ open)$ produces a process that in general offers less ambients (and so has less types). Instead congruent processes have the same types. Both properties can be proved by induction on the definitions of $\equiv$ and $\rightarrow^*$ using Lemma 6.

Lemma 7 (Subject Congruence). $P \equiv Q$ *and* $\Gamma \vdash Q : \alpha \quad \Rightarrow \quad \Gamma \vdash P : \alpha.$

Lemma 8 (Subject Expansion). $P \rightarrow^* Q$ *and* $\Gamma \vdash Q : \alpha \quad \Rightarrow \quad \Gamma \vdash P : \alpha.$

5 The Filter Model

We capitalize on the type assignment system of previous section for defining a filter model of the ambient calculus. We mainly follow the development line of [12].

Let $\langle D; \sqsubseteq \rangle$ be a preorder. A subset L of D is a *filter* if L is a non-empty upper set, i.e., $l \in L$ and $l \sqsubseteq l'$ imply $l' \in L$, and every finite subset of L has a greatest lower bound in L.

Consider the set $\mathcal{T}$ of types with the inclusion $\leq$ defined in Section 3. The greatest lower bound of a finite non-empty set of types is the intersection of the types in the set. It is standard to prove that the of filters over $\mathcal{T}$ is a domain in the usual sense.

Lemma 9. *The set $\mathcal{F}(\mathcal{T})$ of filters over $\mathcal{T}$ ordered by set inclusion is a consistently complete algebraic lattice.*

If $A \subseteq \mathcal{T}$ is a non-empty set of types then $\uparrow A$ denotes the filter generated by A, obtained by closing A under finite intersection and (by upper closing A under) $\leq$. Let $\widetilde{par} : \mathcal{F}(\mathcal{T}) \times \mathcal{F}(\mathcal{T}) \to \mathcal{F}(\mathcal{T})$ be the function defined by $\widetilde{par}(F, G) = \uparrow \{\alpha \,|\, \beta \mid \alpha \in F \text{ and } \beta \in G\}$.

We can now give an interpretation of processes in $\mathcal{F}(\mathcal{T})$. Let Env be the set of environments, i.e.mappings $\rho : \mathcal{V} \to \mathcal{F}(\mathcal{T})$.

Definition 4. *The function $[\![-]\!] : Env \to \mathcal{P} \to \mathcal{F}(\mathcal{T})$ is defined in the following way:*

- $[\![\mathbf{0}]\!]_\rho = \uparrow \{\omega\}$
- $[\![act\ a.P]\!]_\rho = \uparrow \{act\ a.\alpha \mid \alpha \in [\![P]\!]_\rho\}$ *where* $act \in \{in\,, out\,, open\,, so\,\}$
- $[\![a[P]]\!]_\rho = \uparrow \{a[\alpha] \mid \alpha \in [\![P]\!]_\rho\}$
- $[\![(X).P]\!]_\rho = \uparrow \{(\sigma).\alpha \mid \alpha \in [\![P]\!]_{\rho[X:=\uparrow\{\sigma\}]}\}$
- $[\![\langle P\rangle.Q]\!]_\rho = \uparrow \{\langle\sigma\rangle.\alpha \mid \sigma \in [\![P]\!]_\rho \text{ and } \alpha \in [\![Q]\!]_\rho\}$
- $[\![P \,|\, Q]\!]_\rho = \widetilde{par}([\![P]\!]_\rho, [\![Q]\!]_\rho)$
- $[\![!P]\!]_\rho = fix(\lambda X \in \mathcal{F}(\mathcal{T}).\ \widetilde{par}([\![P]\!]_\rho, X))$,

where σ ranges over simple types.

The basic property of the filter model is that the interpretation of a term is defined by the set of its types. Define $\Gamma \models \rho$ if, for all X in the domain of Γ, $\Gamma(X) = \sigma$ implies $\sigma \in \rho(X)$.

Theorem 1. $[\![P]\!]_\rho = \{\alpha \in \mathcal{T} \mid$ *for some Γ: $\Gamma \models \rho$ and $\Gamma \vdash P : \alpha$*$\}$.

From rules (ω), $(\leq)$, and $(\wedge)$ we have that $[\![P]\!]_\rho \in \mathcal{F}(\mathcal{T})$ for all P and ρ. Subject expansion can now be rephrased into the following statement:

$$\text{if } P \to^* Q \text{ then } [\![P]\!]_\rho \supseteq [\![Q]\!]_\rho \text{ for all } \rho.$$

The inclusion on filters induces an ordering on terms.

Definition 5. *Let $P, Q \in \mathcal{P}$. $P \sqsubseteq_F Q$ if and only if $[\![P]\!]_\rho \subseteq [\![Q]\!]_\rho$ for all ρ.*

The order relation $\sqsubseteq_F$ can be easily characterized by means of the deducibility of types as follows.

Proposition 1. *Let $P, Q \in \mathcal{P}$. $P \sqsubseteq_F Q$ if and only if $\Gamma \vdash P : \alpha$ implies $\Gamma \vdash Q : \alpha$ for all Γ, α.*

We will prove that the filter model exactly mirrors the operational semantics, i.e., that it is adequate and complete, i.e. it is fully abstract.

Adequacy

The adequacy proof requires a double induction on types and deductions. Following a standard methodology, we split this induction by introducing a realizability interpretation of types as sets of terms. The underlying idea is that a process P belongs to the interpretation of a type α if and only if α can be derived for P.

First we give an interpretation of simple types, and then we build the interpretation of all types, taking into account Lemmas 1 and 5. In defining the interpretation of types we will use a somewhat stronger notion of reduction over processes.

Definition 6. *The reduction relation $\rightsquigarrow$ over $\mathcal{P}$ is defined by adding to the rules of Fig. 1 the following rules:*

$$\begin{array}{ll} (seq) & \mathsf{m}.P \,|\, Q \rightsquigarrow \mathsf{m}.(P \,|\, Q) \\ (red-out{-}in) & in\,a.out\,a.in\,a.P \rightsquigarrow in\,a.P \\ (red-in{-}out) & out\,a.in\,a.out\,a.P \rightsquigarrow out\,a.P \end{array}$$

where in rule (seq) we do not allow capturing of free variables when m is an input action.

Note that, as in the case of $\rightarrow$, the relation $\rightsquigarrow$ corresponds to a loss of capabilities. It is easy to verify that $\rightsquigarrow$ does not modify the notion of convergence, i.e. that $P \Downarrow a$ iff $P \rightsquigarrow^* a[Q] \,|\, R$ for some processes Q, R. A standard induction on the definition of $\rightsquigarrow$ shows that the subject expansion property holds also with respect to $\rightsquigarrow$ reductions.

Lemma 10. *$P \rightsquigarrow^* Q$ and $\Gamma \vdash Q : \alpha \quad \Rightarrow \quad \Gamma \vdash P : \alpha$.*

The interpretation of normal types as sets of terms is given by induction on the *weight* of normal and simple types defined as follows:

$$\begin{array}{ll} |\omega| = 1 & |\mathsf{m}.\sigma| = 1 + |\sigma| \text{ if } \mathsf{m} \in \mathcal{M}^{\updownarrow} \\ |(\sigma).\tau| = 1 + |\sigma| + |\tau| & |\langle\sigma\rangle.\tau| = 1 + |\sigma| + |\tau| \\ |a[\sigma]| = 1 + |\sigma| & |\sigma \,|\, \tau| = 1 + |\sigma| + |\tau| \end{array}$$

It is easy to verify that, if $nf(\sigma) = \bigwedge_{i\in I} \phi_i$, then for all $i \in I$ we get $|\phi_i| \leq |\sigma|$. This is crucial for the soundness of the following definition.

Let $\mathcal{P}^0$ be the set of *closed processes*.

Definition 7. *The interpretation of normal types is defined by:*

1. $[\![\omega]\!] = \mathcal{P}^0$.
2. *If $\mathsf{m} \in \mathcal{M}^{\updownarrow}$ then* $[\![\mathsf{m}.\phi]\!] = \{P \in \mathcal{P}^0 \mid P \rightsquigarrow^* \mathsf{m}.Q$ *such that* $Q \in [\![\phi]\!]\}$.
3. *Let $nf(\sigma) = \bigwedge_{i\in I} \phi_i$. Then*
   $[\![(\sigma).\phi]\!] = \{P \in \mathcal{P}^0 \mid P \rightsquigarrow^* (X).Q$ *such that* $\forall S \in \bigcap_{i\in I} [\![\phi_i]\!].\ Q[X := S] \in [\![\phi]\!]\}$.
4. *Let $nf(\sigma) = \bigwedge_{i\in I} \phi_i$. Then*
   $[\![\langle\sigma\rangle.\phi]\!] = \{P \in \mathcal{P}^0 \mid P \rightsquigarrow^* \langle S\rangle.Q$ *such that* $Q \in [\![\phi]\!]$ *and* $S \in \bigcap_{i\in I} [\![\phi_i]\!]\}$.
5. $[\![a[\phi]]\!] = \{P \in \mathcal{P}^0 \mid P \rightsquigarrow^* a[Q] \,|\, R$ *such that* $Q \in [\![\phi]\!]\}$.
6. $[\![\phi \,|\, a[\psi]]\!] = \{P \in \mathcal{P}^0 \mid P \rightsquigarrow^* Q \,|\, a[R]$ *such that* $Q \in [\![\phi]\!]$ *and* $R \in [\![\psi]\!]\}$.

We need to prove the soundness of the normal type inclusion relation with respect to the interpretation of normal types. To this aim we need the following Lemma.

Lemma 11. *Let ϕ, ψ be normal types. Then $P \in [\![\phi]\!]$ and $Q \in [\![\psi]\!]$ imply $P \,|\, Q \in [\![\xi]\!]$ for all $\xi \propto nf(\phi \,|\, \psi)$.*

The soundness of the type inclusion relation can be shown by induction on $\leq_N$ definition. The most interesting case are axioms $(open^N)$ and (*selfopen*^{N}), which can be handled using Lemma 11.

Lemma 12. *Let ϕ, ψ be normal types. Then $\phi \leq_N \psi$ implies $[\![\phi]\!] \subseteq [\![\psi]\!]$.*

We can now define the interpretation of all types.

Definition 8. *The interpretation of arbitrary types is defined by:*

$$[\![\alpha]\!] = \bigcap_{\phi \propto nf(\alpha)} [\![\phi]\!].$$

We need the soundness of the type inclusion relation with respect to the interpretation of types.

Lemma 13. *If $\alpha \leq \beta$ then $[\![\alpha]\!] \subseteq [\![\beta]\!]$.*

As expected the type interpretation perfectly matches the type assignment system. Let $FV(P) = \{X_1, \ldots, X_n\}$: define

$$\Gamma \models P : \alpha \iff \forall Q_1 \in [\![\Gamma(X_1)]\!], \ldots, Q_n \in [\![\Gamma(X_n)]\!].P[X_1 := Q_1] \ldots [X_n := Q_n] \in [\![\alpha]\!].$$

Theorem 2 (Soundness and completeness of $\vdash$). *$\Gamma \vdash P : \alpha$ iff $\Gamma \models P : \alpha$.*

Proof. Soundness is proved by induction on the derivation of $\Gamma \vdash P : \alpha$, using Lemma 13 for rule ($\leq$).
As for completeness, by definition it suffices to show that if $\Gamma \models P : \phi$ then $\Gamma \vdash P : \phi$, when ϕ is normal. This proof can be done by induction on $|\phi|$ using Subject Expansion with respect to $\rightsquigarrow^*$ (Lemma 10).

Now we are able to characterize convergency by means of typing.

Lemma 14 (Resource property). *Let $P \in \mathcal{P}^0$. Then $\Gamma \vdash P : a[\omega]$ iff $P \Downarrow a$.*

Proof. $\Gamma \vdash P : a[\omega]$ iff (by Theorem 2) $P \in [\![a[\omega]]\!]$ iff (by Definition 7) $P \rightsquigarrow^* a[Q] \,|\, R$ for some processes Q, R.

We can now conclude the adequacy proof.

Theorem 3 (Adequacy). *If $P \sqsubseteq_F Q$ then $P \sqsubseteq Q$.*

Proof. If $\mathcal{C}[P] \Downarrow a$ then by Lemma 14 we get $\Gamma \vdash \mathcal{C}[P] : a[\omega]$. This together with $P \sqsubseteq_F Q$ imply $\Gamma \vdash \mathcal{C}[Q] : a[\omega]$, so we can conclude $\mathcal{C}[Q] \Downarrow a$ using again Lemma 14.

Completeness

Our completeness proof relies on building *test terms* $T_\phi^{x,y}$, where ϕ is a normal type and x, y are fresh ambient names with respect to ϕ. Their intended behavior is that, for all normal types ϕ, $x[P] \mid T_\phi^{x,y} \Downarrow y$ iff $\Gamma \vdash P : \phi$. The process P under testing is formerly enclosed in an ambient x for technical convenience.

In building these terms it is useful to have a process which exhibits ambient y iff it is in parallel with a process which exhibits all ambients $x_1, \ldots, x_n$.

Lemma 15. *Let w be a fresh ambient name. Let's define*

$$H^{x_1,\ldots,x_n \Rightarrow y} = w[in\, x_1.out\, x_1 \ldots in\, x_n.out\, x_n.y[out\, w]]\,.$$

Then $H^{x_1,\ldots,x_n \Rightarrow y} \mid P \Downarrow y$ iff $P \Downarrow x_i$ for all $i \in \{1, \ldots, n\}$.

To define test terms we will use *characteristic terms*. If σ is a simple type, the characteristic term C_σ is the "typical" term of type σ, i.e. we require that $\vdash \mathsf{C}_\sigma : \sigma$ and that

$$\vdash \mathsf{C}_\sigma : \alpha \text{ implies } \vdash Q : \alpha \text{ for all } Q \in [\![\sigma]\!].$$

Test terms and characteristic terms are build by simultaneous induction on the weight of normal types (defined at page 11). We define a test term for each normal type and a characteristic term for each simple type. We assume to have an unlimited source of ambient names, and to be able to pick new ambients names without clashing with the ambient names occurring in the processes we are testing.

Definition 9 (Test and Characteristic Terms).

Let ϕ be a normal type and σ a simple type. The test terms $T_\phi^{x,y}$ *and* characteristic terms C_σ *are defined by induction the weight of normal and simple types in the following way.*

Test terms:

- $T_\omega^{x,y} = p[in\, x.out\, x.y[out\, p]]$
- $T_{\omega \mid \phi}^{x,y} = T_\phi^{x,y}$
- $T_{in\, a.\phi}^{x,y} = a[p[in\, x.so\, p.out\, a.in\, v.in\, z]] \mid v[z[open\, x.t[out\, z.out\, v.open\, v.open\, a]]] \mid open\, t \mid T_\phi^{z,y}$
- $T_{out\, a.\phi}^{x,y} = p[in\, x.so\, p.in\, v.in\, a.in\, z] \mid v[a[z[open\, x.t[out\, z.out\, v.open\, v.open\, a]]]] \mid open\, t \mid T_\phi^{z,y}$
- $T_{open\, a.\phi}^{x,y} = p[in\, x.so\, p.a[in\, v.in\, z]] \mid v[z[open\, x.t[out\, z.out\, v.open\, v]]] \mid open\, t \mid T_\phi^{z,y}$
- $T_{so\, a.\phi}^{x,y} = p[in\, x.so\, p.in\, v.in\, z.in\, a] \mid v[z[a[open\, x.t[out\, z.out\, v.open\, v]]]] \mid open\, t \mid T_\phi^{z,y}$
- $T_{(\sigma).\phi}^{x,y} = p[in\, x.so\, p.in\, v.in\, z] \mid v[z[open\, x \mid \langle \mathsf{C}_\sigma \rangle.t[out\, z.out\, v.open\, v]]] \mid open\, t \mid T_\phi^{z,y}$

– $T^{x,y}_{\langle\sigma\rangle.\phi} = p[in\, x.so\, p.in\, v.in\, z] \mid$
$v[z[open\, x \mid (X).t[T^{q_1,w_1}_{\phi_1} \mid q_1[X] \mid \ldots \mid T^{q_n,w_n}_{\phi_n} \mid q_n[X]$
$\mid H^{w_1,\ldots,w_n \Rightarrow w} \mid open\, w.out\, z.out\, v.open\, v]]] \mid open\, t \mid T^{z,y}_{\phi}$
where $\bigwedge_{i\in[1\ldots n]} \phi_i = nf(\sigma)$.

– $T^{x,y}_{a[\psi]} = p[in\, x.in\, a.so\, p.out\, x.in\, v.in\, w]$
$\mid v[w[open\, a.t[out\, w.out\, v.open\, v]]] \mid open\, t \mid T^{w,y}_{\psi}$

– $T^{x,y}_{\phi \mid a[\psi]} = p[in\, x.in\, a.so\, p.out\, x.in\, v.in\, w]$
$\mid v[w[open\, a.t[out\, w.out\, v.open\, v]]] \mid open\, t \mid T^{x,q}_{\phi} \mid T^{w,z}_{\psi} \mid H^{q,z\Rightarrow y}$

Characteristic terms:

– $C_\omega = \mathbf{0}$
– $C_{m.\sigma} = m.C_\sigma \quad if\ m \in \mathcal{M}^{\updownarrow}$
– $C_{\langle\sigma\rangle.\tau} = \langle C_\sigma\rangle.C_\tau$
– $C_{(\sigma).\tau} = (X).(T^{q_1,w_1}_{\phi_1} \mid q_1[X] \mid \ldots \mid T^{q_n,w_n}_{\phi_n} \mid q_n[X] \mid H^{w_1,\ldots,w_n\Rightarrow y} \mid open\, y.C_\tau)$
where $\bigwedge_{i\in[1\ldots n]} \phi_i = nf(\sigma)$.
– $C_{a[\sigma]} = a[C_\sigma]$
– $C_{\sigma \mid \tau} = C_\sigma \mid C_\tau$

where we assume that all ambient names ($p, q, v, w, x, y, z, \ldots$*, except* a*) introduced in the definition of each* $T^{x,y}_{\phi}$ *are fresh. We call them the* extra *names of* $T^{x,y}_{\phi}$*, denoted* $EN(T^{x,y}_{\phi})$*. Similarly for* C_σ.

We can roughly summarize the behaviors of the test terms in the following way:

– $T^{x,y}_{\mathsf{m}.\phi}$ for a mobility action m:
  - the process $p[\,]$ moves the process $x[\,]$ inside the ambient z (which is inside the ambient v);
  - the process inside the ambient z opens x in an environment which allows the process formerly enclosed in x to consume m;
  - if the test is successful the process $t[\,]$ goes at top level and opens the ambient v;
  - the remaining process inside the ambient z will be tested by $T^{z,y}_{\phi}$.
– $T^{x,y}_{(\sigma).\phi}$:
  - the process $p[\,]$ moves the process $x[\,]$ inside the ambient z (which is inside the ambient v);
  - the process inside the ambient z opens the ambient x and offers then as output the characteristic term C_σ;
  - if the output is consumed the process $t[\,]$ goes at top level and opens the ambient v;
  - the remaining process inside the ambient z will be tested by $T^{z,y}_{\phi}$.
– $T^{x,y}_{\langle\sigma\rangle.\phi}$:
  - the process $p[\,]$ moves the process $x[\,]$ inside the ambient z (which is inside the ambient v);
  - the process inside the ambient z opens the ambient x and takes an input;

- if the input satisfies all tests ϕ_i for $i \in \{1, \ldots, n\}$ (where $\bigwedge_{i\in[1\ldots n]} \phi_i = nf(\sigma)$) then the process $t[\,]$ goes at top level and opens the ambient v;
- the remaining process inside the ambient z will be tested by $T^{z,y}_{\phi}$.

– $T^{x,y}_{\phi \,|\, a[\psi]}$:
- first the process $p[\,]$ moves the process $a[\,]$ inside the ambient w (which is inside the ambient v);
- then the process $t[\,]$ goes at top level and opens the ambient v;
- the remaining processes inside the ambients x, w will be tested respectively by the terms $T^{x,q}_{\phi}$ and $T^{w,z}_{\psi}$. The process $H^{q,z\Rightarrow y}$ lastly checks that both tests are successful.

Note that all the terms $T^{x,y}_{\phi}$ are reducible only when they are put in parallel with the ambient x and can exhibit y at top level only when reduced. So all of them must interact with x in the proper way to do the job. The basic property of test terms is the following.

Lemma 16. *1.* $\vdash \mathsf{C}_\sigma : \alpha$ *iff* $\sigma \leq \alpha$.

2. Let $P \in \mathcal{P}^0$ *be a process containing no occurrences of any ambient name belonging to* $EN(T^{x,y}_{\phi})$. *Then* $x[P] \,|\, T^{x,y}_{\phi} \Downarrow y$ *iff* $\vdash P : \phi$.

Remark 2. Note that characteristic terms do not seem to exist for arbitrary types containing intersection. The natural choice would be to take $\mathsf{C}_{\sigma\wedge\tau} = \mathsf{C}_\sigma + \mathsf{C}_\tau$ where $+$ is the nondeterministic choice operator, but it does not seem possible to represent $+$ our system.

Completeness now follows easily.

Theorem 4 (Completeness). *If* $P \sqsubseteq Q$ *then* $P \sqsubseteq_F Q$.

Proof. If $P \not\sqsubseteq_F Q$ then there are Γ, α such that $\Gamma \vdash P : \alpha$ and $\Gamma \not\vdash Q : \alpha$. Then by Lemmas 1, 5, and rule $(\leq)$ there is a normal type ϕ such that $\Gamma \vdash P : \phi$ and $\Gamma \not\vdash Q : \phi$. Let $\tau = (\sigma_1).\ldots.(\sigma_n).\phi$, where $FV(P \,|\, Q) = \{X_1, \ldots, X_n\}$ and $\Gamma(X_i) = \sigma_i$ for $i \in \{1, \ldots, n\}$. By Lemma 16 we get that $T^{x,y}_{\phi} \,|\, x[(X_1).\ldots.(X_n).P] \Downarrow y$ and $T^{x,y}_{\phi} \,|\, x[(X_1).\ldots.(X_n).Q] \not\Downarrow y$. So we conclude $P \not\sqsubseteq Q$.

6 Final Remarks

If we drop from our language the so primitive, leaving then only the standard mobility actions of the Ambient Calculus, we have immediately that $\mathcal{F}(\mathcal{T})$ is also a model of the resulting language. This model is adequate ($P \sqsubseteq_F Q$ implies$P \sqsubseteq Q$) but not fully abstract. To show this take, for instance:

$$P_1 = a[b[out\, c]]$$
$$P_2 = a[b[open\, d]] \,|\, d[in\, b.out\, c]$$

The process P_1 and P_2 are incomparable in the model (we can find proper types separating them) but operationally, in the standard Ambient Calculus, we have $P_1 \sqsubseteq P_2$. In

fact both exhibit the ambient a which contains only an ambient b. To show that $P_1 \not\sqsubseteq P_2$ we should find a context allowing to exercise the action $out\, c$ in P_1. But to obtain this we need to enclose b in an ambient c and this is possible only if we eventually open a. In this case, in a may perspective, we cannot avoid that d jumps into b and is opened there, allowing P_2 to show the same behavior as P_1. Instead using the so action we are able to build a context that separatesP_1 and P_2.

We are aware that so can cause undesired behaviors, but we are confident that a suitable type discipline can avoid them. The design of such a discipline will be subject of further investigations.

Acknowledgements

We are grateful to Daniel Hirschkoff and to the anonymous referees for their useful comments and suggestions.

References

1. S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51(1-2):1–77, 1991.
2. S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.
3. T. Amtoft, A. J. Kfoury, and S. M. Pericas-Geertsen. What are polymorphically-typed ambients? In *ESOP'01*, volume 2028 of *LNCS*, pages 206–220, Berlin, 2001. Springer-Verlag.
4. H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940, 1983.
5. G. Boudol. Lambda-calculi for (strict) parallel functions. *Information and Computation*, 108(1):51–127, 1994.
6. M. Bugliesi and G. Castagna. Secure safe ambients. In *POPL'01*, pages 222–235, New York, 2001. ACM Press.
7. L. Cardelli, G. Ghelli, and A. D. Gordon. Mobility types for mobile ambients. In *ICALP'99*, volume 1644 of *LNCS*, pages 230–239, Berlin, 1999. Springer-Verlag.
8. L. Cardelli and A. D. Gordon. Mobile ambients. In *FoSSaCS'98*, volume 1378 of *LNCS*, pages 140–155, Berlin, 1998. Springer-Verlag.
9. L. Cardelli and A. D. Gordon. Types for mobile ambients. In *POPL'99*, pages 79–92, New York, 1999. ACM Press.
10. L. Cardelli and A. D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *POPL'00*, pages 365–377, New York, 2000. ACM Press.
11. M. Coppo and M. Dezani-Ciancaglini. A fully abstract model for mobile ambients. In *TOSCA'01*, volume 62 of *ENTCS*. Elsevier Science, 200X. to appear.
12. F. Damiani, M. Dezani-Ciancaglini, and P. Giannini. A filter model for mobile processes. *Mathematical Structures in Computer Science*, 9(1):63–101, 1999.
13. P. Degano, F. Levi, and C. Bodei. Safe ambients: Control flow analysis and security. In *ASIAN'00*, volume 1961 of *LNCS*, pages 199–214, Berlin, 2000. Springer-Verlag.
14. M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. Finite models for conjunctive-disjunctive λ-calculi. *Theoretical Computer Science*, 170(1–2):83–128, 1996.
15. M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. A filter model for concurrent λ-calculus. *SIAM Journal on Computing*, 27(5):1376–1419, 1998.

16. M. Dezani-Ciancaglini and S. Ghilezan. A lambda model characterizing computational behaviors of terms. In *RPC'01*, pages 100–118. Tohoku University, 2001.
17. M. Dezani-Ciancaglini and I. Salvo. Security types for safe mobile ambients. In *ASIAN'00*, volume 1961 of *LNCS*, pages 215–236, Berlin, 2000. Springer-Verlag.
18. A. D. Gordon and L. Cardelli. Equational properties of mobile ambients. In *FoSSaCS'99*, volume 1578 of *LNCS*, pages 212–226, Berlin, 1999. Springer-Verlag.
19. R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In *SAS'99*, number 1694 in LNCS, pages 134–148, Berlin, 1999. Springer-Verlag.
20. C. Hartonas and M. Hennessy. Full abstractness for a functional/concurrent language with higher-order value-passing. *Information and Computation*, 145(1):64–106, 1998.
21. M. Hennessy. A fully abstract denotational model for higher-order processes. *Information and Computation*, 112(1):55–95, 1994.
22. M. Hennessy. Higher-order process and their models. In *ICALP'94*, volume 820 of *LNCS*, pages 286–303, Berlin, 1994. Springer-Verlag.
23. C.-H. L. Ong. Non-determinism in a functional setting. In *LICS'93*, pages 275–286, Montreal, Canada, 1993. IEEE Computer Society Press.
24. D. Sangiorgi. Extensionality and intensionality of the ambient logic. In *POPL'01*, pages 4–13, New York, 2001. ACM Press.
25. D. S. Scott. Domains for denotational semantics. In *ICALP'82*, volume 140 of *LNCS*, pages 577–613, Berlin, 1982. Springer-Verlag.