Logical Methods in Computer Science
Vol. 2 (2:3) 2006, pp. 1–35
www.lmcs-online.org

Submitted Apr. 4, 2005
Published Mar. 24, 2006

ON THE EXPRESSIVENESS OF THE AMBIENT LOGIC

DANIEL HIRSCHKOFF [a], ÉTIENNE LOZES [b], AND DAVIDE SANGIORGI [c]

[a] LIP - ENS Lyon
e-mail address: Daniel.Hirschkoff@ens-lyon.fr

[b] LSV- ENS Cachan
e-mail address: Etienne.Lozes@ens-lyon.fr

[c] Università di Bologna
e-mail address: Davide.Sangiorgi@cs.unibo.it

ABSTRACT. The Ambient Logic (AL) has been proposed for expressing properties of process mobility in the calculus of Mobile Ambients (MA), and as a basis for query languages on semistructured data.

In this paper, we study the expressiveness of AL. We define formulas for capabilities and for communication in MA. We also derive some formulas that capture finitess of a term, name occurrences and persistence. We study extensions of the calculus involving more complex forms of communications, and we define characteristic formulas for the equivalence induced by the logic on a subcalculus of MA. This subcalculus is defined by imposing an image-finiteness condition on the reducts of a MA process.

CONTENTS

1. Introduction 2
2. Background 5
2.1. Syntax of Mobile Ambients 5
2.2. Operational Semantics 6
2.3. The Ambient Logic 7
3. Formulas for capabilities and communications 8
3.1. Preliminary formulas: counting components and comparing names 9
3.2. Formulas for capabilities 9
3.3. Formulas for communication 12
4. Other intensional properties 15
4.1. Capturing finiteness 16
4.2. Formula for name occurrence 17

Key words and phrases: Spatial Logics, Mobile Ambients.
Work supported by european project FET-Global computing PROFUNDIS..
This work is a revised and extended version of parts of [San01] and [HLS02] (precisely, those parts that deal with expressiveness issues).

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-2 (2:3) 2006

4.3. Formulas for persistence 18
5. Characteristic formulas 21
5.1. Intensional bisimilarity 22
5.2. The sub-calculus $\mathrm{MA_{IF}}$ 23
6. Extensions of the calculus 26
6.1. Capabilities in messages 26
6.2. Synchronous Ambients 30
6.3. Other Extensions 34
References 34

1. Introduction

The *Ambient Logic*, AL, [CG00] is a modal logic for expressing properties of processes in the calculus of Mobile Ambients, MA [CG98a, CG99]. In MA the unit of movement is an ambient, which, intuitively, is a named location. An ambient may contain other ambients, and *capabilities*, which determine the ambient movements. The primitives for movement allow: an ambient to enter a sibling ambient; an ambient to exit the parent ambient; a process to dissolve an ambient boundary. MA has a replication operator to make a process persistent, that is, to make infinite copies of the process available.

An ambient can be thought of as a labelled tree. The sibling relation on subtrees represents spatial contiguity; the subtree relation represents spatial nesting. A label may represent an ambient name or a capability; moreover, a replication tag on labels indicates the resources that are persistent.[1] The trees are unordered: the order of the children of a node is not important. As an example, the process $P \stackrel{\mathrm{def}}{=} !a[\mathsf{in}\ c] \mid \mathsf{open}\ a.\, b[\mathbf{0}]$ is represented by the tree:

$$\begin{tikzcd} & \cdot \arrow[dl, "!a"'] \arrow[dr, "\mathsf{open}\ a"] & \\ \cdot \arrow[d, "\mathsf{in}\ c"'] & & \cdot \arrow[d, "b"] \\ \cdot & & \cdot \end{tikzcd}$$

The replication $!a$ indicates that the resource $a[\mathsf{in}\ c]$ is persistent: unboundedly many such ambients can be spawned. By contrast, $\mathsf{open}\ a$ is ephemeral: it can open only one ambient.

Syntactically, each tree is finite. Semantically, however, due to replications, a tree is an infinite object. As a consequence, the temporal developments of a tree can be quite rich. The process P above (we freely switch between processes and their tree representation) has only one reduction, to $\mathsf{in}\ c \mid !a[\mathsf{in}\ c] \mid b[\mathbf{0}]$. However, the process $!a[\mathsf{in}\ c] \mid !\mathsf{open}\ a.\, b[\mathbf{0}]$ can evolve into any process of the form

$$\mathsf{in}\ c \mid \ldots \mid \mathsf{in}\ c \mid b[\mathbf{0}] \mid \ldots \mid b[\mathbf{0}] \mid !a[\mathsf{in}\ c] \mid !\mathsf{open}\ a.\, b[\mathbf{0}]\,.$$

In general, a tree may have an infinite temporal branching, that is, it can evolve into an infinite number of trees, possibly quite different from each other (for instance, pairwise behaviourally unrelated). Technically, this means that the trees are not *image-finite*.

In summary, MA is a calculus of dynamically-evolving unordered edge-labelled trees, and AL is a logic for reasoning on such trees. The actual definition of satisfaction of the formulas of AL is given on MA processes quotiented by a relation of *structural congruence*,

[1] We are using a tree representation different from that of Cardelli and Gordon, but more convenient to our purposes.

which equates processes with the same tree representation. (This relation is similar to Milner's structural congruence for the π-calculus [Mil99].)

AL has also been advocated as a foundation of query languages for semistructured data [Car01]. Here, the laws of the logic are used to describe query rewriting rules and query optimisations. This line of work exploits the similarities between dynamically-evolving edge-labelled trees and standard models of semistructured data.

AL has a connective that talks about *time*, that is, how processes can evolve: the formula $\diamond \mathcal{A}$ is satisfied by those processes with a future in which $\mathcal{A}$ holds. The logic has also connectives that talk about *space*, that is, the shape of the edge-labelled trees that describe process distributions: the formula $n[\mathcal{A}]$ is satisfied by ambients named n whose content satisfies $\mathcal{A}$ (read on trees: $n[\mathcal{A}]$ is satisfied by the trees whose root has just a single edge n leading to a subtree that satisfies $\mathcal{A}$); the formula $\mathcal{A}_1 \mid \mathcal{A}_2$ is satisfied by the processes that can be decomposed into parallel components P_1 and P_2 where each P_i satisfies $\mathcal{A}_i$ (read on trees: $\mathcal{A}_1 \mid \mathcal{A}_2$ is satisfied by the trees that are the juxtaposition of two trees that respectively satisfy the formulas $\mathcal{A}_1$ and $\mathcal{A}_2$); the formula $\mathbf{0}$ is satisfied by the terminated process $\mathbf{0}$ (on trees: $\mathbf{0}$ is satisfied by the tree consisting of just the root node).

AL is quite different from standard modal logics. First, such logics do not talk about space. Secondly, they have more precise temporal connectives. The only temporal connective of AL talks about the many-step evolution of a system on its own. In standard modal logics, by contrast, the temporal connectives also talk about the potential interactions between a process and its environment. For instance, in the Hennessy-Milner logic [HM85], the temporal modality $\langle\mu\rangle.\mathcal{A}$ is satisfied by the processes that can perform the action μ and become a process that satisfies $\mathcal{A}$. The action μ can be a reduction, but also an input or an output. The lack of temporal connectives in the ambient logic is particularly significant because in MA interaction between a process and its environment can take several forms, originated by the communication and the movement primitives. (There are 9 such forms; they appear as labels of transitions in a purely SOS semantics of MA [CG98b, LS00].)

This paper is essentially devoted to the study of the expressiveness of AL. The results we present show that AL is actually a very expressive formalism. In particular, we are able to derive formulas expressing capabilities of processes for movement and for communication, as well as the persistence of processes (as given by the replication operator), and free occurrences of names in processes. The ability to derive such constructions is surprising, considering that there is no connective in the logic that is directly related to such properties: no construct mentions the capabilities of the calculus, nor does the logic include infinitary operators, or operators that talk about resources with infinite multiplicity.

Our results are established using nontrivial technical developments, and the methods we exploit are of interest in their own. More precisely, the general approach to derive expressiveness formulas is to exploit adjunct connectives to introduce a form of contextual reasoning, together with the temporal modality to make it possible to observe the desired properties. It can be noted that related constructions have been introduced in the setting of Separation Logic [Rey02] in order to express weakest preconditions for pointer manipulation instructions in an imperative language.

The expressive power of AL that we thus prove has several consequences. The first consequence is that we are able to define *characteristic formulas* for image-finite Ambient processes, i.e., formulas that capture the equivalence class of a process with respect to the induced logical equivalence. This is in contrast with usual results in modal logics. Typically, the definition of characteristic formulas exploits fixed-point operators, and the characterised processes are finite-state [GS86, SI94]. As mentioned above, AL has no fixed-point operator; moreover the image-finiteness condition on processes is weaker than finite-state. ('Image-finite' expresses finiteness on internal reductions, whereas 'finite-state' also takes into account computations containing visible actions such as input and output actions.)

Another major consequence of our results is to show that AL is an *intensional* logic. Informally, this holds because the logic allows one to inspect the structure of processes, not only by separating subcomponents of a process, but also by capturing its interaction capabilities. More formally, intensionality of the Ambient Logic is expressed by showing that the equivalence induced by the logic coincides with structural congruence on processes. This result, that is established using the constructions we have discussed above (and, in particular, characteristic formulas), says that AL is a very fine grained logic.

Structure of the paper. Section 2 introduces the calculus and the logic we study in this paper. Sections 3 and 4 present two main contributions in terms of expressiveness of AL: we define some formulas capturing respectively some syntactical constructions of the calculus (capabilities for movement and communication) and some nontrivial properties of processes (finiteness, occurrences of free names, and persistence). In Section 5, we exploit these constructions to define characteristic formulas for logical equivalence. Intensional bisimilarity, which, for the purposes of the present work, is a technical device that is needed to reason about characteristic formulas, is presented in Subsection 5.1. The proofs of the main properties enjoyed by intensional bisimilarity are not provided, and can be found in a companion paper [HLS05]. Finally, in Section 6, we study extensions of the calculus we work with, and show our results can be adapted to the corresponding settings.

The results of this paper come from the two conference papers [San01] and [HLS02]: in [San01], the author presented the encoding of the modalities for capabilities and communications (Sections 3 and 6) and the definition of intensional bisimilarity, whereas the formulas capturing finiteness, name occurrence, and persistence (Section 4) and the characteristic formulas (Section 5) come from [HLS02]. This paper focuses on the expressiveness results coming from these two conference papers, whereas a companion paper [HLS05] presents the separability results.

Developments. By the time the writing of the present paper was completed, a few works have appeared that make use of results or methods presented here. We discuss them below.

The 'contextual games' we have discussed above have been exploited in several settings. Along the lines of the derivation of formulas capturing Mobile Ambients capabilities, [HLS03] extends and develops this line of research in the setting of a sub-logic of AL, that is applied to reason about MA and π-calculus processes. Other interesting properties can be derived using this approach. An example is quantifiers elimination [CL04]. Another study [Hir04] demonstrates that in some sense, contextual games represent the logical counterpart of 'contextual testing' as in barbed equivalence [SW01].

Our expressiveness results also allow us to bring to light redundancies in spatial logics for concurrency. For example, an operator to express occurrences of free names in processes is analysed in related works [CG01, HLS03]. In the setting of the present work, such an operator is encodable in AL.

$h, k, \ldots n, m$			*Names*				*Processes*
η			*Names* $\cup$ *Variables*	P, Q, R	$::=$	$\mathbf{0}$	*(nil)*
					$\mid$	$P \mid Q$	*(parallel)*
			Expressions		$\mid$	$!P$	*(replication)*
M, N	$::=$	cap	*(capability)*		$\mid$	$M.P$	*(prefixing)*
					$\mid$	$\eta[P]$	*(ambient)*
			Capabilities		$\mid$	$\{\eta\}$	*(message)*
cap	$::=$	$\mathsf{in}\ \eta$	*(enter)*		$\mid$	$(x)\,P$	*(abstraction)*
	$\mid$	$\mathsf{out}\ \eta$	*(exit)*				
	$\mid$	$\mathsf{open}\ \eta$	*(open)*				

Table 1: The syntax of finite MA

This kind of encodability results allow one to compare different versions of spatial logics for concurrency, and are useful to assess minimality properties of the logics.

2. Background

This section collects the necessary background for this paper. It includes the MA calculus [CG98a] (semantic and syntax), and the Ambient Logic [CG00].

2.1. Syntax of Mobile Ambients. We recall here the syntax of MA [CG98a] (we sometimes call this calculus the *Ambient calculus*). We first consider the calculus in which only names, not capabilities, can be communicated; this allows us to work in an untyped calculus. We analyse extensions of the calculus in Section 6.

As in [CG00, Car99, CG04], the calculus has no restriction operator for creating new names. The restriction-free calculus has a more direct correspondence with edge-labelled trees and semistructured data.

Table 1 shows the syntax. Both the set of names and that of variables are infinite. Letters n, m, h range over names, x, y, z over variables; η ranges over names and variables. The expressions $\mathsf{in}\ \eta$, $\mathsf{out}\ \eta$, and $\mathsf{open}\ \eta$ are the *capabilities*, and are ranged over using cap. Messages and abstractions are the *input/output* (I/O) primitives. The metavariables M, N, for messages, will become usefull when considering extensions of the language (see Section 6). A *closed* process has no free variables. We ignore syntactic differences due to alpha conversion, and we write $P\{^n\!/\!_x\}$ for the result of substituting x with n in P. In the paper, all definitions and results are given only for closed processes, unless otherwise stated.

Given an integer $n > 0$, we will write $P_i, (1 \leq i \leq n)$ for a (finite) sequence of processes $P_1, \ldots, P_n$.

Processes having the same internal structure are identified. This is expressed by means of the *structural congruence relation*, $\equiv$, the smallest congruence such that:

$$\frac{}{\mathsf{open}\, n.\, P \mid n[Q] \longrightarrow P \mid Q}\ \texttt{Red-Open} \qquad \frac{}{n[\mathsf{in}\, m.\, P_1 \mid P_2] \mid m[Q] \longrightarrow m[n[P_1 \mid P_2] \mid Q]}\ \texttt{Red-In}$$

$$\frac{}{m[n[\mathsf{out}\, m.\, P_1 \mid P_2] \mid Q] \longrightarrow n[P_1 \mid P_2] \mid m[Q]}\ \texttt{Red-Out} \qquad \frac{P \longrightarrow P'}{n[P] \longrightarrow n[P']}\ \texttt{Red-Amb}$$

$$\frac{}{\{M\} \mid (x)\, P \longrightarrow P\{^M/_x\}}\ \texttt{Red-Com} \qquad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}\ \texttt{Red-Par}$$

$$\frac{P \equiv P' \qquad P' \longrightarrow P'' \qquad P'' \equiv P'''}{P \longrightarrow P'''}\ \texttt{Red-Str}$$

Table 2: The rules for reduction

$$P \mid \mathbf{0} \;\equiv\; P \qquad P \mid Q \;\equiv\; Q \mid P \qquad P \mid (Q \mid R) \;\equiv\; (P \mid Q) \mid R$$

$$!P \;\equiv\; !P \mid P \qquad !\mathbf{0} \;\equiv\; \mathbf{0} \qquad !(P \mid Q) \;\equiv\; !P \mid !Q \qquad !!P \;\equiv\; !P$$

As a consequence of results in [DZ00], that studies a richer calculus than the one we study, we have:

Theorem 2.1. *The relation $\equiv$ is decidable.* □

The two following syntactic notions will be useful below.

Definition 2.1 (Finite and single processes)**.**

- A closed process P is *finite* if there exists a process P' with no occurrence of the replication operator such that $P \equiv P'$.
- A closed process P is *single* if there exists P' such that either $P \equiv \mathsf{cap}.\, P'$ for some cap, or $P \equiv n[P']$ for some n, or $P \equiv (x)P$ for some x.

Unless otherwise stated, all results and definitions we state in the sequel are on closed terms.

2.2. **Operational Semantics.** The operational semantics of the calculus is given by a reduction relation $\longrightarrow$, defined by the rules presented in Table 2.2. The reflexive and transitive closure of $\longrightarrow$ is written $\Longrightarrow$.

Lemma 2.2. *If $P \longrightarrow Q$ then there is a derivation of the reduction in which* `Red-Str` *is applied, if at all, only as the last rule.* □

Lemma 2.2 shows that every reduction $P \longrightarrow P'$ has a normalised derivation proof. As a consequence, we have:

Lemma 2.3. *If $P \longrightarrow Q$ then either*

(1) $P \equiv R \mid m[n[\mathsf{out}\, m.\, P_1 \mid P_2] \mid P_3]$ *and* $Q \equiv R \mid n[P_1 \mid P_2] \mid m[P_3]$*, or*
(2) $P \equiv R \mid n[\mathsf{in}\, m.\, P_1 \mid P_2] \mid m[P_3]$ *and* $Q \equiv R \mid m[n[P_1 \mid P_2] \mid P_3]$*, or*
(3) $P \equiv R \mid \mathsf{open}\, n.\, P_1 \mid n[P_2]$ *and* $Q \equiv R \mid P_1 \mid P_2$*, or*

$$
\begin{array}{lcll}
\mathcal{A} & ::= & \top & \textit{(true)} \\
 & | & \neg\mathcal{A} & \textit{(negation)} \\
 & | & \mathcal{A} \vee \mathcal{B} & \textit{(disjunction)} \\
 & | & \forall\, x\,.\,\mathcal{A} & \textit{(universal quantification over names)} \\
 & | & \diamond\,\mathcal{A} & \textit{(sometime)} \\
 & | & \mathbf{0} & \textit{(void)} \\
 & | & \eta[\mathcal{A}] & \textit{(edge)} \\
 & | & \mathcal{A} \mid \mathcal{B} & \textit{(composition)} \\
 & | & \mathcal{A}@\eta & \textit{(localisation)} \\
 & | & \mathcal{A} \triangleright \mathcal{B} & \textit{(guarantee)}
\end{array}
$$

Table 3: The syntax of logical formulas

(4) $P \equiv R \mid \{n\} \mid (x)\, P_1$ *and* $Q \equiv R \mid P_1\{n/x\}$, *or*
(5) $P \equiv R \mid n[P_1]$, $Q \equiv R \mid n[Q_1]$ *and* $P_1 \longrightarrow Q_1$. □

We now introduce some forms of labelled transitions that we will use to give the interpretation of some of our logical constructions.

Definition 2.2 (Labelled transitions)**.** Let P be a closed process. We write:

- $P \stackrel{\mathsf{cap}}{\to} P'$, where cap is a capability, if $P \equiv \mathsf{cap}.\, P_1 \mid P_2$ and $P' = P_1 \mid P_2$.
- $P \stackrel{\{n\}}{\to} P'$ if $P \equiv \{n\} \mid P'$.
- $P \stackrel{?n}{\to} P'$ if $P \equiv (x)\, P_1 \mid P_2$ and $P' \equiv P_1\{n/x\} \mid P_2$.
- $P \stackrel{\mu}{\Longrightarrow} P'$, where μ is one of the above labels, if $P \Longrightarrow \stackrel{\mu}{\to} \Longrightarrow P'$ (where $\Longrightarrow \stackrel{\mu}{\to} \Longrightarrow$ is relation composition).
- **(stuttering)** $P \xrightarrow{(\mathsf{cap}_1,\mathsf{cap}_2)^\star} P'$ if there is $i \geq 1$ and processes $P_1, \ldots, P_i$ with $P = P_1$ and $P' = P_i$ such that $P_r \stackrel{\mathsf{cap}_1}{\Longrightarrow} \stackrel{\mathsf{cap}_2}{\Longrightarrow} P_{r+1}$ for all $1 \leq r < i$.
- Finally, $\stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow}$ is a convenient notation for compacting statements involving capability transitions. We let $\stackrel{\langle \mathsf{in\ n} \rangle}{\Longrightarrow}$ stand for $\xrightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^\star}$; similarly $\stackrel{\langle \mathsf{out\ n} \rangle}{\Longrightarrow}$ is $\xrightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star}$; and $\stackrel{\langle \mathsf{open\ n} \rangle}{\Longrightarrow}$ is $\Longrightarrow$.

2.3. **The Ambient Logic.** The logic has the propositional connectives, $\top, \neg\mathcal{A}, \mathcal{A} \vee \mathcal{B}$, and universal quantification on names, $\forall x.\, \mathcal{A}$, with the standard logical interpretation. The temporal connective, $\diamond\mathcal{A}$ has been discussed in the introduction. The spatial connectives, $\mathbf{0}$, $\mathcal{A} \mid \mathcal{B}$, and $\eta[\mathcal{A}]$, are the logical counterpart of the corresponding constructions on processes. $\mathcal{A} \triangleright \mathcal{B}$ and $\mathcal{A}@\eta$ are the logical adjuncts of $\mathcal{A} \mid \mathcal{B}$ and $\eta[\mathcal{A}]$ respectively, in the sense of being roughly their 'contextual inverse', as expressed in Definition 2.3 below.

The logic in [CG00] has also a *somewhere* connective, that holds of a process containing, at some arbitrary level of nesting of ambients, an ambient whose content satisfies $\mathcal{A}$. We do not consider this connective in the paper because we find it less fundamental than the other operators; in any case, its addition would not affect the results in the paper and has been seldomly considered in other works. (Further, we discuss in the final section a "strong" version of the sometimes modality.)

Definition 2.3 (Satisfaction)**.** The satisfaction relation between closed processes and closed formulas, written $P \models \mathcal{A}$, is defined as follows:

$$\begin{array}{lcl}
P \models \top & \stackrel{\mathrm{def}}{=} & \text{always true} \\
P \models \forall\, x \,.\, \mathcal{A} & \stackrel{\mathrm{def}}{=} & \text{for any } n,\ P \models \mathcal{A}\{n/x\} \\
P \models \neg\, \mathcal{A} & \stackrel{\mathrm{def}}{=} & \text{not } P \models \mathcal{A} \\
P \models \mathcal{A}_1 \mid \mathcal{A}_2 & \stackrel{\mathrm{def}}{=} & \exists P_1, P_2 \text{ s.t. } P \equiv P_1 \mid P_2 \text{ and } P_i \models \mathcal{A}_i,\ \ i = 1,2 \\
P \models \mathcal{A} \vee \mathcal{B} & \stackrel{\mathrm{def}}{=} & P \models \mathcal{A} \text{ or } P \models \mathcal{B} \\
P \models n[\mathcal{A}] & \stackrel{\mathrm{def}}{=} & \exists P' \text{ s.t. } P \equiv n[P'] \text{ and } P' \models \mathcal{A} \\
P \models \mathbf{0} & \stackrel{\mathrm{def}}{=} & P \equiv \mathbf{0} \\
P \models \Diamond \mathcal{A} & \stackrel{\mathrm{def}}{=} & \exists P' \text{ s.t. } P \Longrightarrow P' \text{ and } P' \models \mathcal{A} \\
P \models \mathcal{A}@n & \stackrel{\mathrm{def}}{=} & n[P] \models \mathcal{A} \\
P \models \mathcal{A} \triangleright \mathcal{B} & \stackrel{\mathrm{def}}{=} & \forall R,\ R \models \mathcal{A} \text{ implies } P \mid R \models \mathcal{B}
\end{array}$$

By definition, satisfaction is closed by structural congruence:

Lemma 2.4. *If $P \equiv Q$ and $P \models \mathcal{A}$, then also $Q \models \mathcal{A}$.* □

We give $\vee$ and $\wedge$ the least syntactic precedence, thus $\mathcal{A}_1 \triangleright \mathcal{A}_2 \wedge \mathcal{A}_3$ reads $(\mathcal{A}_1 \triangleright \mathcal{A}_2) \wedge \mathcal{A}_3$, and $\mathcal{A}_1 \triangleright (\Diamond \mathcal{A}_2 \wedge \Diamond \mathcal{A}_3)$ reads $\mathcal{A}_1 \triangleright ((\Diamond \mathcal{A}_2) \wedge (\Diamond \mathcal{A}_3))$. We shall use the dual of some connectives, namely the duals of linear implication ($\mathcal{A} \blacktriangleright \mathcal{B}$), of the sometime modality ($\Box \mathcal{A}$), of the parallel operator ($\|$), and the standard duals of universal quantification ($\exists\, x \,.\, \mathcal{A}$) and disjunction ($\mathcal{A} \wedge \mathcal{B}$); we also define (classical) implication ($\mathcal{A} \rightarrow \mathcal{B}$):

$$\mathcal{A} \wedge B \stackrel{\mathrm{def}}{=} \neg(\neg\mathcal{A} \vee \neg\mathcal{B}) \qquad \Box\mathcal{A} \stackrel{\mathrm{def}}{=} \neg\Diamond\neg\mathcal{A} \qquad \mathcal{A} \rightarrow \mathcal{B} \stackrel{\mathrm{def}}{=} \neg\mathcal{A} \vee \mathcal{B}$$

$$\exists\, x \,.\, \mathcal{A} \stackrel{\mathrm{def}}{=} \neg\forall\, x \,.\, \neg\mathcal{A} \qquad \mathcal{A} \blacktriangleright \mathcal{B} \stackrel{\mathrm{def}}{=} \neg(\mathcal{A} \triangleright \neg\mathcal{B}) \qquad \bot \stackrel{\mathrm{def}}{=} \neg\top$$

Thus $P \models \mathcal{A} \blacktriangleright \mathcal{B}$ iff there exists Q with $Q \models \mathcal{A}$ and $P \mid Q \models \mathcal{B}$, and $P \models \Box\mathcal{A}$ iff $P' \models \mathcal{A}$ for all P' such that $P \Longrightarrow P'$.

We now define the induced equivalence between processes induced by the logic:

Definition 2.4 (Logical equivalence)**.** For processes P and Q, we write $P {=_L} Q$ if for any closed formula $\mathcal{A}$ it holds that $P \models \mathcal{A}$ iff $Q \models \mathcal{A}$.

3. Formulas for capabilities and communications

In this section, we show that we can capture at a logical level prefixes of the language, both for movement and for communication.

3.1. **Preliminary formulas: counting components and comparing names.** We start by recalling some formulas from [CG00] that will be useful for some constructions presented below.

The Ambient Logic allows one to count the number of parallel components of a process. The formula below is true of a process that has exactly one parallel component that is different from $\mathbf{0}$.

$$\boxed{\mathtt{1comp} \stackrel{\text{def}}{=} \neg\,(\neg\,\mathbf{0} \mid \neg\,\mathbf{0}) \wedge \neg\,\mathbf{0}}$$

Lemma 3.1. *It holds that $P \models \mathtt{1comp}$ iff P is single.* $\square$

Similarly we define

$$\boxed{\mathtt{2comp} \stackrel{\text{def}}{=} \mathtt{1comp} \mid \mathtt{1comp}}$$

We may impose a given formula $\mathcal{A}$ to be satisfied by all single parallel components of a process, using the following definitions:

$$\boxed{\begin{array}{lcl} \mathcal{A}^{\forall} & \stackrel{\text{def}}{=} & \neg(\neg\mathcal{A} \mid \top) \\ \mathcal{A}^{\omega} & \stackrel{\text{def}}{=} & (\mathtt{1comp} \rightarrow \mathcal{A})^{\forall} \end{array}}$$

Lemma 3.2.

- *$P \models \mathcal{A}^{\forall}$ iff for any Q, R such that $P \equiv Q \mid R$, it holds that $Q \models \mathcal{A}$.*
- *$P \models \mathcal{A}^{\omega}$ iff all single parallel components of P satisfy $\mathcal{A}$.* $\square$

We shall use later the following derived formula, from [CG00], that expresses equality between names:

$$\boxed{m = n \stackrel{\text{def}}{=} (n[\top])@m}$$

Lemma 3.3. *$P \models m = n$ iff names m and n are equal.* $\square$

3.2. **Formulas for capabilities.** The two formulas below are true of a process that is (structurally congruent to) an ambient and (to) an empty ambient, respectively.

$$\boxed{\begin{array}{lcl} \mathtt{1amb} & \stackrel{\text{def}}{=} & \exists\, x\,.\, x[\top] \\ \mathtt{1amb0} & \stackrel{\text{def}}{=} & \exists\, x\,.\, x[\mathbf{0}] \end{array}}$$

Lemma 3.4.

- *$P \models \mathtt{1amb}$ iff $P \equiv n[Q]$, for some n and Q.*
- *$P \models \mathtt{1amb0}$ iff $P \equiv n[\mathbf{0}]$, for some n.* $\square$

To help understanding the definitions of the capability formulas, we first discuss some simpler formulas, which do not talk about the process underneath the prefix. We define, for names $n \neq h$:

$$\begin{array}{rcl}\langle\!\langle \mathsf{open}\ \ n\rangle\!\rangle & \stackrel{\text{def}}{=} & n[h[\mathbf{0}]] \triangleright \Diamond\ (h[\mathbf{0}] \mid \top) \\ & & \wedge\ \mathtt{1comp} \\ & & \wedge \neg\, \mathtt{1amb} \\ \langle\!\langle \mathsf{out}\ \ n\rangle\!\rangle & \stackrel{\text{def}}{=} & (\Diamond\ (h[\top] \mid n[\mathbf{0}]))@n@h \\ & & \wedge\ \mathtt{1comp} \\ & & \wedge \neg\, \mathtt{1amb}\end{array}$$

It holds that $P \models \langle\!\langle \mathsf{open}\ \ n\rangle\!\rangle$ iff $P \equiv \mathsf{open}\, n.\, P'$ for some P'. We sketch the proof. The sub-formula $\mathtt{1comp} \wedge \neg\, \mathtt{1amb}$ says that P is single and is not an ambient. Thus, modulo $\equiv$, process P can only be $\mathbf{0}$, $\mathsf{open}\, m.\, P'$, $\mathsf{in}\, m.\, P'$, $\mathsf{out}\, m.\, P'$, $(x)\ P'$, or $\{m\}$, for some m. The sub-formula $n[h[\mathbf{0}]] \triangleright \Diamond\ (h[\mathbf{0}] \mid \top)$ says that $P \mid n[h[\mathbf{0}]]$ can reduce to a process with an empty ambient h at the outermost level. From these requirements, we conclude that $P \equiv \mathsf{open}\, n.\, P'$, for some P'.

Similarly we prove that $P \models \langle\!\langle \mathsf{out}\ \ n\rangle\!\rangle$ iff $P \equiv \mathsf{out}\, n.\, P'$, for some P'. By the sub-formula $\mathtt{1comp} \wedge \neg\, \mathtt{1amb}$, process P is single and is not an ambient. By the sub-formula $(\Diamond\ (h[\top] \mid n[\mathbf{0}]))@n@h$,

$$n[h[P]] \models \Diamond\ (h[\top] \mid n[\mathbf{0}])$$

hence $P \equiv \mathsf{out}\, n.\, P'$, for some P', otherwise $h[P]$ could not exit n.

To obtain the full capability formulas we add some quantification on names. Formula $\langle\!\langle \mathsf{open}\ \ n\rangle\!\rangle.\, \mathcal{A}$ is thus defined as follows:

$$\begin{array}{rcl}\langle\!\langle \mathsf{open}\ \ n\rangle\!\rangle.\, \mathcal{A} & \stackrel{\text{def}}{=} & \forall\, y\, .\ n[y[\mathbf{0}]] \triangleright \Diamond\ (y[\mathbf{0}] \mid \mathcal{A}) \\ & & \wedge\ \mathtt{1comp} \\ & & \wedge \neg\, \mathtt{1amb} \\ \mathtt{1open} & \stackrel{\text{def}}{=} & \exists\, x\, .\ \forall\, y\, .\ x[y[\mathbf{0}]] \triangleright \Diamond\ (y[\mathbf{0}] \mid \top) \\ & & \wedge\ \mathtt{1comp} \\ & & \wedge \neg\, \mathtt{1amb}\end{array}$$

Remark 3.1 (Formulas containing free variables)**.** It will often be the case in the remainder of the paper that we define a formula involving a name, say n, and need the corresponding logical construction where a variable x is used instead of n. For instance, the formula $\mathtt{1open}$ above could be defined as as "$\exists\, x\, .\ \langle\!\langle \mathsf{open}\ \ x\rangle\!\rangle.\, \top$", which is not correct because $\langle\!\langle \mathsf{open}\ \ n\rangle\!\rangle.\, \mathcal{A}$ has been defined but $\langle\!\langle \mathsf{open}\ \ x\rangle\!\rangle.\, \mathcal{A}$ has not. In the sequel, when clear from the context, we shall allow ourselves to adopt nevertheless this abuse of notation, that should be understood as 'rewrite the definition of the corresponding formula using x instead of n' (see in particular the formulas to capture name reception, and their interpretation, in Lemma 3.15, and characteristic formulas for input guarded processes in Section 5).

Satisfaction being defined only between closed processes and closed formulas, the important point in doing so is to avoid reasoning about the satisfaction of formulas containing free variables: we shall therefore only write formulas containing an 'x' under the scope of a variable quantification.

Lemma 3.5. *$P \models \langle\!\langle \mathsf{open}\ n \rangle\!\rangle . \mathcal{A}$ iff $P \equiv \mathsf{open}\, n.\, P'$, for some P' such that $P' \Longrightarrow P''$ and $P'' \models \mathcal{A}$.*

$P \models \mathtt{1open}$ iff $P \equiv \mathsf{open}\, n.\, P'$ for some n and P'.

Proof. We only consider the first property, from which the second follows easily. The implication from right to left is easy.

For the reverse implication, we set

$$G \stackrel{\text{def}}{=} n[h[\mathbf{0}]] \rhd \Diamond\ (h[\mathbf{0}] \mid \mathcal{A})$$

where $h \notin \mathrm{n}(P)$. Since $P \models \mathtt{1comp}$, we have $P \equiv Q$, for some Q that is not a parallel composition. Since also $P \models \neg\, \mathtt{1amb}$, we infer that Q is not an ambient. Finally since $P \models G$, process Q cannot be of the form $\mathbf{0}, \mathsf{in}\, n.\, Q', \mathsf{out}\, n.\, Q', (x)\, Q', \{p\}$. For the same reason, Q cannot be a prefix $\mathsf{open}\, m.\, Q'$ with $m \neq n$. The only possibility left is $Q = \mathsf{open}\, n.\, Q'$, for some Q'.

Moreover, we have

$$n[h[\mathbf{0}]] \mid \mathsf{open}\, n.\, Q' \Longrightarrow R \text{ and } R \models h[\mathbf{0}] \mid \mathcal{A}$$

for some R. The first step of this reduction must be

$$n[h[\mathbf{0}]] \mid \mathsf{open}\, n.\, Q' \longrightarrow h[\mathbf{0}] \mid Q'$$

(up to $\equiv$). Since h is fresh, $h[\mathbf{0}]$ cannot interact with Q'. Hence

$$R \equiv h[\mathbf{0}] \mid Q''$$

for some Q'' such that $Q' \Longrightarrow Q''$. $\square$

Along the lines of our construction for the open prefix, we can define characteristic formulas for the in and out prefixes.

$$\boxed{\begin{array}{rcl}
\langle\!\langle \mathsf{out}\ n \rangle\!\rangle . \mathcal{A} & \stackrel{\text{def}}{=} & \forall\, x\, .\, \Big((\Diamond\ (x[\mathcal{A}] \mid n[\mathbf{0}]))@n@x \Big) \\
 & & \wedge\ \mathtt{1comp} \\
 & & \wedge\ \neg\, \mathtt{1amb} \\
\mathtt{1out} & \stackrel{\text{def}}{=} & \exists\, x\, .\, \langle\!\langle \mathsf{out}\ x \rangle\!\rangle \\
 & & \\
\langle\!\langle \mathsf{in}\ n \rangle\!\rangle . \mathcal{A} & \stackrel{\text{def}}{=} & \forall\, x\, .\, \Big((n[\mathbf{0}] \rhd \Diamond\ n[x[\mathcal{A}]])@x \Big) \\
 & & \wedge\ \mathtt{1comp} \\
 & & \wedge\ \neg\, \mathtt{1amb} \\
\mathtt{1in} & \stackrel{\text{def}}{=} & \exists\, x\, .\, \langle\!\langle \mathsf{in}\ x \rangle\!\rangle
\end{array}}$$

Lemma 3.6. *$P \models \langle\!\langle \mathsf{out}\ n \rangle\!\rangle . \mathcal{A}$ iff $P \equiv \mathsf{out}\, n.\, P'$, for some P' such that $P' \xrightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star} P''$ and $P'' \models \mathcal{A}$.*

Proof. Similar to the proof for the open prefix. The formula $\mathtt{1comp} \wedge \neg\, \mathtt{1amb}$ forces P to be single and not an ambient. Therefore $P \equiv Q$, for some Q whose outermost operator is not a parallel composition or an ambient. Then we should have

$$n[h[Q]] \models \Diamond\ (h[A] \mid n[\mathbf{0}])$$

This can only happen if Q is of the form $\mathsf{out}\, n.\, Q'$, for some Q' such that $Q' \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star} Q''$ and $Q'' \models \mathcal{A}$. $\square$

Lemma 3.7. *$P \models \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, \mathcal{A}$ iff $P \equiv \mathsf{in}\, n.\, P'$, for some P' such that $P' \xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^\star} P''$ and $P'' \models \mathcal{A}$.*

Proof. Similar to the previous proofs. The formula $\mathtt{1comp} \wedge \neg\, \mathtt{1amb}$ forces P to be single and not an ambient. Therefore $P \equiv Q$, for some Q whose outermost operator is not a parallel composition or an ambient. Then we should have

$$h[Q] \mid n[\mathbf{0}] \models \diamond\, n[h[\mathcal{A}]]$$

where h is fresh. As by previous arguments, this can only happen if Q is of the form $\mathsf{in}\, n.\, Q'$, and Q' reduces (with suttering) to Q'' satisfying $\mathcal{A}$. $\square$

Given a capability cap, we may define the *'necessity'* version of the *'possibility'* formulas we have just introduced as follows:

$$\boxed{[\![\mathsf{cap}]\!].\, \mathcal{A} \stackrel{\text{def}}{=} \langle\!\langle \mathsf{cap} \rangle\!\rangle.\, \top \ \wedge\ \neg \langle\!\langle \mathsf{cap} \rangle\!\rangle.\, \neg \mathcal{A}}$$

Lemma 3.8. *For any capability cap, formula $\mathcal{A}$ and term P, $P \models [\![\mathsf{cap}]\!].\, \mathcal{A}$ iff there is P' such that $P \equiv \mathsf{cap}.\, P'$, and, for any P'' such that $P' \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P''$, $P'' \models \mathcal{A}$.*

Note that necessity formulas are not the dual of the possibility formulas, as in standard modal logics, because of the spatial aspects of AL. For instance, $[\![\mathsf{in}\ n]\!].\, \top$ does not have the same interpretation as $\neg \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, \neg \top$, the latter being actually equivalent to $\top$.

Remark 3.2. We could think of deriving formulas for *modalities* $\stackrel{\mathsf{cap}}{\Longrightarrow}$, as in standard modal logics for concurrency [HM85], instead of capturing the syntactical prefixes corresponding to a capability cap. More precisely, we could look for a formula $\langle\!\langle \mathsf{cap} \rangle\!\rangle \mathcal{A}$ capturing processes P for which there is P' such that $P \stackrel{\mathsf{cap}}{\Longrightarrow} P'$ and $P' \models \mathcal{A}$. It turns out that spatial logics are more intensional, and make actions more difficult to express than connectives. In particular, we do not know how to express directly a modality corresponding to action $\stackrel{\mathsf{open}\ n}{\Longrightarrow}$.

3.3. Formulas for communication.

The first step to characterise I/O processes (i.e., messages or abstractions) is to get rid of other possible constructs for single terms, as follows:

$$\boxed{\mathtt{1comm} \ \stackrel{\text{def}}{=}\ \mathtt{1comp} \wedge \neg\, \mathtt{1amb} \wedge \neg\, \mathtt{1open} \wedge \neg\, \mathtt{1out} \wedge \neg\, \mathtt{1in}}$$

Lemma 3.9. *$P \models \mathtt{1comm}$ iff ($P \equiv \{p\}$ or $P \equiv (x)\, P'$), for some p and P'.*

The following formula, that holds of a process that is the parallel composition of two I/O processes, will also be useful:

$$\mathtt{2comm} \ \stackrel{\text{def}}{=}\ \mathtt{1comm} \mid \mathtt{1comm}\,.$$

The difficult part, however, is the definition of the I/O formulas for separating messages from abstractions, and also, within the messages and the abstractions, messages with different contents and abstractions with different behaviours.

The capability formulas are easier to define than the I/O formulas because capabilities act on ambients, and the logic has a connective, $n[A]$, for talking about ambients. By contrast, the I/O primitives act on themselves. To define the I/O formulas, we proceed as follows:

(1) We define a formula, `TestComm`, that characterises the special abstraction $(x)\, x[\mathbf{0}]$.

(2) We use `TestComm` to define the formula for messages:

$$\mathcal{F}_{\{n\}} \stackrel{\text{def}}{=} \mathtt{1comm} \wedge (\mathtt{TestComm} \rhd \diamond\, n[\mathbf{0}])$$

It holds that $P \models \mathcal{F}_{\{n\}}$ iff $P \equiv \{n\}$.

(3) We then use $\mathcal{F}_{\{n\}}$ to define the formulas for abstractions:

$$\langle\!\langle ?n \rangle\!\rangle. \mathcal{A} \stackrel{\text{def}}{=} \mathtt{1comm} \wedge (\neg \exists\, x \,.\, \mathcal{F}_{\{x\}}) \wedge (\mathcal{F}_{\{n\}} \rhd \diamond\, \mathcal{A})$$

It holds that $P \models \langle\!\langle ?n \rangle\!\rangle. \mathcal{A}$ iff $P \equiv (x)\, Q$ and $\{n\} \mid P \Longrightarrow P'$ with $P' \models \mathcal{A}$.

Lemma 3.10. *Given $(x)\ R$, suppose there is q such that*

$$\{q\} \mid (x)\ R \models \Box(\mathtt{2comm} \vee \mathtt{1amb0})$$

and R contains no abstractions. Then $R \equiv \eta[\mathbf{0}]$, for some η.

We call *ambient abstraction* any closed abstraction described by the following grammar:

$$P ::= (x)\ \eta[\mathbf{0}] \;\big|\; (x)\ (\{\eta\} \mid P)$$

The following lemma shows how to characterise ambient abstractions using formulas.

Lemma 3.11. *Given an abstraction $(x)\ R$, suppose there is q such that*

$$\{q\} \mid (x)\ R \models \Box(\mathtt{2comm} \vee \mathtt{1amb0}) \tag{3.1}$$

and

$$\{q\} \mid (x)\ R \models \diamond\ \mathtt{1amb0}. \tag{3.2}$$

Then $(x)\ R$ is an ambient abstraction.

Proof. By induction on the number of nested abstractions in R. If this number is 0 then by Lemma 3.10 we derive $R \equiv \eta[\mathbf{0}]$.

Suppose the number is greater than 0. From (3.1) and

$$\{q\} \mid (x)\ R \longrightarrow R\{q/x\}$$

we derive

$$R\{q/x\} \models \mathtt{2comm} \vee \mathtt{1amb0}$$

Since R should contain an abstraction, the formula `1amb0` is not satisfied, hence

$$R\{q/x\} \models \mathtt{2comm}$$

Using this, the fact that $R\{q/x\}$ should contain an abstraction, and (3.2) we infer that

$$R\{q/x\} \equiv \{p\} \mid (y)\, Q$$

for some p, y, Q. By induction hypothesis, we deduce that $(y)Q$ is an ambient abstraction. From this, $R\{q/x\}$ is an ambient abstraction too, and this induces that R itself is an ambient abstraction. ◻

We say that an ambient abstraction P is *simple* if $P =_\beta (x)\, x[\mathbf{0}]$, where $=_\beta$ is the least congruence that is closed under the rule

$$\{M\} \mid (x)\, P = P\{^M\!/\!_x\}\,.$$

We recall that the operator $\blacktriangleright$, used in the following lemma, has been introduced at the end of Section 2.

Lemma 3.12. *Suppose $(x)\, Q$ is an ambient abstraction, and*

$$(x)\, Q \models \mathtt{1comm} \blacktriangleright \Diamond\, n[\mathbf{0}]$$
$$(x)\, Q \models \mathtt{1comm} \blacktriangleright \Diamond\, m[\mathbf{0}]$$

with $m \neq n$. Then $(x)\, Q$ is simple.

Proof. From the hypothesis, there are p and q such that

$$\{p\} \mid (x)\, Q \models \Diamond\, n[\mathbf{0}] \quad \text{and}$$
$$\{q\} \mid (x)\, Q \models \Diamond\, m[\mathbf{0}]\,.$$

If $(x)\, Q$ were not simple, then the name of the ambient to which it reduces to would not depend on the argument x. (Note that any ambient abstraction is $=_\beta$ to an abstraction of the form $(x)\, \eta[\mathbf{0}]$, for some x, η. The hypothesis of the lemma implies that $\eta = x$.) □

As hinted above, the key step is the definition of the formula below, which is the characteristic formula of simple ambient abstractions.

$$\begin{aligned}
\mathtt{TestComm} \stackrel{\text{def}}{=}\; & \mathtt{1comm} \\
& \wedge\ \mathtt{1comm} \triangleright \Box(\mathtt{2comm} \vee \mathtt{1amb0}) && (3.3)\\
& \wedge\ \mathtt{1comm} \blacktriangleright \Diamond\, n[\mathbf{0}] && (3.4)\\
& \wedge\ \mathtt{1comm} \blacktriangleright \Diamond\, m[\mathbf{0}] && (3.5)
\end{aligned}$$

where n, m are different names.

Lemma 3.13. $P \models \mathtt{TestComm}$ *iff P is a simple ambient abstraction and is closed.*

Proof. The implication from right to left is easy. We consider the opposite.

Process P must be an I/O, since $P \models \mathtt{1comm}$. Also, P cannot be a message, otherwise it would not satisfy the formula

$$\mathtt{1comm} \triangleright \Box(\mathtt{2comm} \vee \mathtt{1amb0})$$

since a message in parallel with $(x)\, \mathbf{0}$ can reduce to $\mathbf{0}$, which does not satisfy $\mathtt{2comm} \vee \mathtt{1amb0}$.

We conclude that P should be an abstraction, say $(x)\, Q$. Now, from (3.3) and (3.4), we get that there are messages p, q such that

$$\{p\} \mid (x)\, Q \models \Box(\mathtt{2comm} \vee \mathtt{1amb0})$$
$$\{q\} \mid (x)\, Q \models \Diamond\, n[\mathbf{0}]$$

From Lemma 3.11 we infer that $(x)\, Q$ is an ambient abstraction. Moreover, by (3.4), (3.5) and Lemma 3.12, $(x)\, Q$ must be simple. □

Now we are finally in the position of defining the characteristic formula for a message $\{n\}$:

$$\boxed{\begin{array}{rl}\mathcal{F}_{\{n\}} \stackrel{\text{def}}{=} & \texttt{TestComm} \triangleright \Diamond\, n[\mathbf{0}] \\ & \wedge\, \texttt{1comm}\end{array}}$$

and, then, the characteristic formula for a message is

$$\boxed{\texttt{1mess} \stackrel{\text{def}}{=} \exists\, x\, .\, \mathcal{F}_{\{x\}}}$$

Lemma 3.14. *$P \models \mathcal{F}_{\{n\}}$ iff $P \equiv \{n\}$, and $P \models \texttt{1mess}$ iff $P \equiv \{n\}$ for some n.*

Proof. The right to left direction is easy. For the converse, we observe that P must be an I/O, and that P cannot be an abstraction (otherwise, when adding a process satisfying `TestComm`, we could not obtain an ambient). Hence $P \equiv \{m\}$, for some m.

Given a simple ambient abstraction Q, we have that

$$Q \mid \{m\} \models \Diamond\, n[\mathbf{0}] \quad \text{iff} \quad m = n\, .$$

This allows us to deduce that $P \equiv \{n\}$. □

We can now define the two modalities for the input connective:

$$\boxed{\begin{array}{rcl}\langle\!\langle ?n \rangle\!\rangle.\, \mathcal{A} & \stackrel{\text{def}}{=} & \texttt{1comm} \wedge (\neg \exists\, x\, .\, \mathcal{F}_{\{x\}}) \;\wedge (\mathcal{F}_{\{n\}} \triangleright \Diamond\, \mathcal{A}) \\ [\![?n]\!].\, \mathcal{A} & \stackrel{\text{def}}{=} & \langle\!\langle ?n \rangle\!\rangle.\, \top \;\wedge\; \neg \langle\!\langle ?n \rangle\!\rangle.\, \neg \mathcal{A} \\ \texttt{1input} & \stackrel{\text{def}}{=} & \exists x.\; \langle\!\langle ?x \rangle\!\rangle.\, \top\end{array}}$$

Lemma 3.15.

- *$P \models \langle\!\langle ?n \rangle\!\rangle.\, \mathcal{A}$ iff there are P', P'' such that $P \equiv (x)P'$, $(x)P' \mid \{n\} \Longrightarrow P''$, and $P'' \models \mathcal{A}$.*
- *$P \models [\![?n]\!].\, \mathcal{A}$ iff there is P' with $P \equiv (x)P'$, and for all P'' such that $(x)P' \mid \{n\} \Longrightarrow P''$, $P'' \models \mathcal{A}$.*

4. Other intensional properties

As we have just seen, AL can capture several syntactical constructions of the calculus. We now further explore the expressiveness of AL, going beyond the results we have established about capabilities and communications.

We first define a formula ϕ_{fin} that characterises finite terms, using a form of *contextual reasoning*. The same method is applied to derive a formula ©n that characterises the terms containing n as a free name. We then introduce formulas that characterise in a restricted sense *persistent* single terms of the calculus. These formulas will be used in Section 5 to establish characteristic formulas for a sub-calculus of MA.

4.1. **Capturing finiteness.** We now present a formula that is satisfied by all and only the finite processes. Detecting replication seems *a priori* unfeasible in the present version of AL, as it does not provide a recursion operator. We capture the 'finite' character of a term using the fact that a replicated process is *persistent*, i.e., it is always present along the reductions of a term.

The characterisation of finiteness relies on the existence of a scenario which guarantees reachability of $\mathbf{0}$, as expressed by the two following lemmas:

Lemma 4.1. *Let P, Q be two terms such that $P \Longrightarrow Q$. Then P is finite iff Q is finite.*

Proof. By induction over the length of the $\Longrightarrow$ derivation, then induction over the structure of the proof of the $\longrightarrow$ transition. ☐

Lemma 4.2. *P is finite iff there are Q, R, n such that $n[P \mid Q] \mid R \Longrightarrow \mathbf{0}$.*

Proof.
• Let us first assume that P is finite. We prove by induction on the size of P that there exist Q and R such that for any P',

$$n[P \mid P' \mid Q] \mid R \Longrightarrow n[P']$$

The left to right implication can then be obtained using this property with $P' = \mathbf{0}$ and adding $\mathsf{open}\ n$ in parallel with R.

1 For $P = \mathbf{0}$, take $Q = R = \mathbf{0}$.
2 For $P \equiv m[P_1]$, we have by induction Q_1, R_1 such that $n[P_1 \mid P' \mid Q_1] \mid R_1 \Longrightarrow n[P']$ for any P'. Now we set $Q = \mathsf{open}\ m \mid Q_1$ and $R = R_1$. Then it is clear that $n[m[P_1] \mid P' \mid Q] \mid R \Longrightarrow n[P']$ for any P'.
3 For $P \equiv P_1 \mid \ldots \mid P_r$ (with no replicated component), we use the induction hypothesis to obtain Q_i and R_i, and then set $Q = Q_1 \mid \ldots \mid Q_r$, $R = R_1 \mid \ldots \mid R_r$ such that for any P',

$$\begin{aligned} n[P \mid P' \mid Q] \mid R &\Longrightarrow n[P_2 \mid \ldots \mid P_r \mid P' \mid Q_2 \mid \ldots \mid Q_r] \mid R_2 \mid \ldots \mid R_r \\ &\Longrightarrow \ldots \Longrightarrow n[P'] \end{aligned}$$

reasoning inductively on r.
4 For $P \equiv \mathsf{cap}.P_1$, we use the induction hypothesis to get Q_1 and R_1, and we define Q and R according to the shape of cap as follows:
   – $\mathsf{cap} = \mathsf{in}\ m$. Then we set $Q = Q_1$ and $R = m[\mathbf{0}] \mid \mathsf{open}\ m \mid R_1$. Then for any P':

$$\begin{aligned} n[P \mid P' \mid Q] \mid R &\longrightarrow m[n[P_1 \mid P' \mid Q]] \mid \mathsf{open}\ m \mid R_1 \\ &\longrightarrow n[P_1 \mid P' \mid Q_1] \mid R_1 \\ &\Longrightarrow n[P'] \end{aligned}$$

   – $\mathsf{cap} = \mathsf{out}\ m$. We set $Q = \mathsf{in}\ m \mid Q_1$ and $R = m[\mathbf{0}] \mid \mathsf{open}\ m \mid R_1$, so that we can conclude.
   – $\mathsf{cap} = \mathsf{open}\ m$. We set $Q = m[\mathbf{0}] \mid Q_1$ and $R = R_1$.
• For $P \equiv \{m\}$, we set $Q = (x)\mathbf{0}$, and $R = \mathbf{0}$.
• For $P \equiv (x)P_1$: by induction hypothesis applied to $P_1\{n/x\}$, we get Q_1 and R_1; then we set $Q = \{n\} \mid Q_1$ and $R = R_1$.

The first implication is thus established.

• Let us now assume P is not finite. Then for any n, Q, R, $n[P \mid Q] \mid R$ is also infinite, and by the previous lemma, it is also the case for any of its reducts, and hence it cannot reduce to $\mathbf{0}$. $\square$

We can now define:

$$\boxed{\phi_{\mathrm{fin}} \stackrel{\mathrm{def}}{=} \exists x.\big(\top \blacktriangleright (\top \blacktriangleright \Diamond \mathbf{0})@x\big)}$$

Theorem 4.3. *For any P, $P \models \phi_{\mathrm{fin}}$ iff P is finite.*

Proof. From Definition 2.3, $P \models \phi_{\mathrm{fin}}$ holds if there are n, Q, R such that $n[P \mid Q] \mid R \Longrightarrow \mathbf{0}$. We then conclude with Lemma 4.2. $\square$

4.2. **Formula for name occurrence.** Our aim is now to define a formula corresponding to the connective ©n, defined by:

$$P \models ©n \text{ iff } n \in \mathrm{fn}(P)\,.$$

For this, we exploit Lemma 4.4 together with the ability, using the formulas for capabilities, to detect unguarded occurrences of names.

We say that a process P is *flat* if it has no inputs and the only process underneath all capabilities, and inside all ambients of P is $\mathbf{0}$. We say that a process P has an occurrence of name n at top level if $P \equiv \mathsf{cap}.\,P' \mid P''$ with $\mathsf{cap} = \mathsf{in}\ n$, $\mathsf{out}\ n$ or $\mathsf{open}\ n$, $P \equiv n[P'] \mid P''$ or $P \equiv \{n\} \mid P'$.

For the proof of the next lemma, we would also need a more general notion. The *occurrence depth* of a name n in an open term is given by a function $\mathsf{depth}_n : \mathcal{P} \longrightarrow \mathbb{N} \cup \{\infty\}$, stable by $\equiv_{\mathrm{E}}$, inductively defined as follows:

- $\mathsf{depth}_n(\mathbf{0}) = \infty$.
- $\mathsf{depth}_n(n[P_1]) = 0$, and for $n \neq \eta$, $\mathsf{depth}_n(\eta[P_1]) = \mathsf{depth}_n P_1 + 1$.
- $\mathsf{depth}_n((!)P_1 \mid \ldots \mid (!)P_r) = \min_{1 \le i \le r} \mathsf{depth}_n(P_i)$ (here $(!)Q$ stands for Q or $!Q$).
- $\mathsf{depth}_n(\mathsf{cap}.\,P) = \begin{cases} 0 & \text{for } \mathsf{cap} \in \{\mathsf{in}\ n, \mathsf{out}\ n, \mathsf{open}\ n\}, \\ \mathsf{depth}_n(P) + 1 & \text{otherwise.} \end{cases}$
- $\mathsf{depth}_n((x)P) = \mathsf{depth}_n(\downarrow_\beta P) + 1$, where $\downarrow_\beta P$ stands for the smallest term such that $P =_\beta \downarrow_\beta P$
- $\mathsf{depth}_n(\{n\}) = 0$ and $\mathsf{depth}_n(\{\eta\}) = \infty$ for $\eta \neq n$.

Lemma 4.4. *For all P, n, we have $n \in \mathrm{fn}(P)$ iff for any name m, there exist some flat processes Q, R, in which n does not occur free, and a process S with an occurrence of n at top level such that $m[P \mid Q] \mid R \Longrightarrow m[S]$.*

Proof. Note that the property of S having an occurrence of n at top level is equivalent to $\mathsf{depth}_n(S) = 0$. We are now ready to prove the lemma:

- We first consider the implication from left to right. Let us assume that $\mathsf{depth}_n(P)$ is finite. We consider a name m, and prove by induction on $\mathsf{depth}_n(P)$ that there exist Q, R, S satisfying the conditions of the lemma.
  - if $\mathsf{depth}_n(P) = 0$, we take $Q = R = \mathbf{0}$ and $S = P$.

 – if $\mathsf{depth}_n(P) = i+1$, we first consider the case where $P \equiv \mathsf{in}\ m_1.\,P_1 \mid P_2$ with $\mathsf{depth}_n(P_1) = i$. By induction hypothesis, there are Q_1, R_1, S_1 and m satisfying the conditions of the lemma for $P_1 \mid P_2$. We then can set for P: $Q = Q_1$, $R = m_1[\mathbf{0}] \mid \mathsf{open}\ m_1 \mid R_1$ and $S = S_1 \mid P_2$, then Q, R, S can be chosen for P.
 The other cases are treated similarly: we define processes that allow us to trigger a capability in order to decrease the occurrence depth of n in the term. The definition of these processes follows the ideas in the proof of Lemma 4.2.

The first implication is proved.

- For the implication from right to left, we assume that $n \notin \mathrm{fn}(P)$. We consider $m \neq n$, and some Q, R as in the statement of the lemma. Then $n \notin \mathrm{fn}(m[P \mid Q] \mid R)$, so that for any T such that $m[P \mid Q] \mid R \Longrightarrow T$, $n \notin \mathrm{fn}(T)$. □

We can now define the formula ©n to capture the set of free names of a process, together with the two auxiliary formulas flat and ©$^1 n$ needed in the definition of ©n. These formulas are given in Table 4.

$$\begin{array}{lcl}
\mathsf{flat} & \stackrel{\text{def}}{=} & (\exists x.\ [\![\mathsf{in}\ x]\!].\,\mathbf{0}\ \vee\ [\![\mathsf{out}\ x]\!].\,\mathbf{0}\ \vee\ [\![\mathsf{open}\ x]\!].\,\mathbf{0}\ \vee x[\mathbf{0}]\ \vee\ \mathcal{F}_{\{x\}})^{\omega} \\
©^1 n & \stackrel{\text{def}}{=} & \big(\langle\!\langle \mathsf{in}\ n\rangle\!\rangle.\top\ \vee\ \langle\!\langle \mathsf{out}\ n\rangle\!\rangle.\top\ \vee\ \langle\!\langle \mathsf{open}\ n\rangle\!\rangle.\top\ \vee\ n[\top]\ \vee\ \mathcal{F}_{\{n\}}\ \big) \mid \top \\
©n & \stackrel{\text{def}}{=} & \forall x.\ (\mathsf{flat} \wedge \neg ©^1 n) \blacktriangleright \big(\ (\mathsf{flat} \wedge \neg ©^1 n) \blacktriangleright \Diamond\ x[©^1 n]\ \big)@x
\end{array}$$

Table 4: Formulas for free names

Formula ©n detects whether name n occurs in a process, while ©$^1 n$ detects whether n occurs at top level (i.e. P satisfies this formula iff $\mathsf{depth}_n(P) = 0$).

Theorem 4.5 (Name occurrence). *$P \models ©n$ iff $n \in \mathrm{fn}(P)$.*

Proof. Consequence of the previous lemma. □

4.3. **Formulas for persistence.** We now move to the definition of formulas that characterise *persistence*, which is given by the replication operator in MA. In other words, we investigate the possibility of defining formulas $!\mathcal{A}$ that detect replicated term $!P$ such that P satisfies $\mathcal{A}$. However, we cannot hope to define arbitrary formulas with precisely this property. First, the form $!P$ is too restrictive: as $P =_L Q$ implies $!P =_L !P \mid Q$ (see [HLS05]), a formula $!\mathcal{A}$ would not distinguish between a uniquely replicated process $!P$, and a replicated process "with admissible garbage" $!P \mid Q$ or $!P \mid !Q$. Second, if we want to express that the process holds something replicated, one has to reject formulas satisfied by the process $\mathbf{0}$.

We hence restrict our attention to the case of formulas $\mathcal{A}$ whose models are single processes only. For these formulas, $!\mathcal{A}$ characterizes replicated processes, in the sense that

$$P \models !\mathcal{A} \quad \Leftrightarrow \quad \exists P_1, \ldots, P_n \text{ s.t. } \begin{cases} 1)\ P \equiv !P_1 \mid (!)P_2 \mid \ldots \mid (!)P_n \\ 2)\ \forall i \in 1 \ldots n,\ P_i \models \mathcal{A} \end{cases}$$

where (!) denotes an optional replication. In the sequel, we show how to define the formula $!\mathcal{A}$ when $\mathcal{A}$ characterizes a guarded process and has some extra conditions. For the purpose of defining characteristic formulas, this will be sufficient. However, it remains an open question how to define $!\mathcal{A}$ on a larger language.

$$
\begin{array}{lcl}
\mathtt{Rep}_{\mathsf{in}\ n}(\mathcal{A}) & \stackrel{\mathrm{def}}{=} & \mathcal{A}^{\omega} \ \wedge \ \forall m.\ (\neg \copyright m) \ \rightarrow \\
& & (\llbracket \mathsf{out}\ n \rrbracket.\, \mathbf{0})^{\omega} \ \triangleright \ \big(\ n[\mathbf{0}] \ \triangleright \ \Box\Diamond \ (\ n[m[\mathcal{A} \ \mid \ \top]] \ ) \big) @ m \\
\mathtt{Rep}_{\mathsf{out}\ n}(\mathcal{A}) & \stackrel{\mathrm{def}}{=} & \mathcal{A}^{\omega} \ \wedge \ \forall m.\ (\neg \copyright m) \ \rightarrow \\
& & (\llbracket \mathsf{in}\ n \rrbracket.\, \mathbf{0})^{\omega} \ \triangleright \ \big(\ n[\mathbf{0}] \ \triangleright \ \Box \ \Diamond(\ m[\mathcal{A} \ \mid \ \top] \mid n[\mathbf{0}] \ ) \big) @ m \\
\mathtt{Rep}_{\mathsf{open}\ n}(\mathcal{A}) & \stackrel{\mathrm{def}}{=} & \mathcal{A}^{\omega} \ \wedge \ (n[\mathbf{0}])^{\omega} \ \triangleright \ \Box \ (\ \mathcal{A} \ \mid \ \top \ ) \\
\mathtt{Rep}_{n[]}(\mathcal{A}) & \stackrel{\mathrm{def}}{=} & (n[\mathcal{A}])^{\omega} \ \wedge \ (\llbracket \mathsf{open}\ n \rrbracket.\, \mathbf{0})^{\omega} \ \triangleright \ \Box \ (\ n[\mathcal{A}] \ \mid \ \top \ ) \\
\mathcal{F}_{!\{n\}} & \stackrel{\mathrm{def}}{=} & {\mathcal{F}_{\{n\}}}^{\omega} \ \wedge \ \mathtt{TestComm}^{\omega} \ \triangleright \ \Box \ (\ \mathcal{F}_{\{n\}} \ \mid \ \top \ ) \\
\mathtt{Rep}_{input}(\mathcal{A}) & \stackrel{\mathrm{def}}{=} & \mathcal{A}^{\omega} \ \wedge \ \mathtt{1mess}^{\omega} \ \triangleright \ \Box \ (\ \mathcal{A} \ \mid \ \top \ )
\end{array}
$$

Table 5: Formulas for persistent single terms

The definition of $!\mathcal{A}$ has two parts. The first part says that if $P \models !\mathcal{A}$ then all parallel components in P that are single and at top level satisfy $\mathcal{A}$. This is expressed by the formula $\mathcal{A}^{\omega}$. The second part of the definition of $!\mathcal{A}$ addresses persistence, by saying that there are infinitely many processes at top level that satisfy $\mathcal{A}$ in the sense that we may not consume all copies by some finite sequence of reduction. Definitions are given in Table 5: there is one formula for each possible topmost constructor (recall that we are considering a single process).

Formula $\mathcal{F}_{!\{n\}}$ is actually a characteristic formula, since it is satisfied only by the process $!\{n\}$. For this reason, we anticipate the notation $\mathcal{F}_P$ of the characteristic formula of P (see Section 5). For the other formulas, we express the replication of a process satisfying $\mathcal{A}$; the interpretation of these formulas hence relies on the actual meaning of $\mathcal{A}$.

To illustrate the point, consider formula $\mathtt{Rep}_{\mathsf{open}\ n}(\langle\!\langle \mathsf{open}\ n \rangle\!\rangle.\, \top)$. This formula only specifies that any number of capabilities $\mathsf{open}\ n$ should be present at top-level, and thus holds for process $!\mathsf{open}\ n.\, \mathbf{0}$, but also for $\mathsf{open}\ n.\, !\mathsf{open}\ n.\, \mathbf{0}$. On the other hand, $\langle\!\langle \mathsf{open}\ n \rangle\!\rangle.\, \top$ can be replaced by the more discriminating formula $\llbracket \mathsf{open}\ n \rrbracket.\, \mathbf{0}$: then we obtain a formula that only accepts process $!\mathsf{open}\ n.\, \mathbf{0}$.

In light of these observations, we define the following measures on terms:

Definition 4.1 (Sequentiality degree, sd)**.** The sequentiality degree of an open term is defined as follows:

- $\mathsf{sd}(\mathbf{0}) = 0$, $\mathsf{sd}(P \mid Q) = \max\big(\mathsf{sd}(P), \mathsf{sd}(Q)\big)$;
- $\mathsf{sd}(\eta[P]) = \mathsf{sd}(!P) = \mathsf{sd}(P)$;
- $\mathsf{sd}(\mathsf{cap}.\, P) = \mathsf{sd}(P) + 1$.
- $\mathsf{sd}(\{\eta\}) = 1$ and $\mathsf{sd}((x)P) = \mathsf{sd}(\downarrow_{\beta} P) + 1$

Definition 4.2 (Depth degree)**.** The depth degree of a process is given by a function dd from MA processes to natural numbers, inductively defined by:

- $\mathsf{dd}(\mathbf{0}) \ = \ 0$, $\mathsf{dd}(\mathsf{cap}.\, P) = \mathsf{dd}((x)P) = \mathsf{dd}(\{\eta\}) \ = \ 0$;
- $\mathsf{dd}(\eta[P]) \ = \ \mathsf{dd}(P) + 1$;
- $\mathsf{dd}((!)P_1 \mid \ldots \mid (!)P_r) \ = \ \max_{1 \leq i \leq r} \mathsf{dd}(P_i)$.

Lemma 4.6. *For any processes P and Q, $P \equiv Q$ implies $\mathsf{sd}(P) = \mathsf{sd}(Q)$ and $\mathsf{dd}(P) = \mathsf{dd}(Q)$.*

Definition 4.3 (Selective and expressive formulas)**.** A formula is *sequentially* (resp. *depth*) *selective* if all processes satisfying it have the same sequentiality (resp. depth) degree.

For any capability cap (resp. name n) and formula $\mathcal{A}$, $\mathcal{A}$ is cap-*expressive* (resp. n-*expressive, input-expressive*) if all terms satisfying it are of the form $\mathsf{cap}.\,P$ (resp. $n[P]$,$(x)P$).

Example 1. $\langle\!\langle \mathsf{in}\ n\rangle\!\rangle.\, n[\mathbf{0}]$ is in n-expressive but not sequentially selective: it admits both $\mathsf{in}\ n.\, n[\mathbf{0}]$ and $\mathsf{in}\ n.\,(n[\mathbf{0}] \mid \mathsf{open}\ n.\, n[\mathbf{0}])$ as models. On the other hand, $[\![\mathsf{in}\ n]\!].\, n[\mathbf{0}]$ is both sequentially selective and in n-expressive. As we will see below (Subsection 5.2), the combination of $\langle\!\langle\mathsf{cap}\rangle\!\rangle$ and $[\![\mathsf{cap}]\!]$ modalities allows us to define sequentially selective formulas.

These two forms of selectivity are useful for the characterisation of persistence. Indeed, the sequentiality (resp. depth) degree of a single prefixed (resp. ambient) term is strictly decreasing when consuming the prefix (resp. opening the ambient). This property is needed in order to detect the presence of replication at top-level in a process, and interpret the formulas introduced above.

Lemma 4.4. Let P, Q be two terms of MA. If $P \longrightarrow Q$ or $P \overset{\mu}{\to} Q$ for some μ, then $\mathsf{sd}(P) \geq \mathsf{sd}(Q)$.

Proof. The property for $\overset{\mu}{\to}$ follows from the definition of $\mathsf{sd}(P)$. For $P \longrightarrow Q$, one reasons by induction and case analysis (using Lemma 2.3). □

Corollary 4.5. For all cap, if $P \overset{\langle\mathsf{cap}\rangle}{\Longrightarrow} Q$, then $\mathsf{sd}(P) \geq \mathsf{sd}(Q)$.

In the sequel, $\Pi_{1\leq i\leq t} Q_i$ abbreviates $Q_1 \mid \ldots \mid Q_t$.

Lemma 4.7 (Characterisation of replication of single processes)**.**

(1) *Given a capability* cap*, and a sequentially selective and* cap*-expressive formula* $\mathcal{A}$*, define*

$$!\mathcal{A} \overset{\text{def}}{=} \mathsf{Rep}_{\mathsf{cap}}(\mathcal{A}).$$

Then $P \models\, !\mathcal{A}$ *iff there are* $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ *such that*

$$P \equiv \Pi_{1\leq i\leq r} !\mathsf{cap}.\, P_i \mid \Pi_{r+1\leq i\leq s} \mathsf{cap}.\, P_i \qquad \text{and} \qquad \mathsf{cap}.\, P_i \models \mathcal{A} \quad \text{for all } 1 \leq i \leq s.$$

(2) *For any name* n *and depth selective and* n*-expressive formula* $\mathcal{A}$*, define*

$$!\mathcal{A} \overset{\text{def}}{=} \mathsf{Rep}_{n[]}(\mathcal{A}).$$

Then $P \models\, !\mathcal{A}$ *iff there are* $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ *such that*

$$P \equiv \Pi_{1\leq i\leq r} !n[P_i] \mid \Pi_{r+1\leq i\leq s} n[P_i] \qquad \text{and} \qquad n[P_i] \models \mathcal{A} \quad \text{for all } 1 \leq i \leq s.$$

(3) *For any formula* $\mathcal{A}$ *that is sequentially selective and input expressive, define*

$$!\mathcal{A} \overset{\text{def}}{=} \mathsf{Rep}_{input}(\mathcal{A}).$$

Then $P \models\, !\mathcal{A}$ *iff there are* $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ *such that*

$$P \equiv \Pi_{1\leq i\leq r} !(x)P_i \mid \Pi_{r+1\leq i\leq s} (x)P_i \qquad \text{and} \qquad (x)P_i \models \mathcal{A} \quad \text{for all } 1 \leq i \leq s.$$

Proof. Let us examine some cases:

Case 1, $\mathsf{cap} = \mathsf{in}\ n$**.** Assume there exist some terms $P_1, \ldots, P_s$ satisfying the condition expressed in 1. Then the first part of $!\mathcal{A}$ is satisfied, i.e. $P \models \mathcal{A}^{\omega}$.

To establish the second part, we have to show that for any $Q \equiv \mathsf{out}\ n^{\omega}$ (where $\omega \in \mathbb{N}^* \cup \{\infty\}$), any fresh name m, and any term R such that $m[P \mid Q] \mid n[\mathbf{0}] \Longrightarrow R$, there is a further reduction $R \Longrightarrow n[m[R_1 \mid R_2]]$ for some R_1, R_2 such that $R_1 \models \mathcal{A}$, which entails in particular $R_1 \equiv \mathsf{in}\ n.\, R_1'$. Since ambient n does not contain any active process, and since there is no active process at top-level in $m[P \mid Q] \mid n[\mathbf{0}]$, ambient n remains at top-level in all evolutions of this term. Moreover, we have that m is fresh for P and Q; therefore, no ambient may get out of m, so for any reduct R, there exists R' such that either (i) $R \equiv m[R'] \mid n[\mathbf{0}]$, and $P \mid Q \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star} R'$, or (ii) $R \equiv n[m[R']]$, and $P \mid Q \xrightarrow{\mathsf{in}\ n} \xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^\star} R'$. In the first case, because of the shape of P, we may perform one more step of reduction to reach a situation like (ii), and then, since $P \mid Q \xrightarrow{\mathsf{in}\ n} \xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^\star} R'$, there exists R'' such that $R' \equiv {!\mathsf{in}}\ n.\, P_1 \mid R''$. The first implication is thus proved.

Conversely, let us assume that $P \models \mathtt{Rep}_{\mathsf{in}\ n}(\mathcal{A})$. Then according to the first part of the formula, there exist some P_i's satisfying $P \equiv (!)\mathsf{in}\ n.\, P_1 \mid \ldots \mid (!)\mathsf{in}\ n.\, P_r$ and $\mathsf{in}\ n.\, P_i \models \mathcal{A}$. Suppose now by absurd that no component is replicated. We exploit the sequential selectivity hypothesis to obtain a contradiction. Indeed, we have the reduction $m[P \mid (\mathsf{out}\ n)^r] \mid n[\mathbf{0}] \Longrightarrow R = n[m[P_1 \mid \ldots \mid P_r]]$ and R is a term whose sequentiality degree is strictly smaller than $\mathsf{sd}(P)$. Then it is also the case for any of its reducts, and therefore the same reasoning holds for any R_1, R_2 such that $R \Longrightarrow n[m[\mathsf{in}\ n.\, R_1 \mid R_2]]$, $\mathsf{in}\ n.\, R_1$ has a sequentiality degree too small to satisfy $\mathcal{A}$ because of sequential selectivity. Thus, P cannot satisfy $\mathtt{Rep}_{\mathsf{in}\ n}(\mathcal{A})$, and we obtain a contradiction. Hence, at least one of the P_i's is replicated, and the reverse implication is proved.
The proofs for **Case 1**, other capabilities, and **Case 3** follow from similar arguments.
Case 2. Assume that $P \equiv {!n}[P_1] \mid \ldots \mid (!)n[P_r]$, with the P_i's such that $P_i \models \mathcal{A}$. Then P satisfies $\mathtt{Rep}_{n[]}(\mathcal{A})$ iff for any $Q \equiv \mathsf{open}\ n^\omega$, and any R such that $P \mid Q \Longrightarrow R$, there are R_i's such that $R \equiv n[R_1] \mid R_2$ with $n[R_1] \models \mathcal{A}$. Since for any R, $R \equiv {!n}[P_1] \mid R'$, the first implication is established.

Conversely, suppose P satisfies $\mathtt{Rep}_{n[]}(\mathcal{A})$. Then $P \equiv (!)n[P_1] \mid \ldots \mid (!)n[P_r]$. Moreover, if no P_i is replicated, $P \mid {!\mathsf{open}}\ n \Longrightarrow P_1 \mid \ldots \mid P_r \mid {!\mathsf{open}}\ n$, and if in some P_i there are $P_{i,j}$ $(j = 1, 2)$ such that $P_i \equiv n[P_{i,1}] \mid P_{i,2}$, then the depth degree of $P_{i,1}$ is too small for $n[P_{i,1}]$ to satisfy $\mathcal{A}$, which gives us the second implication. $\square$

The formulas for persistence, together with the constructions of Section 3, will be used to derive characteristic formulas with respect to $=_L$ for a sub-calculus of MA in Section 5.

5. Characteristic formulas

In this section we establish the existence of characteristic formulas for a large class of processes. Given a process P, a characteristic formula for P is a formula $\mathcal{F}_P$ such that:

$$\forall Q.\ Q \models \mathcal{F}_P \text{ iff } Q =_L P\,,$$

where $=_L$ is logical equivalence (i.e., $P =_L Q$ iff P and Q satisfy the same formulas).

The definability of characteristic formulas is an interesting property, though for now only a purely theoretical result. The effectiveness and efficiency of the construction of characteristic formula are beyond the scope of this paper, though we strongly believe that our definition gives an algorithm for constructing formulas on the semi-decidable fragment $\mathrm{MA_{IF}}$. Having such constructive characteristic formulas, would have some practical impact, since we could relate the logical equivalence and model-checking problem to the validity problem. Interestingly, we may also recall that validity reduces to model-checking the other way round when the spatial logic considered has the guarantee ($\triangleright$) connective.

To be able to carry out our programme, we have first to understand what $=_L$ represents. For this, we use a co-inductive characterisation of $=_L$, as a form of labelled bisimilarity. Then, making an intensive use of the formulas for the connectives of the calculus previously defined, we derive the characteristic formulas.

5.1. Intensional bisimilarity.

Note for this subsection only. The results presented in this subsection have appeared previously in [San01, HLS02] and therefore are not a contribution of the present paper. Their complete proofs, which are rather long and complex, can be found in a companion paper [HLS05]. We will use the notion of intensional bisimilarity and all the properties that are recalled in this subsection only in the proof about characteristic formulas for AL (Theorem 5.5), which is one of our main expressiveness results.

We use the labelled transitions (Definition 2.2) to define a notion of intensional bisimilarity in order to capture $=_L$.

Definition 5.1. *Intensional bisimilarity* is the largest symmetric relation $\simeq_{\text{int}}$ on closed processes such that $P \simeq_{\text{int}} Q$ implies:

(1) If $P \equiv P_1 \mid P_2$ then there are Q_1, Q_2 such that $Q \equiv Q_1 \mid Q_2$ and $P_i \simeq_{\text{int}} Q_i$, for $i = 1, 2$.
(2) If $P \equiv \mathbf{0}$ then $Q \equiv \mathbf{0}$.
(3) If $P \longrightarrow P'$ then there is Q' such that $Q \Longrightarrow Q'$ and $P' \simeq_{\text{int}} Q'$.
(4) If $P \stackrel{\mathsf{in}\, n}{\to} P'$ then there is Q' such that $Q \stackrel{\mathsf{in}\, n}{\Longrightarrow} \xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^\star} Q'$ and $P' \simeq_{\text{int}} Q'$.
(5) If $P \stackrel{\mathsf{out}\, n}{\longrightarrow} P'$ then there is Q' such that $Q \stackrel{\mathsf{out}\, n}{\Longrightarrow} \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star} Q'$ and $P' \simeq_{\text{int}} Q'$.
(6) If $P \stackrel{\mathsf{open}\, n}{\longrightarrow} P'$ then there is Q' such that $Q \stackrel{\mathsf{open}\, n}{\Longrightarrow} Q'$ and $P' \simeq_{\text{int}} Q'$.
(7) If $P \stackrel{\{n\}}{\to} P'$ then there is Q' such that $Q \stackrel{\{n\}}{\Longrightarrow} Q'$ and $P' \simeq_{\text{int}} Q'$.
(8) If $P \stackrel{?n}{\to} P'$ then there is Q' such that $Q \mid \{n\} \Longrightarrow Q'$ and $P' \simeq_{\text{int}} Q'$.
(9) If $P \equiv n[P']$ then there is Q' such that $Q \equiv n[Q']$ and $P' \simeq_{\text{int}} Q'$.

The definition of $\simeq_{\text{int}}$ has (at least) two intensional clauses, namely (1) and (2), which allow us to observe parallel compositions and the terminated process. These clauses correspond to the intensional connectives '$|$' and '$\mathbf{0}$' of the logic. The clause (8) for abstraction is similar to the input clause of bisimilarity in asynchronous message-passing calculi [ACS98]. This is the case because communication in MA is asynchronous. Another consequence of this is that the logic is insensitive to the following rewrite rule (modulo associativity-commutativity of $|$):

$$(x)\big(\{x\} \mid (x)P\big) \longrightarrow_\eta (x)P\,.$$

This rule induces a notion of normal form of processes, that we shall call the *eta-normalised form*.

Definition 5.2 (Eta-equivalence)**.** We will note $P \equiv_{\text{E}} Q$ if the normal forms of P and Q for $\longrightarrow_\eta$ are related by $\equiv$.

Lemma 5.1 ([HLS02, HLS05])**.** *For any closed process P, Q in MA, $P \equiv_{\text{E}} Q$ implies $P \simeq_{\text{int}} Q$.*

By Theorem 5.2 below, this result says that the logic is insensitive to $\longrightarrow_\eta$. We shall thus reason using normalised processes with respect to $\longrightarrow_\eta$ in the proof of Theorem 5.5.

The most peculiar aspect of the definition of $\simeq_{\text{int}}$ is the use of the stuttering relations. Although they can be avoided on finite processes, they cannot in the full calculus. By contrast, stuttering does not show up in Safe Ambients [LS00], where movements are achieved by means of synchronisations involving a capability and a *co-capability*.

We now state some results about $\simeq_{\text{int}}$ that are proved in [HLS02, HLS05].

Theorem 5.2. *For any P, Q, $P \simeq_{\text{int}} Q$ implies $P =_L Q$.*

The latter result establishes *correctness* of $\simeq_{\text{int}}$ with respect to $=_L$. Given a process P, we try and characterise the equivalence class of P with respect to $\simeq_{\text{int}}$ with a formula $\mathcal{F}_P$. The definability of such a formula will actually entail that $=_L \;\subseteq\; \simeq_{\text{int}}$ (*completeness*), and hence that $\mathcal{F}_P$ actually characterises the $=_L$-equivalence class of P.

We now mention a useful induction principle that allows us to reason *'almost inductively'* on the structure of a process when checking relation $\simeq_{\text{int}}$. This principle is given by the following inductive order:

Definition 5.3. We write $P > Q$ if either $\mathsf{sd}(P) > \mathsf{sd}(Q)$ or Q is a sub-term of P.

This order allows us, using the following result, to derive an inductive characterisation of $\simeq_{\text{int}}$ [HLS02, HLS05].

Proposition 5.3. *Let P, P_1, P_2, Q be processes of MA. Then*

(1) $\mathbf{0} \simeq_{\text{int}} Q$ *iff* $Q \equiv \mathbf{0}$.
(2) $n[P] \simeq_{\text{int}} Q$ *iff there exists* Q' *such that* $Q \equiv n[Q']$ *and* $P \simeq_{\text{int}} Q'$.
(3) $P_1 \mid P_2 \simeq_{\text{int}} Q$ *iff there exist* Q_1, Q_2 *such that* $Q \equiv Q_1 \mid Q_2$ *and* $P_i \simeq_{\text{int}} Q_i$ *for* $i = 1, 2$.
(4) $!P \simeq_{\text{int}} Q$ *iff there exist* $r \geq 1, s \geq r, Q_i$ $(1 \leq i \leq s)$ *such that* $P \simeq_{\text{int}} Q_i$ *for* $i = 1 \ldots s$, *and* $Q \equiv \Pi_{1 \leq i \leq r} !Q_i \mid \Pi_{r+1 \leq i \leq s} Q_i$.
(5) $\mathsf{cap}.\, P \simeq_{\text{int}} Q$ *iff there exists* Q' *such that* $Q \equiv \mathsf{cap}.\, Q'$ *with* $P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} \simeq_{\text{int}} Q'$ *and* $Q' \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} \simeq_{\text{int}} P$.
(6) $\{n\} \simeq_{\text{int}} Q$ *iff* $Q \equiv \{n\}$.
(7) $(x)P \simeq_{\text{int}} Q$ *iff there exists* P', Q', Q'' *and* $n \notin \text{fn}(P) \cup \text{fn}(Q)$ *such that* $Q \equiv (x)Q'$, $Q \mid \{n\} \Longrightarrow Q''$, $\downarrow_\beta P\{^n\!/\!_x\} \simeq_{\text{int}} Q''$, $(x)P \mid \{n\} \Longrightarrow P'$ *and* $P' \simeq_{\text{int}} Q'\{^n\!/\!_x\}$.

5.2. **The sub-calculus $\mathbf{MA}_{\text{IF}}$.** As we mentioned above, characteristic formulas and completeness for an algebraic characterisation of logical equivalence are two related problems. In fact, the existence of characteristic formulas is a stronger result than completeness of $\simeq_{\text{int}}$ with respect to $=_L$: while we establish completeness in [HLS05] on the whole calculus, we are only able to derive characteristic formulas on a sub-calculus of MA. To introduce the necessity of restricting the class of processes we consider, and to illustrate the basic ideas behind the construction of characteristic formulas, we examine some examples.

Example 2. We introduce the following processes: $P_1 = !\mathsf{open}\ n.\, n[\mathbf{0}]$, $P_2 = \mathsf{open}\ n \mid n[\mathbf{0}]$, $P_3 = !\mathsf{open}\ n.\, P_2$, and $P_4 = \mathsf{open}\ n.\, P_2$.

A characteristic formula for P_1 is easy to define since the continuation term $n[\mathbf{0}]$ has no reducts. Hence the formula $[\![\mathsf{open}\ n]\!].\, n[\mathbf{0}]$, using a formula for *necessity*, satisfies the conditions of Lemma 4.7, and a characteristic formula for P_1 is

$$\mathcal{F}_1 \stackrel{\text{def}}{=} ![\![\mathsf{open}\ n]\!].\, n[\mathbf{0}]\,.$$

In order to define a characteristic formula for P_3, we first look for a characteristic formula for P_2. We can set

$$\mathcal{F}_2 \stackrel{\text{def}}{=} [\![\mathsf{open}\ n]\!].\, \mathbf{0} \ \mid\ n[\mathbf{0}]\,.$$

$\mathcal{F}_2$ is indeed a characteristic formula for P_2. However, $[\![\mathsf{open}\ n]\!].\,\mathcal{F}_2$ is not a characteristic formula for P_4, nor for P_3. The reason is that the continuation process (P_2) is not static, as it may reduce to $\mathbf{0}$. Hence $[\![\mathsf{open}\ n]\!].\,\mathcal{F}_2$ does not satisfy the conditions of Lemma 4.7, so that we need to add the possibility to reduce to $\mathbf{0}$, yielding the formula $[\![\mathsf{open}\ n]\!].\,(\mathcal{F}_2 \vee \mathbf{0})$. But then we also accept the term $\mathsf{open}\ n.\,\mathbf{0}$, which shows why we are led to add a *possibility* condition to the formula, and we finally define the following characteristic formula for P_3:

$$\mathcal{F}_3 \stackrel{\text{def}}{=} \mathsf{Rep}_{\mathsf{open}\ n}(\langle\!\langle \mathsf{open}\ n\rangle\!\rangle.\,\mathcal{F}_2 \ \wedge\ [\![\mathsf{open}\ n]\!].\,(\mathcal{F}_2 \ \vee\ \mathbf{0}))\,.$$

We see on this example that characterising the continuation of a process starting with a capability or an input requires to enumerate also all the possible reducts after consuming the topmost constructor. Therefore, the definition of characteristic formulas relies on the actual feasibility of such an enumeration, which leads us to the definition of a subclass of MA processes.

In the definition below, we use the following notation: given a set $\mathcal{S}$ of processes, $\mathcal{S}_{/\simeq_{\text{int}}}$ will stand for the quotient of $\mathcal{S}$ with respect to $\simeq_{\text{int}}$ (which is, technically, a set of $\simeq_{\text{int}}$-equivalence classes of processes).

Definition 5.4 (Sub-calculus $\mathrm{MA}_{\mathrm{IF}}$)**.** A process P is *image-finite* iff any sub-term of P of the form $\mathsf{cap}.\,P'$ (resp. $(x)P'$) is such that the set $\{P'' \ :\ P' \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} P''\}_{/\simeq_{\text{int}}}$ (resp. $\{P'' \ :\ P'\{{}^{n}\!/\!_{x}\} \Longrightarrow P''\}_{/\simeq_{\text{int}}}$, for some $n \notin \mathrm{fn}(P)$) is finite.

$\mathrm{MA}_{\mathrm{IF}}$ is the set of image-finite MA processes.

$\mathrm{MA}_{\mathrm{IF}}$ is only a semi-decidable fragment of MA. A stronger restriction is considered in [HLS05], whose definition involves decidable syntactic conditions. We however stick to this larger fragment for the sake of generality.

For example, process $\mathsf{in}\ n.\,!(n[\mathbf{0}] \mid \mathsf{open}\ n.\,\mathbf{0})$ is in $\mathrm{MA}_{\mathrm{IF}}$, but $\mathsf{in}\ n.\,!(n[\mathbf{0}] \mid \mathsf{open}\ n.\,a[\mathbf{0}])$ is not.

To construct a characteristic formula $\mathcal{F}_P$ for a closed $\mathrm{MA}_{\mathrm{IF}}$ process P, we can suppose (up to $\equiv_{\mathrm{E}}$) that replication only appears above single terms and that P is eta normalised. We then define the characteristic formula $\mathcal{F}_P$ of P by induction using the order of Definition 5.3 (this defines a valid induction by Lemma 4.4). The defining formulas are given in Table 6. Two technical remarks should be made regarding the definition of $\mathcal{F}_{(x)P}$. First, in the disjunction over the quotiented set $\{P' \ :\ P\{{}^{n_x}\!/\!_{x}\}\Longrightarrow P'\}_{/\simeq_{\text{int}}}$, it is intended that we pick a representative in each equivalence class. Second, to avoid reasoning about processes containing free variables (characteristic formulas are defined only for closed processes), we introduce the auxiliary name n_x, that is used as a placeholder for x, to be replaced by x again once the characteristic formula of process P' has been computed (see the defining clause of $\mathcal{F}_{(x)P}$). So $\mathcal{F}_{P'}\{{}^{x}\!/\!_{n_x}\}$ is a slight abuse of notation that denotes the operation consisting of (i) alpha-converting $\mathcal{F}_{P'}$ so that no bound variable is named x, and (ii) textually replacing n_x with x in the resulting formula.

$$
\begin{array}{llllll}
\mathcal{F}_{\mathbf{0}} & \stackrel{\text{def}}{=} \mathbf{0} & \mathcal{F}_{P|Q} & \stackrel{\text{def}}{=} \mathcal{F}_P \mid \mathcal{F}_Q & & \\
\mathcal{F}_{n[P]} & \stackrel{\text{def}}{=} n[\mathcal{F}_P] & \mathcal{F}_{\mathsf{cap}.P} & \stackrel{\text{def}}{=} \langle\!\langle \mathsf{cap} \rangle\!\rangle.\, \mathcal{F}_P \;\wedge\; [\![\mathsf{cap}]\!].\, \bigvee_{\{P',\ P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P'\}_{/\simeq_{\text{int}}}} \mathcal{F}_{P'} & & \\
\mathcal{F}_{!n[P]} & \stackrel{\text{def}}{=} \mathtt{Rep}_{n[]}(\mathcal{F}_P) & \mathcal{F}_{!\mathsf{cap}.P} & \stackrel{\text{def}}{=} \mathtt{Rep}_{\mathsf{cap}}(\mathcal{F}_{\mathsf{cap}.P}) & & \\
\mathcal{F}_{(x)P} & \multicolumn{3}{l}{\stackrel{\text{def}}{=} \exists x.\ \neg\copyright x \;\wedge\; \langle\!\langle ?x \rangle\!\rangle.\, \mathcal{F}_P \;\wedge} & (n_x \notin \mathrm{fn}(P)) & \\
 & \multicolumn{5}{l}{\quad [\![?x]\!]\Big(\big(\mathcal{F}_{\{x\}} \mid (\mathtt{1input} \wedge \neg\copyright x)\big) \;\vee\; \bigvee_{\{P' \colon P\{n_x/x\} \Longrightarrow P'\}_{/\simeq_{\text{int}}}} \mathcal{F}_{P'}\{x/n_x\} \Big)} \\
\mathcal{F}_{\{n\}} & \stackrel{\text{def}}{=} \text{cf. Lemma 3.14} & \mathcal{F}_{!\{n\}} & \stackrel{\text{def}}{=} \text{cf. Table 5} & \mathcal{F}_{!(x)P} & \stackrel{\text{def}}{=} \mathtt{Rep}_{input}(\mathcal{F}_{(x)P})
\end{array}
$$

Table 6: Characteristic formulas in $\mathrm{MA_{IF}}$

Theorem 5.5 (Characteristic formulas for $\mathrm{MA_{IF}}$)**.** For any closed term P, define $\mathcal{F}_P$ according to Table 6. Then

$$Q \models \mathcal{F}_P \qquad \text{iff} \qquad P \simeq_{\text{int}} Q.$$

Proof. The proof is by induction, using the order of Definition 5.3.

- $\mathcal{F}_{\mathbf{0}}$ characterises $\mathbf{0}$: this holds by Proposition 5.3.
- $\mathcal{F}_{\{n\}}$ characterises $\{n\}$ and $\mathcal{F}_{!\{n\}}$ characterises $!\{n\}$: by Lemma 3.14, Lemma 4.7 and Proposition 5.3.
- if $\mathcal{F}_P$ characterises P, then $\mathcal{F}_{n[P]}$ characterises $n[P]$: by Proposition 5.3.
- if $\mathcal{F}_{P_1}$ characterises P_1 and $\mathcal{F}_{P_2}$ characterises P_2, then $\mathcal{F}_{P_1|P_2}$ characterises $P_1 \mid P_2$: by Proposition 5.3.
- Suppose now that for every P' such that $\mathsf{sd}(P') \leq \mathsf{sd}(P)$, $\mathcal{F}_{P'}$ is a characteristic formula for P'. We then have:
  - $\mathcal{F}_{\mathsf{cap}.P}$ characterises $\mathsf{cap}.\,P$.

    By Lemma 4.4, $\mathsf{sd}(P') \leq \mathsf{sd}(P)$ for any P' such that $P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P'$, so $\mathcal{F}_{P'}$ is a characteristic formula for such processes. We examine each of the two implications. In one direction, $\mathsf{cap}.\,P \models \langle\!\langle \mathsf{cap} \rangle\!\rangle.\,\mathcal{F}_P$, and by Lemma 3.8, $\mathsf{cap}.\,P \models [\![\mathsf{cap}]\!].\, \bigvee_{\{P',\ P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P'\}_{/\simeq_{\text{int}}}} \mathcal{F}_{P'}$, so $\mathsf{cap}.\,P \models \mathcal{F}_{\mathsf{cap}.P}$. Conversely, if $Q \models \mathcal{F}_P$, then from $Q \models \langle\!\langle \mathsf{cap} \rangle\!\rangle.\,\mathcal{F}_P$ we deduce the existence of Q', Q'' such that $Q \equiv \mathsf{cap}.\,Q'$, $Q' \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q''$, and $Q'' \models \mathcal{F}_P$. Moreover, from $Q \models [\![\mathsf{cap}]\!].\, \bigvee_{\{P',\ P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P'\}_{/\simeq_{\text{int}}}} \mathcal{F}_{P'}$, we deduce that there is P' such that $P \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} P'$ and $Q' \models F_{P'}$, so $Q' \simeq_{\text{int}} P'$, and by Proposition 5.3, $Q \simeq_{\text{int}} \mathsf{cap}.\,P$.
  - $\mathcal{F}_{(x)P}$ characterises $(x)P$.

    We first prove that $(x)P \models \mathcal{F}_{(x)P}$. We pick n_0 fresh for P. We can apply the induction hypothesis for $P\{n_0/x\}$ and for all of its reducts P'. Then the implication from right to left follows from Lemma 3.15.

    For the other direction, let Q be such that $Q \models \mathcal{F}_{(x)P}$. We assume first that Q is eta normalised. Let n_0 be a name that can be used to satisfy formula $\mathcal{F}_{(x)P}$. Then $n_0 \notin \mathrm{fn}(Q)$, and there are Q', Q'' such that $Q \equiv (x)Q'$, $\{n_0\} \mid (x)Q' \Longrightarrow Q''$, and $Q'' \models \mathcal{F}_{P\{n_0/x\}}$, that is, by hypothesis, $Q'' \simeq_{\text{int}} P\{n_0/x\}$. Moreover, since Q is eta normalised, $Q'\{n_0/x\}$ is not of the form $\{n_0\} \mid (x)R$ with $n_0 \notin \mathrm{fn}((x)R)$, and hence this process does not satisfy the formula $(\mathcal{F}_{\{n_0\}} \mid (\mathtt{1input} \wedge \neg \copyright n_0))$. Therefore, there exists P' such that $P\{n_0/x\} \Longrightarrow P'$ and $Q'\{n_0/x\} \models \mathcal{F}_{P'}$, that is, by induction, $Q'\{n_0/x\} \simeq_{\mathrm{int}} P'$. Using Proposition 5.3, we deduce $Q \simeq_{\mathrm{int}} (x)P$. We consider now the case when Q is not eta normalised. Let Q_0 be the eta normal form of Q. Then by Lemma 5.1 and Theorem 5.2, $Q =_L Q_0$. Since by hypothesis $Q \models \mathcal{F}_{(x)P}$, $Q_0 \models \mathcal{F}_{(x)P}$ and by the previous arguments, $(x)P \simeq_{\mathrm{int}} Q_0$. Finally, by Lemma 5.1, $(x)P \simeq_{\mathrm{int}} Q$.

– $\mathcal{F}_{!\mathsf{cap}.P}$ characterises $!\mathsf{cap}.\, P$ and $\mathcal{F}_{!(x)P}$ characterises $!(x)P$: these results follow from the replication case in Proposition 5.3 and from Lemma 4.7. In particular, the requirements in terms of sequential (or depth) selectiveness, and cap (or n, input) expressiveness are satisfied because the formulas we are using in our constructions are characteristic formulas, which, by induction, satisfy such requirements. □

Corollary 5.4. *On the sub-calculus MA_{IF}, we have $\simeq_{\mathrm{int}} \; = \; =_L$.*
For any closed processes P and Q of MA_{IF}, we have

$$Q \models \mathcal{F}_P \qquad \textit{iff} \qquad P =_L Q\,. \qquad \square$$

6. Extensions of the calculus

In this section, we study extensions of MA with different forms of communication: we first examine the possibility to emit capabilities (in addition to names) in messages, and then consider synchronous communication. We only show how to capture the modifications brought to the language, without porting all the constructions seen in the previous sections. We however believe that our approach would go through without any major modification.

We start by pointing out that Lemmas 3.5,3.7,3.6 about the interpretation of formulas $\langle\!\langle \mathsf{cap} \rangle\!\rangle.\, \mathcal{A}$ hold in the extensions we consider, since their proofs are insensitive to the presence of communication in the calculus.

6.1. Capabilities in messages.

In the original MA calculus [CG98a], messages can also carry *paths of capabilities*. To accommodate this in the grammar of Table 1, all occurrences of η are replaced by M, and the path productions

$$M ::= \mathsf{cap} \mid M_1.\, M_2 \mid \epsilon\,,$$

are added to those for expressions, where ϵ stands for the empty path. Thus a capability can be a path, such as $\mathsf{open}\, n.\, \mathsf{in}\, m.\, \mathsf{open}\, h$. Also, the rules

$$\epsilon.\, P \;\equiv\; P \qquad\qquad (M_1.\, M_2).\, P \;\equiv\; M_1.\, M_2.\, P$$

are added to those of $\equiv$. Since messages can now carry names or capabilities, a type system is introduced [CG99] to avoid run-time errors. We shall assume that all processes are well-typed (according to the basic Ambient types), which means in particular that in the interpretation of a formula of the form $\mathcal{A} \triangleright \mathcal{B}$, processes that are added in parallel are of the right type. Moreover, we will say that the argument of an abstraction $(x)P$ is *of capability type* whenever the typing ensures that capabilities, and not names, can be sent to instantiate x.

Our main focus will be on the characterisation of these new forms of messages. For this, we need a formula `TestCap`, the analogous of the formula `TestComm` of Section 3.3, satisfied by all abstractions that are eta-congruent to $(x)\, m[x.\, \mathbf{0}]$, where m is some fixed name.

We also need a formula $(\!|M|\!)$, for any closed capability M, that identifies those processes that are structurally congruent to $M.\, \mathbf{0}$. We first discuss an example, namely the formula $(\!|\mathsf{in}\, n.\, \mathsf{open}\, m|\!)$. For this, $\langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{open}\ \ m \rangle\!\rangle.\, \mathbf{0}$ is not enough: this formula is satisfied by $\mathsf{in}\, n.\, \mathsf{open}\, m.\, \mathbf{0}$ but also, for instance, by processes such as $\mathsf{in}\, n.\, (\{M\} \mid (x)\, \mathsf{open}\, m)$, which has some additional I/O, or $\mathsf{in}\, n.\, \mathsf{out}\, n.\, \mathsf{in}\, n.\, \mathsf{open}\, m.\, \mathbf{0}$, which stutters. A formula $\mathcal{F}$ for $(\!|\mathsf{in}\, n.\, \mathsf{open}\, m|\!)$ could thus be (the actual definition of $(\!|\mathsf{in}\, n.\, \mathsf{open}\, m|\!)$ will be different; the formula below is easier to read and semantically equivalent):

$$\begin{aligned}
\mathcal{F} \ \stackrel{\mathrm{def}}{=}\ & \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{open}\ \ m \rangle\!\rangle.\, \mathbf{0} \\
& \wedge \neg\, \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \neg\, \mathtt{1comp} \\
& \wedge \neg\, \langle\!\langle \mathsf{in}\, n \rangle\!\rangle.\, \langle\!\langle \mathsf{out}\, n \rangle\!\rangle.\, \top \\
& \wedge \neg\, \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{open}\ \ m \rangle\!\rangle.\, \neg\, \mathbf{0}
\end{aligned}$$

In the definition of $\mathcal{F}$, the second, third and fourth conjuncts take care of the problems with I/O and stuttering mentioned above.

Here is the complete definition of $(\!|M|\!)$ for any path M:

$$\boxed{\begin{aligned}
(\!|\mathsf{open}\, n.\, M|\!) \ &\stackrel{\mathrm{def}}{=}\ \langle\!\langle \mathsf{open}\ \ n \rangle\!\rangle.\, (\!|M|\!) \\
&\qquad \wedge \neg\langle\!\langle \mathsf{open}\ \ n \rangle\!\rangle.\, (\neg\mathtt{1comp} \vee \mathtt{1amb}) \\
(\!|\mathsf{out}\, n.\, M|\!) \ &\stackrel{\mathrm{def}}{=}\ \langle\!\langle \mathsf{out}\ \ n \rangle\!\rangle.\, (\!|M|\!) \\
&\qquad \wedge \neg\langle\!\langle \mathsf{out}\ \ n \rangle\!\rangle.\, (\neg\mathtt{1comp} \vee \mathtt{1amb}) \\
&\qquad \wedge \neg\langle\!\langle \mathsf{out}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{out}\ \ n \rangle\!\rangle.\, (\!|M|\!) \\
(\!|\mathsf{in}\, n.\, M|\!) \ &\stackrel{\mathrm{def}}{=}\ \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, (\!|M|\!) \\
&\qquad \wedge \neg\langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, (\neg\mathtt{1comp} \vee \mathtt{1amb}) \\
&\qquad \wedge \neg\langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{out}\ \ n \rangle\!\rangle.\, \langle\!\langle \mathsf{in}\ \ n \rangle\!\rangle.\, (\!|M|\!) \\
(\!|\epsilon.\, M|\!) \ &\stackrel{\mathrm{def}}{=}\ (\!|M|\!) \\
(\!|\mathbf{0}|\!) \ &\stackrel{\mathrm{def}}{=}\ \mathbf{0}
\end{aligned}}$$

In the definition of $(\!|M|\!)$, sub-formula $\neg\mathtt{1comp}\vee\mathtt{1amb}$ is used to control process reductions, see Lemma 6.1.

We now define `TestCap`:

$$\boxed{\begin{aligned}
\mathtt{TestCap} \ \stackrel{\mathrm{def}}{=}\ & \mathtt{1comm} \\
& \wedge\ \mathtt{1comm} \rhd \Box(\mathtt{2comm} \vee m[\mathtt{1comp}]) \\
& \wedge\ \mathtt{1comm} \blacktriangleright \Diamond\, m[(\!|\mathsf{in}\, n|\!)] \\
& \wedge\ \mathtt{1comm} \blacktriangleright \Diamond\, m[\mathbf{0}] \\
& \text{where } (n, m) \text{ is any pair of different names.}
\end{aligned}}$$

The correctness of this definition is proved along the lines of that of $\mathtt{TestComm}$. The formula $\mathcal{F}_{\{M\}}$, where M is any closed capability, is then

$$\boxed{\mathcal{F}_{\{M\}} \stackrel{\text{def}}{=} \mathtt{1comm} \wedge \Big(\mathtt{TestCap} \rhd \diamond\, m[(\!|M|\!)]\Big)}$$

We now give the key steps that allow us to derive the interpretation of the formulas presented above.

Lemma 6.1. *Suppose $P \longrightarrow P'$. Then $P \models \neg\mathtt{1comp} \vee \mathtt{1amb}$.* □

Lemma 6.2. *Suppose M, P are closed. Then $P \models (\!|M|\!)$ iff $P \equiv M.\mathbf{0}$.*

Proof. By induction on the size of M. If the size is 0 then $M = \mathbf{0}$ and the result follows easily. For the inductive case, we proceed by a case analysis.

- $M = \mathsf{in}\, n.\, N$. We have $P \models \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, (\!|M|\!)$, therefore by Lemma 3.7 $P \equiv \mathsf{in}\, n.\, P'$ for some P' such that $P' \xrightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^\star} P''$ and $P'' \models (\!|M|\!)$.

  However P' cannot stutter, otherwise $P \models \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, \langle\!\langle \mathsf{out}\ n \rangle\!\rangle.\, \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, (\!|M|\!)$. Also, it cannot be $P' \longrightarrow P''' \Longrightarrow P''$ otherwise by Lemma 6.1 $P' \models \neg\mathtt{1comp} \vee \mathtt{1amb}$, hence $P \models \langle\!\langle \mathsf{in}\ n \rangle\!\rangle.\, (\neg\mathtt{1comp} \vee \mathtt{1amb})$.
- $M = \mathsf{in}\, n.\, N$: similar.
- $M = \mathsf{open}\, n.\, N$: similar (without any stuttering phenomenon).
- $M = \epsilon.\, N$. In this case, we also have $P \models (\!|N|\!)$, hence by induction $P \equiv N$, hence $P \equiv M$. □

We now adapt the notion of ambient abstraction, introduced in Section 3.3, in order to define a class of processes that will be used to give the interpretation of formula $\mathtt{TestCap}$.

Definition 6.1 (ambient abstraction and ambient semi-abstraction)**.** The *ambient abstractions* are the subset of processes defined by the following grammar:

$$P ::= (x)\ m[x.\,\mathbf{0}] \;\big|\; (x)\ (\{N\} \mid P)\,.$$

The *ambient semi-abstractions* are the subset of processes defined by the following grammar:

$$P ::= (x)\ m[Q] \;\big|\; (x)\ (\{N\} \mid P)$$

where Q is single.

Lemma 6.3. *Given an abstraction $(x)R$ whose argument is of capability type and R contains no abstractions, suppose there are messages M, N and substitutions $\{\widetilde{L}/\widetilde{z}\}, \{\widetilde{L'}/\widetilde{z}\}$ such that*

$$\{M\} \mid (x)\ (R\{\widetilde{L}/\widetilde{z}\}) \models \Box(\mathtt{2comm} \vee m[\mathtt{1comp}])$$

and

$$\{N\} \mid (x)\ (R\{\widetilde{L'}/\widetilde{z}\}) \models \diamond\, m[\mathtt{1comp}].$$

Then $(x)\, R$ is an ambient semi-abstraction (i.e., $R \equiv m[P]$ where P is single). □

Lemma 6.4. *Given an abstraction $(x)\,R$ whose argument is of capability type, suppose there are messages M, N and substitutions $\{\widetilde{L}/\widetilde{z}\}, \{\widetilde{L'}/\widetilde{z}\}$ such that*

$$\{M\} \mid (x)\,(R\{\widetilde{L}/\widetilde{z}\}) \models \Box(\mathsf{2comm} \vee m[\mathsf{1comp}])$$

and

$$\{N\} \mid (x)\,(R\{\widetilde{L'}/\widetilde{z}\}) \models \Diamond\, m[\mathsf{1comp}].$$

Then $(x)\,R$ is an ambient semi-abstraction.

Proof. By induction on the number of nested abstractions in R. If this number is 0 then use Lemma 6.3.

Suppose the number is greater than 0. From

$$\{M\} \mid (x)\,(R\{\widetilde{L}/\widetilde{z}\}) \longrightarrow R\{\widetilde{L}/\widetilde{z}\}\{M/x\}$$

we derive

$$R\{\widetilde{L}/\widetilde{z}\}\{M/x\} \models \mathsf{2comm} \vee m[\mathsf{1comp}]$$

Since R should contain an abstraction, the formula $m[\mathsf{1comp}]$ is not satisfied, hence

$$R\{\widetilde{L}/\widetilde{z}\}\{M/x\} \models \mathsf{2comm}$$

Using this, the fact that R should contain an abstraction, and the other judgement in the hypothesis of the lemma we infer that

$$R \equiv \{M'\} \mid (y)\,Q$$

for some M', y, Q. This information on R and the judgements in the hypothesis of the lemma imply:

$$\{M'\{\widetilde{L}/\widetilde{z}\}\{M/x\}\} \mid (x)\,(Q\{\widetilde{L}/\widetilde{z}\}\{M/x\}) \models \Box(\mathsf{2comm} \vee m[\mathsf{1comp}])$$

and

$$\{M'\{\widetilde{L}/\widetilde{z}\}\{N/x\}\} \mid (x)\,(Q\{\widetilde{L}/\widetilde{z}\}\{N/x\}) \models \Diamond\, m[\mathsf{1comp}].$$

We can now conclude, using the inductive hypothesis on Q. $\square$

Lemma 6.5. *Suppose $(x)\,R$ is an ambient semi-abstraction, whose argument is of capability type, and suppose there are messages M, N and substitutions $\{\widetilde{L}/\widetilde{z}\}, \{\widetilde{L'}/\widetilde{z}\}$ such that*

$$\{M\} \mid (x)\,(R\{\widetilde{L}/\widetilde{z}\}) \models \Diamond\, m[(\!|\mathsf{in}|\!)n]$$

and

$$\{N\} \mid (x)\,(R\{\widetilde{L'}/\widetilde{z}\}) \models \Diamond\, m[\mathbf{0}].$$

Then $(x)\,R$ is an ambient abstraction.

Proof. By induction on the number of abstractions in R. The case when this number is 0 is easy: if $R \not\equiv x.\,\mathbf{0}$ then R does not satisfy the given formulas.

If the number of abstractions is greater than 0 then $Q \equiv (\{O\} \mid P)$, for some message O and process P and then we derive:

$$\{M\} \mid (x)\,(O\{\widetilde{L}/\widetilde{z}\} \mid (y)\,P\{\widetilde{L}/\widetilde{z}\}) \longrightarrow O\{\widetilde{L}/\widetilde{z}\}\{M/x\} \mid (y)\,P\{\widetilde{L}/\widetilde{z}\}\{M/x\} \models \Diamond\, m[(\!|\mathsf{in}|\!)n]$$

and similarly

$$O\{\widetilde{L'}/\widetilde{z}\}\{N/x\} \mid (y)\,P\{\widetilde{L'}/\widetilde{z}\}\{N/x\} \models \Diamond\, m[\mathbf{0}]$$

and then we conclude using induction. $\square$

We say that an ambient abstraction P is *simple* if $P =_\beta (x)\, m[x.\,\mathbf{0}]$ where $=_\beta$ is the least congruence that is closed under the rule

$$\{M\} \mid (x)\, P = P\{^M\!/\!_x\}\,.$$

Lemma 6.6. *Suppose $(x)\, Q$ is an ambient abstraction, and that we have*

$$(x)\, Q \models \mathtt{1comm} \blacktriangleright \Diamond\, m[(\!|\mathsf{in}|\!)n] \quad \textit{and} \quad (x)\, Q \models \mathtt{1comm} \blacktriangleright \Diamond\, m[\mathbf{0}]\,.$$

Then $(x)\, Q$ is simple.

Proof. Any ambient abstraction is equivalent with respect to $\equiv_{\mathrm{E}}$ (structural equivalence plus the eta law – see Definition 5.2) $(x)\, m[M.\,\mathbf{0}]$. □

Lemma 6.7. $P \models \mathtt{TestCap}$ *iff P is a simple ambient abstraction.*

Proof. We observe that P has to be an I/O, and cannot be a message (otherwise by adding $(x)\,\mathbf{0}$ in parallel with P we could violate the definition of $\mathtt{TestCap}$).

Hence P is an abstraction, and there are M, N such that

$$\begin{array}{rcl} \{M\} \mid P & \models & \Diamond\, m[(\!|\mathsf{in}\, n|\!)] \wedge \Box(\mathtt{2comm} \vee m[\mathtt{1comp}]) \\ \{N\} \mid P & \models & \Diamond\, m[\mathbf{0}] \wedge \Box(\mathtt{2comm} \vee m[\mathtt{1comp}]) \end{array}$$

By Lemma 6.4, P must be an ambient semi-abstraction (note that $\mathbf{0}$ implies $\mathtt{1comp}$). Now Lemma 6.5 shows that P must be an ambient abstraction, which by Lemma 6.6 is simple. □

$$\boxed{\mathcal{F}_{\{M\}} \stackrel{\text{def}}{=} \mathtt{1comm} \wedge \mathtt{TestCap} \rhd \Diamond\, m[(\!|M|\!)]}$$

Lemma 6.8. $P \models \mathcal{F}_{\{M\}}$ *iff* $P \equiv \{M\}$. □

6.2. **Synchronous Ambients.** Since the modal logic does not talk about the I/O primitives, it is interesting to examine variations of these primitives, to see the effect on the equality induced by the logic. In MA communication is asynchronous: since a message has no continuation, no process is blocked until the message is consumed. The most natural variation consists in making communication *synchronous*. For this the production $\{\eta\}$ for messages in the grammar of MA in Table 1 is replaced by the production $\{\eta\}.\,P$. Reduction rule Red-Com becomes:

$$\frac{}{\{M\}.\,Q \mid (x)\, P \longrightarrow Q \mid P\{^M\!/\!_x\}}\ \mathtt{Red\text{-}Com}$$

The communication act liberates, at the same time, both the continuation P of the abstraction *and* the continuation Q of the message. We write $\mathrm{MA}^{\mathrm{sync}}$ for the resulting synchronous calculus.

Synchrony leads to some important modifications in the assertions and in the proofs of the results in the paper. In $\mathrm{MA}^{\mathrm{sync}}$, the eta law fails in the sense that the logic can separate eta equivalent terms (cf. Definition 5.2). Indeed, we will define a formula $\langle\!\langle\{n\}\rangle\!\rangle.\,\mathcal{A}$ whose models are processes $\{n\}.\,P$ with $P \Longrightarrow\ \models \mathcal{A}$. Then, returning to the eta law, formula $\mathtt{1input} \wedge (\langle\!\langle\{n\}\rangle\!\rangle.\, n[\mathbf{0}]) \blacktriangleright \Box\neg\mathtt{3Comp}$ is satisfied by $(x)\big(\{x\} \mid (y)\mathbf{0}\big)$, and not by $(x)\mathbf{0}$, where by $\mathtt{3Comp}$ we mean the formula $\mathtt{1Comp} \mid \mathtt{1Comp} \mid \mathtt{1Comp}$.

We will focus now on the characterisation of this new form of communication. In asynchronous MA, our separation of messages from abstractions exploited their asymmetry:

abstractions, but not messages, have a continuation. In the synchronous case the asymmetry disappears, therefore we have to use a different route for the proof, which makes it a bit more involved.

Again, the most delicate point is to find a replacement for the formula $\mathtt{TestComm}$. We sketch how the new definition is obtained.

- We first define a formula, $\mathtt{OnlyCom}$, that is satisfied only by abstractions $(x)\,P$ and messages $\{M\}.\,P$ in which capability prefixes and ambients do not appear in the continuation P and, moreover, no sub-term of P contains more than two non-trivial parallel components.
- Using $\mathtt{OnlyCom}$ we define a formula, $\mathtt{ComAmb}$, that is satisfied only by processes defined as those that satisfy $\mathtt{OnlyCom}$ except that the innermost operator is an ambient $\eta[x[\mathbf{0}]]$.
- We then define a formula that characterises the abstraction $(x)\;h[x[\mathbf{0}]]$; we write $\mathtt{3comm}$ for $\mathtt{1comm} \mid \mathtt{1comm} \mid \mathtt{1comm}$:

$$\begin{aligned} \mathtt{Imm}\,h \stackrel{\text{def}}{=}\;& \mathtt{ComAmb} \\ & \wedge \mathtt{OnlyCom} \triangleright (\Box\neg\,\mathtt{1comm} \wedge \Box\neg\,\mathtt{3comm}) \\ & \wedge \mathtt{OnlyCom} \blacktriangleright \Diamond\; h[n[\top]] \\ & \wedge \mathtt{OnlyCom} \blacktriangleright \Diamond\; h[m[\top]]\,, \end{aligned}$$

where n and m are different names.

Roughly, the first $\wedge$-component implies that a process that satisfies $\mathtt{Imm}\,h$ has an abstraction or a message as its outermost operator, and an ambient $\eta[x[\mathbf{0}]]$ as the innermost. The second $\wedge$-component, call it $\mathcal{F}$, ensures that the process does not have any other operators; that is, the ambient $\eta[x[\mathbf{0}]]$ is reached immediately after the initial communication. For instance, the process $R \stackrel{\text{def}}{=} \{M\}.\,(x)\;h[x[\mathbf{0}]]$ does not satisfy $\mathcal{F}$ because $R \mid (x)\,\mathbf{0} \Longrightarrow (x)\,h[x[\mathbf{0}]]$ and $(x)\,h[x[\mathbf{0}]]$ satisfies $\mathtt{1comm}$. Finally, the third and fourth $\wedge$-components rule out the messages and the abstraction $(x)\,x[x[\mathbf{0}]]$.

Once we have defined formulas to capture primitives for synchronous communication, the other expressiveness results in the paper also hold for synchronous MA. The corresponding proofs follow closely the arguments in the previous sections.

We now move to the formal definition and analysis of the formulas we alluded to above.

Modifications between Lemma 3.10 and 3.14. To define a formula that captures synchronous outputs (Lemma 6.14 below), we introduce tester processes of the form $(x)h[x[\mathbf{0}]]$, for a given name h. The logical characterisation of these (Lemma 6.13) is slightly more complicated than the corresponding result in the asynchronous case (Lemma 3.13), and is based on four grammars describing communicating processes, that are defined as follows.

$$\mathtt{OnlyCom} \stackrel{\text{def}}{=} \;\; \mathtt{1comm} \blacktriangleright (\Box(\mathtt{2comm} \vee \mathbf{0}) \;\wedge \Diamond\; \mathbf{0})$$

To interpret formula `OnlyCom`, we introduce the following grammars:

$$H ::= \{\eta\}.\, \mathbf{0} \;\big|\; (x)\, \mathbf{0} \;\big|\; \{\eta\}.\, (H \mid H) \;\big|\; (x)\, (H \mid H) \;\big|\; \{\eta\}.\, H \;\big|\; (x)\, H$$

$$K ::= \{\eta'\}.\, \eta[y[\mathbf{0}]] \;\big|\; (x)\, \eta[y[\mathbf{0}]] \;\big|\; \{\eta'\}.\, (H \mid K) \;\big|\; (x)\, (H \mid K) \;\big|\; \{\eta'\}K \;\big|\; (x)\, K$$

$$H^\star ::= H \;\big|\; \mathbf{0}$$

$$K^\star ::= \{\eta'\}.\, \eta[\eta''[\mathbf{0}]] \;\big|\; (x)\, \eta[\eta''[\mathbf{0}]] \;\big|\; \{\eta'\}.\, (H \mid K) \;\big|\; (x)\, (H \mid K) \;\big|\; \{\eta'\}K \;\big|\; (x)\, K \;\big|\; \eta[\eta'[\mathbf{0}]]$$

We write

- $\mathcal{G}H$ for the set of terms described by H,
- $\mathcal{G}K$ for those described by K,
- $\mathcal{G}H^\star$ for those described by $H^\star$, and
- $\mathcal{G}K^\star$ for those described by $K^\star$.

(The grammar for $K^\star$, with respect to that for K, has the additional production for $\eta[\eta'[\mathbf{0}]]$, and has $\eta[\eta'[\mathbf{0}]]$ in place of $\eta[y[\mathbf{0}]]$ in the other productions.)

Lemma 6.9. *Suppose P is a MA^{sync} process. $P\{\eta/x\} \in \mathcal{G}H$ iff $P \in \mathcal{G}H$.* $\square$

Lemma 6.10. *Suppose P is a MA^{sync} process and $P \models$ `OnlyCom`. Then $P \equiv P'$ for some $P' \in \mathcal{G}H$.*

Proof. Suppose $P_1 \models$ `OnlyCom`. Then there is a process P_2 with $P_2 \models$ `1comm` such that

$$P_1 \mid P_2 \models (\Box(\mathtt{2comm} \vee \mathbf{0}) \;\wedge \Diamond\, \mathbf{0}).$$

In particular, it holds that $P_1 \mid P_2 \models$ `2comm`, hence $P_1 \models$ `1comm`.

We show that if Q_1, Q_2 are processes that satisfy `1comm` and such that

$$Q_1 \mid Q_2 \models (\Box(\mathtt{2comm} \vee \mathbf{0}) \;\wedge \Diamond\, \mathbf{0}),$$

then $Q_1, Q_2 \in \mathcal{G}H$.

The proof is by induction on the maximal depth of Q_1, Q_2. The case when this depth is 1 is easy. If this depth is greater than 1, then $Q_1 \mid Q_2 \longrightarrow Q_1' \mid Q_2'$, using the `com` rule, where Q_i is the continuation of Q_i. We have three cases:

- $Q_1' \equiv \mathbf{0}$, $Q_2' \equiv R_1 \mid R_2$ for some non-trivial R_1, R_2;
- the symmetric case;
- none of Q_1' and Q_2' is structurally congruent to $\mathbf{0}$.

In the first two cases, we deduce that R_1, R_2 satisfy `1comm`, and then use induction to infer $R_1, R_2 \in \mathcal{G}H$. Then using the first 4 productions of the grammar, and Lemma 6.9, $Q_1, Q_2 \in \mathcal{G}H$. In the third case, use induction to infer $Q_1', Q_2' \in \mathcal{G}H$. Hence also $Q_1, Q_2 \in \mathcal{G}H$, using the last 2 productions of the grammar and Lemma 6.9. $\square$

Lemma 6.11. *Suppose P is a MA^{sync} process, and $P \models \Box(\mathtt{2comm} \vee h[n[\mathbf{0}]]) \;\wedge \Diamond\, h[n[\mathbf{0}]]$, where h, n are any names. Then $P \equiv P_1 \mid P_2$ with $P_1 \in \mathcal{G}H$ and $P_2 \in \mathcal{G}K^\star$.*

Proof. By induction on the size of P, where the size is the number of operators in P. The size cannot be 0 or 1. If the size is 2 then $P = h[n[\mathbf{0}]]$, and $P \equiv \mathbf{0} \mid h[n[\mathbf{0}]]$, hence the assertion of the lemma, for $P_1 \stackrel{\mathrm{def}}{=} \mathbf{0}$.

Suppose the size if greater than 2. Call

$$\mathcal{F} \;\stackrel{\mathrm{def}}{=}\; \Box(\mathtt{2comm} \vee h[n[\mathbf{0}]]) \;\wedge \Diamond\, h[n[\mathbf{0}]]$$

Then

$$P \equiv P_1 \mid P_2 \quad \text{ where, for } i = 1, 2, \text{we have} \quad P_i \models \mathtt{1comm}$$

Since P must reduce,

$$P_1 \mid P_2 \equiv \{m\}.\, Q_1 \mid (x)\, Q_2$$

and

$$Q_1 \mid Q_2\{m/x\} \models \mathcal{F}\,.$$

The size of $Q_1 \mid Q_2\{m/x\}$ is smaller.

It can be that Q_1 or Q_2 are $\mathbf{0}$, or none is $\mathbf{0}$ (they cannot both be $\mathbf{0}$). In both cases we can conclude by referring to the appropriate grammar productions and by using the inductive hypothesis. $\square$

Define now:

$$\begin{aligned}\texttt{ComAmb} \stackrel{\text{def}}{=} \exists\, x\,.\, \Big(&\ \texttt{OnlyCom} \blacktriangleright \Big(\Box(\texttt{2comm} \vee x[n[\mathbf{0}]])\ \wedge \Diamond\, x[n[\mathbf{0}]]\Big) \\ \wedge &\quad \texttt{OnlyCom} \blacktriangleright \Diamond\, x[m[\mathbf{0}]]\Big)\end{aligned}$$

where n and m are different names.

Lemma 6.12. *Suppose P is a MA$^{\text{sync}}$ process, and $P \models \texttt{ComAmb}$. Then $P \equiv P'$ for some $P' \in \mathcal{G}K$.*

Proof. Suppose $P_1 \models \texttt{ComAmb}$. Then there is a process P_2 and some h with $P_2 \models \texttt{OnlyCom}$ such that $P_1 \mid P_2 \models \Box(\texttt{2comm} \vee h[n[\mathbf{0}]])\ \wedge \Diamond\ h[n[\mathbf{0}]]$. By Lemma 6.10, $P_2 \equiv\in \mathcal{G}H$. By Lemma 6.11, $P_1 \in \mathcal{G}K^\star$. Moreover, since $P_1 \models \texttt{1comm}$, it holds that $P_1 \not\equiv h[n[\mathbf{0}]]$; from this and $P_1 \models \texttt{OnlyCom} \blacktriangleright \Diamond\ h[m[\mathbf{0}]]$, we deduce $P_1 \in \mathcal{G}K$.

Define

$$\begin{aligned}\texttt{Imm}\, h \stackrel{\text{def}}{=}\ & \texttt{ComAmb} \wedge \\ & \texttt{OnlyCom} \triangleright (\Box\neg\, \texttt{1comm} \wedge \Box\neg\, \texttt{3comm}) \\ & \texttt{OnlyCom} \blacktriangleright \Diamond\ h[n[\top]] \\ & \texttt{OnlyCom} \blacktriangleright \Diamond\ h[m[\top]]\end{aligned}$$

where $m \neq n$.

Lemma 6.13. *Suppose $P \models \texttt{Imm}\, h$. Then $P \equiv (x)\ h[x[\mathbf{0}]]$.*

Proof. By Lemma 6.12, $P \equiv P' \in \mathcal{G}K$. One then shows that $(x)\ h[x[\mathbf{0}]]$ satisfies $\texttt{Imm}\, h$, whereas the other terms in $\mathcal{G}K$ do not (choosing the appropriate $\texttt{OnlyCom}$). $\square$

Now we can define the formula:

$$\begin{aligned}\langle\!\langle \{n\} \rangle\!\rangle.\, \mathcal{A} \stackrel{\text{def}}{=}\ & \texttt{1comm} \\ & \wedge \forall\, x\,.\, (\texttt{Imm}\, x \triangleright \Diamond\ (x[n[\mathbf{0}]] \mid \mathcal{A}))\end{aligned}$$

Lemma 6.14. *Suppose P is a MA$^{\text{sync}}$ process. It holds that $P \models \langle\!\langle \{n\} \rangle\!\rangle.\, \mathcal{A}$ iff $P \equiv \{n\}.\, Q$ and $Q \Longrightarrow Q'$ and $Q' \models \mathcal{A}$.*

Proof. Take h fresh. Then by Lemma 6.13,

$$P \mid (x)\ h[x[\mathbf{0}]] \models \Diamond\ (h[x[\mathbf{0}]] \mid \mathcal{A})\,.$$

From this, and $P \models \texttt{1comm}$, we deduce $P \equiv \{m\}.\, P'$, for some m, P'. We also deduce that

$$P' \mid h[m[\mathbf{0}]] \models \Diamond\ (h[m[\mathbf{0}]] \mid \mathcal{A})\,.$$

Since h is fresh, P' cannot interact with h. Hence $m = n$, and moreover $P' \Longrightarrow\models \mathcal{A}$. $\square$

Lemma 6.15. *Suppose P is a MA$^{\text{sync}}$ process. It holds that $P \models \langle\langle ?n \rangle\rangle. A$ iff $P \equiv (x)\ P'$ and $P'\{^n/_x\} \Longrightarrow P'' \models A$.* $\square$

6.3. **Other Extensions.**

6.3.1. *Name restriction and revelation.* Usually [CG98a, CG99], the syntax of MA also has the restriction operator. In [CG01], Cardelli and Gordon propose an extension of AL with logical connectives to describe restriction. In particular, the operator of *name revelation* allows one to derive ©n (Subsection 4.2). In presence of restriction in the calculus, we cannot adapt our construction to capture finiteness of processes, intuitively because our approach consists in exhibiting a context that allows a finite process to reduce to $\mathbf{0}$, which is not possible in general in presence of restriction. However, characteristic formulas can be derived, by enriching our constructions with a formula that says that a process has no restriction (which is definable using name revelation).

6.3.2. *Strong sometimes modality.* One could consider a "strong" version of the sometimes ($\diamond$) modality, where $\longrightarrow$ replaces $\Longrightarrow$ in the definition of $\models$. This variant is easier to study, and less interesting in a sense. We explain the effects it would have. The only drawback is that with a strong version of $\diamond$ we could not derive the formulas of Section 4, and as a consequence characteristic formulas can be given for finite processes only. On the other hand, the formulas for capabilities and communications would become much simpler; we would not have to consider stuttering and eta conversions; logical equivalence would coincide with structural congruence.

6.3.3. *Recursion.* In a different direction, a variant of MA can be considered in which a recursion operator is used instead of replication (see for example [LS03]). Recursion gives trees with infinite depth; this prevents us from defining the measures $\mathsf{sd}(P)$ and $\mathsf{dd}(P)$ up to structural congruence. Moreover, the constructions in Subsection 4.3 are based on the characterisation of persistence (that provides a form of 'recursion in width') of replicated processes. We do not think that they could be easily adapted to a calculus with recursion.

Acknowledgments. We thank the anonymous referees for their careful reading of the paper, and for their comments and suggestions, which resulted in a number of improvements for the paper.

References

[ACS98] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 195:291–324, 1998.

[Car99] L. Cardelli. Semistructured computations. Proc. 7th Intl. workshop on Data Base Programming Languages (DBPL'99), invited talk (accompanying paper available from the author's web page), 1999.

[Car01] L. Cardelli. Describing Semistructured Data. *SIGMOD Record, Database Principles Column*, 30(4), 2001.

[CG98a] L. Cardelli and A.D. Gordon. Mobile ambients. In *Proc. FoSSaCS '98*, volume 1378 of *Lecture Notes in Computer Science*, pages 140–155. Springer Verlag, 1998.

[CG98b] L. Cardelli and A.D. Gordon. Technical annex to [CG98a]. Unpublished notes, 1998.

[CG99] L. Cardelli and A.D. Gordon. Types for mobile ambients. In *Proc. 26th POPL*, pages 79–92. ACM Press, 1999.

[CG00] L. Cardelli and A. Gordon. Anytime, Anywhere, Modal Logics for Mobile Ambients. In *Proc. of 27th POPL*, pages 365–377. ACM Press, 2000.

[CG01] L. Cardelli and A. Gordon. Logical Properties of Name Restriction. In *Proc. of TLCA'01*, volume 2044 of *LNCS*. Springer Verlag, 2001.

[CG04] L. Cardelli and G. Ghelli. A query language for semistructured data based on the ambient logic. *Mathematical Structures in Computer Science, 14(3)*, pages 285–327, 2004.

[CL04] L. Caires and E. Lozes. Elimination of Quantifiers and Undecidability in Spatial Logics for Concurrency. In *Proc. of CONCUR'04*, volume 3170 of *LNCS*, pages 240–257. Springer Verlag, 2004.

[DZ00] S. Dal-Zilio. Structural Congruence for Ambients is Decidable. In *Proc. of ASIAN'00*, volume 1961 of *LNCS*. Springer Verlag, 2000.

[GS86] S. Graf and J. Sifakis. A modal characterization of observational congruence on finite terms of CCS. *Information and Control*, 68:125–145, 1986.

[Hir04] D. Hirschkoff. An Extensional Spatial Logic for Mobile Processes. In *Proc. of CONCUR'04*, volume 3170 of *LNCS*, pages 325–339. Springer Verlag, 2004.

[HLS02] D. Hirschkoff, E. Lozes, and D. Sangiorgi. Separability, Expressiveness and Decidability in the Ambient Logic. In *17th IEEE Symposium on Logic in Computer Science*, pages 423–432. IEEE Computer Society, 2002.

[HLS03] D. Hirschkoff, E. Lozes, and D. Sangiorgi. Minimality Results for the Spatial Logics. In *Proc. of FSTTCS'03*, volume 2914 of *LNCS*, pages 252–264. Springer Verlag, 2003.

[HLS05] D. Hirschkoff, E. Lozes, and D. Sangiorgi. On the separability of the ambient logic. in preparation, 2005.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.

[LS00] F. Levi and D. Sangiorgi. Controlling interference in ambients. Short version appeared in *Proc. 27th POPL*, ACM Press, 2000.

[LS03] F. Levi and D. Sangiorgi. Mobile Safe Ambients. *ACM Trans. Program. Lang. Syst.*, 25(1):1–69, 2003. Short version appeared in *Proc. 27th POPL*, ACM Press.

[Mil99] R. Milner. *Communicating and Mobile Systems: the π-Calculus.* Cambridge University Press, 1999.

[Rey02] J. Reynolds. Separation logic: a logic for shared mutable data structures. In *17th IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2002.

[San01] D. Sangiorgi. Extensionality and Intensionality of the Ambient Logic. In *Proc. of 28th POPL*, pages 4–17. ACM Press, 2001.

[SI94] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110(1):149–163, 1994.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press, 2001.