Separability, Expressiveness, and Decidability in the Ambient Logic*

Daniel Hirschkoff
LIP - ENS Lyon
Daniel.Hirschkoff@ens-lyon.fr

Étienne Lozes
LIP - ENS Lyon
Etienne.Lozes@ens-lyon.fr

Davide Sangiorgi
INRIA
Davide.Sangiorgi@sophia.inria.fr

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.

We study some basic questions concerning the descriptive and discriminating power of AL, focusing on the equivalence on processes induced by the logic ($=_L$). We consider MA, and two Turing complete subsets of it, $\mathrm{MA_{IF}}$ and $\mathrm{MA_{IF}^{syn}}$, respectively defined by imposing a semantic and a syntactic constraint on process prefixes.

The main contributions include: coinductive and inductive operational characterisations of $=_L$; an axiomatisation of $=_L$ on $\mathrm{MA_{IF}^{syn}}$; the construction of characteristic formulas for the processes in $\mathrm{MA_{IF}}$ with respect to $=_L$; the decidability of $=_L$ on $\mathrm{MA_{IF}}$ and on $\mathrm{MA_{IF}^{syn}}$, and its undecidability on MA.

*Work supported by european project FET-Global computing PROFUNDIS

Contents

1 Introduction

The *Ambient Logic*, AL, [CG00] is a modal logic for expressing properties of processes in the calculus of Mobile Ambients, MA [CG98b, 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 $P \stackrel{\text{def}}{=} !a[\mathsf{in}\ c] \mid \mathsf{open}\ a.\, b[\mathbf{0}]$ is represented by the tree:

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

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. AL is a logic for reasoning on such trees. Indeed, the actual definition of satisfaction of the formulas is given on MA processes quotiented by a relation of *structural congruence*, 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 similar-

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

ities 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 0 is satisfied by the terminated process $\mathbf{0}$ (on trees: 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 [CG98c, LS00].)

In this paper we study some basic questions concerning the descriptive and discriminating power of AL. We consider, besides the calculus MA, two subsets of it, obtained by imposing constraints on the processes underneath capabilities. In $\mathrm{MA_{IF}}$, these processes must be image-finite; this is more an ad hoc definition since inhabitance in $\mathrm{MA_{IF}}$is undecidable. In $\mathrm{MA_{IF}^{syn}}$, the same processes must be finite, which is more a syntactic constraint.

Althought they could seem quite arbitrary, these definitions express precisely the constraints needed in some of our results. Both $\mathrm{MA_{IF}}$ and $\mathrm{MA_{IF}^{syn}}$ are Turing complete, and contain processes that are not image-finite.

We describe the main contributions of the paper. We write $=_L$ to indicate the process equivalence induced by the logic, whereby two terms are equated if they satisfy the same sets of formulas. First, we exhibit two operational characterisations of $=_L$ on MA, which do not mention the logic. Characterisations of the equivalence of a logic allow us to understand the notion of equality on processes – a fundamental notion in process calculi – induced by the logic. One characterisation is coinductive, as a form of labelled bisimilarity. The other is inductive, and uses a well-founded measure on the structure of processes.

Second, we prove that $=_L$ coincides with structural congruence on $\mathrm{MA_{IF}^{syn}}$. This gives us an axiomatisation of $=_L$ on $\mathrm{MA_{IF}^{syn}}$. This axiomatic characterisation is false on the larger class $\mathrm{MA}_{\mathrm{IF}}$.

Our third contribution is the construction of characteristic formulas for equivalence classes for $=_L$ in $\mathrm{MA}_{\mathrm{IF}}$. We define, for any process $P \in \mathrm{MA}_{\mathrm{IF}}$, a formula $\mathcal{F}_P$ such that $Q \models \mathcal{F}_P$ holds iff $Q =_L P$, for all $Q \in \mathrm{MA}$. (Although P must be in $\mathrm{MA}_{\mathrm{IF}}$, Q need not be so). The result shows that we can talk about the discriminating power of the logic from within the logic itself, at least under some image-finiteness conditions. A corollary is the undecidability of the model-checking problem on $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ and richer calculi.

Our fourth contribution is on (un)decidability. As a consequence of the inductive characterisation of $=_L$, we can prove that $=_L$ is decidable on $\mathrm{MA}_{\mathrm{IF}}$ and $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$. However, if we drop the image-finiteness conditions of $\mathrm{MA}_{\mathrm{IF}}$, then $=_L$ becomes undecidable. We show this via an encoding of the halting problem of Turing Machines. The encoding of Turing Machines is actually in $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$, which is thus proved to be Turing Complete. This result is not in contradiction with the decidability of $=_L$ in $\mathrm{MA}_{\mathrm{IF}}$ and $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$, because the encoding is correct for reductions but not behaviourally (the process encoding a machine and its derivatives do not need to be in the relation $=_L$).

Most of the results mentioned above are rather different from the usual results of modal logics. Typically, the definition of characteristic formulas exploits fixed-point operators, and the characterised processes are finite-state [GS86, SI94].

AL, by contrast, 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.)

Also, coinductive characterisations of an inductive relation or of the equivalence of a logic usually rely on either image-finiteness of the processes, or on some infinitary operator of the logics, such as infinite conjunctions. In our case we need none of these hypotheses.

Similarly, inductive characterisations of a bisimilarity are usually formulated by means of an infinite intersection of the stratification of the bisimilarity, and require image-finiteness on processes. By contrast, our inductive characterisation is defined by induction on the depth of nesting of operators in a process, and does not require image-finiteness. Finally, in process calculi decidability is usually unrelated to image-finiteness: for instance, the transition relation of the π-calculus is image-finite, yet strong bisimilarity is undecidable [SW01].

Also the actual form of image-finiteness that we use is non-standard. Behavioural equivalence in MA is insensitive to *stuttering* phenomena, originated by processes that may repeatedly enter and exit an ambient. As a consequence, a computation in which all visible actions are stuttering is semantically equivalent to an internal reduction.

In the proofs of the results two groups of technical lemmas are important. The first group is about the construction of formulas for describing all forms of labels of the trees of MA. The formulas for the replicated labels give us (some of) the power of the ! operator ('of course') of linear logic; this was somehow unexpected, because AL has no infinitary operators, or operators that talk about resources with infinite multiplicity. (We obtain only *some* of this power, because we have to impose constraints on the replicated formulas.)

Other useful formulas that we have derived are the following: a formula ϕ_{fin} that characterises the ephemeral processes (that is, $P \models \phi_{\text{fin}}$ iff the tree of P has no replicated labels); for any set $\mathcal{S}$ of names, a formula $\mathsf{refers}\mathcal{S}$ that characterises the processes whose set of free names is precisely $\mathcal{S}$.

There are strong connections among all the results discussed above. For instance, both the characterisations of $=_L$ and the characteristic formulas talk about the separability power of AL. The connections are explicit in the proofs: for instance, the proof of undecidability relies on most of the other results.

Related work. Characterisations and axiomatisations of $=_L$ have already been presented in [San01], on *finite* MA (without replications). The proofs rely on the ephemeral nature of the processes, precisely on the property that all (complete) computations of a process, comprising its interactions with the environment, are finite and terminate with the $\mathbf{0}$ process. Therefore we could not adapt these proofs to processes with replications. The need for stuttering in MA is already pointed out in [San01], but all examples use trees with unbounded depth.

The axiomatic characterisation gives us an axiomatisation of $=_L$ on a class of non-finite MA processes. We are not aware of other axiomatisations of semantic equivalences (defined by operational, denotational, logical, or other means) in non-finite higher-order process calculi; (the only other axiomatisation of higher-order process calculi we are aware of is that of [San01], and is on *finite* MA. Here, we show that stuttering is also necessary on trees with finite depth (the trees with persistencies). Formulas in AL, or similar logics, that characterise the free names of processes were known [CG01b, CC01], but use additional operators (notably the *revelation* operator). The undecidability of the model-checking problem of AL – in fact of an even smaller logic – had already been established, using different techniques [CT01].

2 Background

In this section we set up the general framework of this paper. We introduce the language of Mobile Ambients and the Ambient Logic, and recall some known results that will be useful for our purposes.

2.1 The Mobile Ambient Calculus

We recall here the syntax of 'pure' MA, from [CG98b]. 'Pure' means that computation is only movement. Indeed, we work in a calculus without communication; this extension will be discussed in Section 7. Also, as in [CG00, Car01, CG01a], the calculus has no restriction operator for creating new names. The restriction-free calculus is simpler, and has a more direct correspondence with edge-labelled trees and semistructured data. Table 1 presents the syntax.

$a, b, \ldots, n, m$			*Names*				*Processes*
			Capabilities	P, Q, R	$::=$	$\mathbf{0}$	*(nil)*
cap	$::=$	$\mathsf{in}\ n$	*(enter)*		$\mid$	$P \mid Q$	*(parallel)*
	$\mid$	$\mathsf{out}\ n$	*(exit)*		$\mid$	$\mathsf{cap}.\, P$	*(prefixing)*
	$\mid$	$\mathsf{open}\ n$	*(open)*		$\mid$	$n[P]$	*(ambient)*
					$\mid$	$!P$	*(replication)*

Table 1: The syntax of MA

Names are ranged over by $a, b, \ldots, n, m, \ldots$, capabilities by cap, and processes by P, Q, R, S. Processes with the same internal structure are identified.

$$\frac{}{\mathsf{open}\ n.\, P \mid n[Q] \longrightarrow P \mid Q}\ \texttt{Red-Open}$$

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

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

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}\ \texttt{Red-Par} \qquad \frac{P \longrightarrow P'}{n[P] \longrightarrow n[P']}\ \texttt{Red-Amb}$$

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

Table 2: The rules for reduction

This is expressed by means of the *structural congruence relation*, $\equiv$, the smallest congruence such that $(\mathbf{0}, |, !)$ is a multiset algebra, that is, satisfies the following rules:

$$\begin{array}{cccc} P \mid \mathbf{0} \ \equiv\ P & P \mid Q \ \equiv\ Q \mid P & P \mid (Q \mid R) \ \equiv\ (P \mid Q) \mid R & \\ !P \ \equiv\ !P \mid P & !(P \mid Q) \ \equiv\ !P \mid !Q & !!P \ \equiv\ !P & !\mathbf{0} \ \equiv\ \mathbf{0} \end{array}$$

Theorem 2.1 ([DZ00]) $\equiv$ *is decidable for the calculus we study.*

[DZ00] actually establishes this result decidable for a richer calculus.

The rules for the reduction relation, $\longrightarrow$, are given in Table 2. The reflexive and transitive closure of $\longrightarrow$ is written $\Longrightarrow$.

Definition 2.2 (Labelled transitions and stuttering) *Let P be a process. We introduce the following notations:*

- $P \stackrel{\mathsf{cap}}{\to} P'$, *if* $P \equiv \mathsf{cap}.\, P_1 \mid P_2$ *and* $P' = P_1 \mid P_2$.
- **(stuttering)** $P \xRightarrow{(M_1,M_2)^*} P'$ *if there are* $r \geq 1$ *and processes* $P_1, \ldots, P_r$ *with* $P = P_1$ *such that* $P_i \Longrightarrow \stackrel{M_1}{\to} \Longrightarrow \stackrel{M_2}{\to} \Longrightarrow P_{i+1}$ *for all* $1 \leq i < r$, *and* $P_r \Longrightarrow P'$.
- *Finally,* $\stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow}$ *is a convenient notation for compacting statements involving capability transitions.* $\stackrel{\langle \mathsf{in}\ n \rangle}{\Longrightarrow}$ *is* $\xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^*}$; *similarly* $\stackrel{\langle \mathsf{out}\ n \rangle}{\Longrightarrow}$ *is* $\xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^*}$; *and* $\stackrel{\langle \mathsf{open}\ n \rangle}{\Longrightarrow}$ *is* $\Longrightarrow$.

We shall need two forms of stuttering: $P \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^*} P'$ and $P \xRightarrow{(\mathsf{out}\ n, \mathsf{in}\ n)^*} P'$.

Some of our results are proved by induction on the *sequentiality degree* of a process, which is the maximal depth of nesting of capabilities in the process.

Definition 2.3 (Sequentiality degree, ds) *The sequentiality degree of a term is defined as follows:*

- $\mathsf{ds}(\mathbf{0}) = 0$, $\mathsf{ds}(P \mid Q) = \max\big(\mathsf{ds}(P), \mathsf{ds}(Q)\big)$;
- $\mathsf{ds}(n[P]) = \mathsf{ds}(!P) = \mathsf{ds}(P)$;
- $\mathsf{ds}(\mathsf{cap}.\, P) = 1 + \mathsf{ds}(P)$.

Note that this definition relies on the presence of the ! operator (instead of a recursion operator) in the calculus. An important property of $\mathsf{ds}(P)$ is the following:

Lemma 2.4 *Let P, Q be two terms of MA. Then:*

1. *if* $P \equiv Q$, *then* $\mathsf{ds}(P) = \mathsf{ds}(Q)$
2. *if* $P \longrightarrow Q$ *or* $P \stackrel{\mathsf{cap}}{\to} Q$ *then* $\mathsf{ds}(P) \geq \mathsf{ds}(Q)$.

Proof: 1 is immediate, as is the reult on $\stackrel{\mathsf{cap}}{\to}$ in 2. For $P \longrightarrow Q$, one mayb reason by induction on the inference of $P \longrightarrow Q$. $\square$

Property 2 will be used to state support an important inductive definition below (see Def. 3.9). Another measure on terms, the *depth degree,* will be useful when building characteristic formulas for the logic, in Section 5.1:

Definition 2.5 (Depth degree) *The depth degree of a process is given by a function* dd *from MA processes to natural numbers, inductively defined by:*

- $\mathsf{dd}(\mathbf{0}) \stackrel{\text{def}}{=} 0$, $\mathsf{dd}(\mathsf{cap}.\, P) \stackrel{\text{def}}{=} 0$;
- $\mathsf{dd}(n[P]) \stackrel{\text{def}}{=} \mathsf{dd}(P) + 1$;

- $\mathsf{dd}((!)P_1 \mid \ldots \mid (!)P_r) \stackrel{\text{def}}{=} \max_{1\leq i\leq r} \mathsf{dd}(P_i)$.

Definition 2.6 (Finite and single terms)
- *A term P is* finite *iff there is some P' with no occurrence of the replication operator such that $P \equiv P'$.*
- *A term 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.*

[San01] focuses on finite terms, with goals similar to ours.

2.2 The logic

To define the set of formulas of the Ambient Logic (AL–Table 3) we introduce an infinite set of *variables*, ranged over with x, y, z; η ranges over names and variables.

$\mathcal{A}$	$::=$	$\top$	*(true)*	classical logic
	$\mid$	$\neg\mathcal{A}$	*(negation)*	
	$\mid$	$\mathcal{A} \vee \mathcal{B}$	*(disjunction)*	
	$\mid$	$\forall\, x \,.\, \mathcal{A}$	*(universal quantification over names)*	
	$\mid$	$\diamond\mathcal{A}$	*(sometime)*	temporal and spatial connectives
	$\mid$	0	*(void)*	
	$\mid$	$\eta[\mathcal{A}]$	*(edge)*	
	$\mid$	$\mathcal{A} \mid \mathcal{B}$	*(composition)*	
	$\mid$	$\mathcal{A}@\eta$	*(localisation?!?)*	logical adjuncts
	$\mid$	$\mathcal{A} \triangleright \mathcal{B}$	*(linear implication)*	

Table 3: The syntax of logical formulas

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 briefly discussed in the Introduction. The spatial connectives, 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 $\eta[\mathcal{A}]$, in the sense of being, roughly, their inverse (see below). $\mathcal{A}\{n/x\}$ is the formula obtained from $\mathcal{A}$ by substituting variable x by name n. A formula without free variables is *closed*.

Definition 2.7 (Satisfaction) *The* satisfaction *relation is defined on closed formulas as follows:*

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

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 $\mathcal{A}$. The addition of this connective would not change the results in the paper.

Lemma 2.8 ([CG00]) *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 ($\mathcal{A} \rightarrow \mathcal{B}$).

$$
\begin{array}{rcl@{\qquad}rcl}
\mathcal{A} \wedge B & \stackrel{\text{def}}{=} & \neg(\neg \mathcal{A} \vee \neg \mathcal{B}) & \Box \mathcal{A} & \stackrel{\text{def}}{=} & \neg \Diamond \neg \mathcal{A} \\
\mathcal{A} \rightarrow \mathcal{B} & \stackrel{\text{def}}{=} & \neg \mathcal{A} \vee \mathcal{B} & \mathcal{A} \| \mathcal{B} & \stackrel{\text{def}}{=} & \neg(\neg \mathcal{A} \mid \neg \mathcal{B}) \\
\exists\, x \,.\, \mathcal{A} & \stackrel{\text{def}}{=} & \neg \forall\, x \,.\, \neg \mathcal{A} & \mathcal{A} \blacktriangleright \mathcal{B} & \stackrel{\text{def}}{=} & \neg(\mathcal{A} \triangleright \neg \mathcal{B}) \\
\bot & \stackrel{\text{def}}{=} & \neg \top & & &
\end{array}
$$

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 $P \Longrightarrow P'$. The formula $\mathcal{A}^{\forall} \stackrel{\text{def}}{=} \mathcal{A} \,\|\, \neg \top$, from [CG00], is satisfied by P iff for any Q, R such that $P \equiv Q \mid R$, it holds that $Q \models \mathcal{A}$.

The formula $\mathsf{1Comp} \stackrel{\text{def}}{=} 0 \| 0 \ \wedge\ \neg 0$ from [San01] is satisfied by the single processes (see Def. 2.6).

Let us now define the relation $=_L$ we study in this paper.

Definition 2.9 (Process logical equivalence) *For processes P and Q, we write $P =_L Q$ if for all closed formulas $\mathcal{A}$ it holds that $P \models \mathcal{A}$ iff $Q \models \mathcal{A}$.*

Theorem 2.10 ([San01]) *$=_L$ coincides with $\equiv$ for finite terms.*

As a consequence of this theorem and of Thm. 2.1, $=_L$ is decidable on finite terms. The remainder of this paper is devoted to the extension and further development of this result.

3 Coinductive and inductive operational relations

We now introduce, drawing inspiration from the work on finite terms reported in [San01], a coinductively defined relation $\simeq_{\text{bis}}$ on processes that will be proved to coincide with $=_L$ in Sec. 5. We also provide an inductive characterisation of $\simeq_{\text{bis}}$, by exploiting the intensionality of this relation.

3.1 Intensional bisimilarity and its soundness

Definition 3.1 (Intensional bisimilarity) Intensional bisimilarity *is the largest symmetric relation $\simeq_{\text{bis}}$ on processes such that $P \simeq_{\text{bis}} Q$ implies:*

1. *If $P \equiv P_1 \mid P_2$ then there exist Q_1, Q_2 such that $Q \equiv Q_1 \mid Q_2$ and $P_i \simeq_{\text{bis}} Q_i$, for $i = 1, 2$.*
2. *If $P \equiv \mathbf{0}$ then $Q \equiv \mathbf{0}$.*
3. *If $P \longrightarrow P'$ then there exists Q' such that $Q \Longrightarrow Q'$ and $P' \simeq_{\text{bis}} Q'$.*
4. *For any cap, if $P \stackrel{\mathsf{cap}}{\to} P'$ then there exists Q' such that $Q \stackrel{\mathsf{cap}}{\to} \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q'$ and $P' \simeq_{\text{bis}} Q'$.*
5. *If $P \equiv n[P']$ then there exists Q' such that $Q \equiv n[Q']$ and $P' \simeq_{\text{bis}} Q'$.*

With respect to standard bisimilarities for process calculi, $\simeq_{\text{bis}}$ has intensional clauses, namely (1), (2) and (5), which allow us to observe parallel compositions, the terminated process, and ambients. These clauses correspond to the intensional connectives '$|$', '0', and $n[\mathcal{A}]$ of the logic. The other main peculiarity of $\simeq_{\text{bis}}$ are the stuttering relations. The need for stuttering on infinite trees (i.e., MA with a recursion operator) has been pointed out in [San01]. We show that stuttering is also needed on finite trees with replication.

Example 1 (Stuttering for persistent terms) *Consider the processes*

$$P_0 \stackrel{\text{def}}{=} !\mathsf{open}\ n.\,\mathsf{in}\ n.\,\mathsf{out}\ n.\,\mathsf{in}\ n.\,\mathsf{out}\ n.\,n[\mathbf{0}] \mid n[\mathbf{0}]$$

and

$$P_1 \stackrel{\text{def}}{=} !\mathsf{open}\ n.\,\mathsf{in}\ n.\,\mathsf{out}\ n.\,\mathsf{in}\ n.\,\mathsf{out}\ n.\,n[\mathbf{0}] \mid \mathsf{in}\ n.\,\mathsf{out}\ n.\,n[\mathbf{0}]\,.$$

It holds that $P_0 \not\simeq_{\text{bis}} P_1$; however, since

$$P_0 \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^*} P_1 \xRightarrow{(\mathsf{in}\ n, \mathsf{out}\ n)^*} P_0\,,$$

we have $\mathsf{out}\ n.\,P_0 \ \simeq_{\rm bis}\ \mathsf{out}\ n.\,P_1$. *Without stuttering this equivalence would not hold.*

The proof of congruence of $\simeq_{\rm bis}$ follows the proof of the analogous result in [San01], using a technique similar to Howe's for proving congruence of bisimilarity in higher-order languages [How96].

Lemma 3.2 $\simeq_{\rm bis}$ *is a congruence relation on MA.*

Proof: See [San01]. □

Theorem 3.3 (Soundness of $\simeq_{\rm bis}$) *In MA,* $\simeq_{\rm bis} \subseteq =_L$.

Completeness – the hard implication – is proved in Section 5.3.

We now present an equivalent formulation of $\simeq_{\rm bis}$, which will be useful at some point for technical reasons.

Proposition 3.4 (Reformulation) *Let* $\simeq_{\rm bis'}$ *be the relation defined as* $\simeq_{\rm bis}$ *(Def. 3.1), except that condition 3 is omitted and condition 4 is replaced by:*

- *4' For any* cap, *if* $P = \mathsf{cap}.\,P'$, *then there are* Q_1, Q' *such that* $Q \equiv \mathsf{cap}.\,Q_1$ *and* $Q_1 \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} Q'$ *with* $P' \simeq_{\rm bis} Q'$.

Then $\simeq_{\rm bis'} = \simeq_{\rm bis}$.

Proof:[sketch] It is enough to remark that under condition 1, 4 and 4' are equivalent. Moreover, clause 3 can be derived by induction on the proof of $P \longrightarrow Q$ using clauses 1,4 and 5. A more detailed justification can be found in [San01]. □

Lemma 3.5 (Inversion results for $\simeq_{\rm bis}$) *Let* P, P_1, P_2, Q *be processes of MA. Then*

1. $\mathbf{0} \ \simeq_{\rm bis}\ Q$ *iff* $Q \equiv \mathbf{0}$.
2. $n[P] \ \simeq_{\rm bis}\ Q$ *iff there exists* Q' *such that* $Q \ \equiv\ n[Q']$ *and* $P \ \simeq_{\rm bis}\ Q'$.
3. $P_1 \mid P_2 \ \simeq_{\rm bis}\ Q$ *iff there exist* Q_1, Q_2 *such that* $Q \ \equiv\ Q_1 \mid Q_2$ *and* $P_i \ \simeq_{\rm bis}\ Q_i$ *for* $i = 1, 2$.
4. $!P \ \simeq_{\rm bis}\ Q$ *iff there exist* $r \geq 1, s \geq r, Q_i$ $(1 \leq i \leq s)$ *such that* $Q \equiv \Pi_{1\leq i\leq r} !Q_i \mid \Pi_{r+1\leq i\leq s} Q_i$, *and* $P \ \simeq_{\rm bis}\ Q_i$ *for* $i = 1 \ldots s$.
5. $\mathsf{cap}.\,P \ \simeq_{\rm bis}\ Q$ *iff there exists* Q' *such that* $Q \ \equiv\ \mathsf{cap}.\,Q'$ *with* $P \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\rm bis}\ Q'$ *and* $Q' \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\rm bis}\ P$.

Proof: 1,2,3 are direct consequences of Definition 3.1 and the congruence property. Let us now prove 5; we start with the direct implication. First, applying 1 and 3, we may see that Q is either an ambient or a capability. Moreover, the case of the ambient is also excluded by 2, so $Q \equiv \mathsf{cap}'.Q'$ for some cap', Q'. Now according to condition 4 in Definition 3.1, $\mathsf{cap} = \mathsf{cap}'$ and $P \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\text{bis}} Q' \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\text{bis}} P$. The reverse implication is trivial.

We now turn to the proof of 4; we first establish the result for P single, starting with the direct implication. For any single Q_1 such that $Q \equiv Q_1 \mid Q_2$, we have $Q_1 \simeq_{\text{bis}} P$ (because of 1 and 2). Now this proves that there exist $r \geq 0, s \geq r, Q_i$ $(1 \leq i \leq s)$ such that $Q \equiv \Pi_{1\leq i\leq r}!Q_i \mid \Pi_{r+1\leq i\leq s}Q_i$, and $P \simeq_{\text{bis}} Q_i$ for $i = 1 \dots s$. Let us suppose by absurd that $r = 0$; then applying $s+1$ times the inversion 3, we would get a contradiction. Hence $r > 0$ and the direct implication is proved. To prove the reverse implication, it is enough to see that $!P \equiv \Pi_{1\leq i\leq r}!P \mid \Pi_{r+1\leq i\leq s}P$ and use 3, and conclude with the fact that $P \simeq_{\text{bis}} Q_i$ implies $!P \simeq_{\text{bis}} !Q_i$ (by congruence).

Now suppose P is not single; by applying the structural congruence rule $!!T \equiv !T$, we can rewrite $!P$ in such a way that we can suppose without loss of generality that P consists in a parallel composition of finitely many non structurally congruent single terms (to be exact, we also need to use the law $!T \mid T \equiv !T$ to work with distinct structural equivalence classes). So P may be written modulo $\equiv$ as $P_1 \mid \dots \mid P_k$.

We treat here the case $k = 2$, i.e. $P \equiv P' \mid P''$, the generic case being a direct generalisation. Since P' and P'' are single, we have by 3 and by the result we have just proved the existence of Q', Q'', and of two families $(Q_1^{(1)}, Q_2^{(1)}, \dots, Q_{r_1}^{(1)}, \dots, Q_{s_1}^{(1)})$ and $(Q_1^{(2)}, Q_2^{(2)}, \dots, Q_{r_1}^{(2)}, \dots, Q_{s_1}^{(2)})$ of terms such that

$$Q \equiv Q' \mid Q'', \quad Q' \equiv \Pi_{1\leq i\leq r_1}!Q_i^{(1)} \mid \Pi_{r_1+1\leq i\leq s_1}Q_i^{(1)}, \quad Q'' \equiv \Pi_{1\leq i\leq r_2}!Q_i^{(2)} \mid \Pi_{r_2+1\leq i\leq s_2}Q_i^{(2)},$$
$$P' \simeq_{\text{bis}} Q_i^{(1)} \text{ for } i = 1, \dots, s_1 \quad \text{and} \quad P'' \simeq_{\text{bis}} Q_i^{(2)} \text{ for } i = 1, \dots, s_2 .$$

Now suppose for instance that $r_1 \geq r_2$ and $s_1 \leq s_2$ (this is just an example, the other cases are treated similarly). We set:

$$\begin{array}{rcll} Q_i & = & Q_i^{(1)} \mid Q_i^{(2)} & \text{for } 1 \leq i \leq r_2 \\ & & Q_i^{(1)} \mid Q_{r_2}^{(2)} & \text{for } r_2 < i \leq r_1 \\ & & Q_i^{(1)} \mid Q_i^{(2)} & \text{for } r_1 < i \leq s_1 \\ & & Q_{r_1}^{(1)} \mid Q_i^{(2)} & \text{for } s_1 < i \leq s_2 . \end{array}$$

(note that the above decomposition

$$[1 \dots s_2] = [1 \dots r_2] \cup [r_2 + 1 \dots r_1] \cup [r_1 + 1 \dots s_1] \cup [s_1 + 1 \dots s_2]$$

may contain empty intervals). By construction, and by congruence of $\simeq_{\text{bis}}$, we have that for all $i \in [1 \dots s_2]$, $Q_i \simeq_{\text{bis}} P' \mid P'' \equiv P$, which establishes the first part of the result we want. To obtain the second part, we write:

- using the law $!T \equiv !T \mid !T$,

$$\Pi_{1\le i\le r_2} !Q_i^{(2)} \equiv \Pi_{1\le i\le r_2} !Q_i^{(2)} \mid \underbrace{!Q_{r_2}^{(2)} \mid \ldots \mid !Q_{r_2}^{(2)}}_{r_1 - r_2 \text{ times}};$$

- using the law $!T \equiv !T \mid T$,

$$\Pi_{1\le i\le r_1} !Q_i^{(1)} \mid \Pi_{r_1 < i\le s_1} Q_i^{(1)} \equiv \Pi_{1\le i\le r_1} !Q_i^{(1)} \mid \Pi_{r_1 < i\le s_1} Q_i^{(1)} \mid \underbrace{Q_{s_1}^{(1)} \mid \ldots \mid Q_{s_1}^{(1)}}_{s_2 - s_1 \text{ times}}.$$

Using these observations, we can write, taking $s = \max(s_1, s_2) = s_2$ and $m = \max(r_1, r_2) = r_1$:

$$\begin{aligned} Q &\equiv Q' \mid Q'' \\ &\equiv \Pi_{1\le i\le r_1} !Q_i^{(1)} \mid \Pi_{r_1+1\le i\le s_1} Q_i^{(1)} \bigm| \Pi_{1\le i\le r_2} !Q_i^{(2)} \mid \Pi_{r_2+1\le i\le s_2} Q_i^{(2)} \\ &\equiv \Pi_{1\le i\le r} !Q_i \bigm| \Pi_{r+1\le i\le s} Q_i\,. \end{aligned}$$

This finishes the proof for the direct implication. Note that the kind of unfolding we have been doing here, taking advantage of the properties of replication, is used to derive *expansions* between contexts (introduced below), and shall be needed in the proof of Theorem 3.10.

Concerning the reverse implication, if there exist r, s, and $(Q_i)_{i=1,\ldots s}$ such that $Q \equiv \Pi_{1\le i\le r} !Q_i \mid \Pi_{r+1\le i\le s} Q_i$ and $P \simeq_{\mathrm{bis}} Q_i$, write $!P \equiv \Pi_{1\le i\le r} !P \bigm| \Pi_{r+1\le i\le s} P$ as for the single case above, and use 3 and the congruence of $\simeq_{\mathrm{bis}}$ to conclude. □

3.2 Active contexts and an inductive characterisation

Below we present an inductive characterisation of $\simeq_{\mathrm{bis}}$. This results holds basically because of inversion properties (Lemmas 3.8 and 3.5), which testify the intensionality of $\simeq_{\mathrm{bis}}$.

A *context*, ranged over with $\mathcal{C}, \mathcal{D}, \ldots$, is a process term with some holes $[\,]_i$ in it, $i \in \mathbb{N}$, with the requirement for each hole to occur at most once (note that a hole $[\,]_i$ cannot be confused with ambient constructs thanks to the i subscript). i will be called an *index* of $\mathcal{C}$ if $\mathcal{C}$ contains the hole $[\,]_i$. Two contexts are *disjoint* if no hole $[\,]_i$ occurs in both. Structural congruence is extended on holes by adding the terminal rule $[]_i \equiv []_i$. The application of a context to a vector of terms is inductively defined by $\mathcal{C}[\epsilon] \stackrel{\mathrm{def}}{=} \mathcal{C}$ and $\mathcal{C}[P.\widetilde{P}] \stackrel{\mathrm{def}}{=} \mathcal{C}\{P/[\,]_i\}[\widetilde{P}]$, where i is the smallest index of $\mathcal{C}$ (for the case $\mathcal{C}$ is not a term, otherwise $\mathcal{C}[\widetilde{P}]$ is simply $\mathcal{C}$). In the remainder of the paper, we will actually restrict ourselves to expressions of the form $\mathcal{C}[\widetilde{P}]$ with $\widetilde{P}$ and $\mathcal{C}$ having the same arities, that is "fully applied" contexts.

Definition 3.6 (Active contexts, normalised contexts, expansion) *Let $\mathcal{C}, \mathcal{D}$ be some contexts.*

- *$\mathcal{C}$ is an* active context *if each hole appears underneath exactly one capability in $\mathcal{C}$.*

- *$\mathcal{C}$ is said to be* single *if its topmost constructor is an ambient or a capability construct. A context $\mathcal{C}$ is* normalised *if every replication occurring in $\mathcal{C}$ is applied to a single context.*

- *Given two contexts $\mathcal{C}$ and $\mathcal{D}$, if σ is a function that associates an index of $\mathcal{C}$ to any index of $\mathcal{D}$, we will say that $\mathcal{D}$ is a σ-*expansion *of $\mathcal{C}$, (written $\mathcal{C} \ \lhd_\sigma \ \mathcal{D}$), or equivalently that $\mathcal{C}$ is a σ-*contraction *of $\mathcal{D}$, if $\mathcal{C} \ \equiv \ \mathcal{D}(\sigma)$, where $\mathcal{D}(\sigma)$ is $\mathcal{D}\{[]_{\sigma(j)}/[]_j\}_{j\in I_\mathcal{D}}$, $I_\mathcal{D}$ being the (finite) set of indexes corresponding to the holes that occur in $\mathcal{D}$.*

Remarks:

- Active contexts can be described by the following grammar:

$$\mathcal{C} \quad ::= \quad \mathbf{0} \quad \Big| \quad n[\mathcal{C}] \quad \Big| \quad !\mathcal{C} \quad \Big| \quad \mathcal{C} \mid \mathcal{C} \quad \Big| \quad \mathsf{cap}.\,[\,]_i$$

  where each $[\,]_i$ has a single occurrence. We sometimes use cap_i to refer to the capability occurring immediately above the hole $[\,]_i$ in an active context.

- For any process P, there exists a unique active context $\mathcal{C}$ and a unique vector of terms $\widetilde{P}$ such that $P = \mathcal{C}[\widetilde{P}]$ (where $=$ denotes syntactic equality). This will be referred to as the *active context decomposition* of P. Moreover, we can remark that $\mathsf{ds}(P) = 1 + \max_{P_i \in \widetilde{P}} \mathsf{ds}(P_i)$.

- Using the axioms $!\mathbf{0} \equiv \mathbf{0}$ and $!(P \mid Q) \equiv !P \mid !Q$, we can always exhibit $Q \equiv P$ such that $Q = \mathcal{C}[\widetilde{Q}]$ where $\mathcal{C}$ is a normalised active context.

- Relation $\lhd_\sigma$ is defined up to $\equiv$, so that $\equiv\lhd_\sigma\equiv$ coincides with $\lhd_\sigma$.

Lemma 3.7 (Active context decomposition) *Let P, Q be two terms. Then $P \equiv Q$ iff there exist two active contexts $\mathcal{C}$ and $\mathcal{D}$, two vectors of processes $\widetilde{P}, \widetilde{Q}$ such that $P \equiv \mathcal{C}[\widetilde{P}], Q \equiv \mathcal{D}[\widetilde{Q}]$, and a function σ such that:*

1. *$\mathcal{C} \ \lhd_\sigma \ \mathcal{D}$;*

2. *for any j, $[\,]_{\sigma(j)}$ and $[\,]_j$ are underneath the same capability cap_j in $\mathcal{C}$ and $\mathcal{D}$ respectively, and $P_{\sigma(j)} \ \equiv \ Q_j$.*

Proof: The direct implication is trivial, by taking $\mathcal{C} = \mathcal{D}$. For the reverse implication, we reason by induction on $\mathcal{C}$ using Lemma 3.8 and an analogous of Lemma 3.5 for the relation $\equiv$. $\Box$

We conclude by stating some useful properties about expansion of active contexts, that are the counterpart of the corresponding results for $\simeq_{\text{bis}}$ (Lemma 3.5).

Lemma 3.8 (Inversion results for expansion) *Let $\mathcal{C}, \mathcal{D}, \mathcal{C}_1, \mathcal{C}_2$ be active contexts, and σ an expansion function whose domain contains the indexes of $\mathcal{D}$. Then:*

1. $\mathbf{0} \lhd_\sigma \mathcal{D}$ *iff* $\mathcal{D} \equiv \mathbf{0}$.

2. $n[\mathcal{C}] \lhd_\sigma \mathcal{D}$ *iff there exists some* $\mathcal{D}'$ *such that* $\mathcal{D} \equiv n[\mathcal{D}']$ *and* $\mathcal{C}' \lhd_\sigma \mathcal{D}'$

3. $\mathcal{C}_1 \mid \mathcal{C}_2 \lhd_\sigma \mathcal{D}$ *iff there exist* $\mathcal{D}_1, \mathcal{D}_2$ *such that* $\mathcal{D} \equiv \mathcal{D}_1 \mid \mathcal{D}_2$ *and* $\mathcal{C}_i \lhd_\sigma \mathcal{D}_i$ $(i = 1, 2)$.

4. $!\mathcal{C} \equiv \mathcal{D}$ *iff there exist* $r \geq 1, s \geq r, \mathcal{D}_i$ $(1 \leq i \leq s)$ *such that* $\mathcal{D} \equiv \Pi_{1 \leq i \leq r} !\mathcal{D}_i \mid \Pi_{r+1 \leq i \leq s} \mathcal{D}_i$, *and* $\mathcal{C} \lhd_\sigma \mathcal{D}_i$ *for* $i = 1 \dots s$.

5. $\mathsf{cap}.[]_i \lhd_\sigma \mathcal{D}$ *iff* $\mathcal{D} \equiv \mathsf{cap}.[]_j$ *and* $\sigma(j) = i$.

Proof: Cases 1,2 and 5 are easy, and so are the reverse implications for the other cases. The only points to prove are thus the direct implications for results 3 and 4.

We begin by establishing a useful result, in the case where two contexts $\mathcal{C}$ and $\mathcal{D}$ are related by $\mathcal{C} = \mathcal{D}(\sigma)$ for some σ. Suppose we have two families of single active contexts $(\mathcal{C}_i)_i, (\mathcal{D}_j)_j$ such that in any family, every two contexts belong to distinct congruence classes. Suppose further that we have two corresponding families $(\alpha_i)_i, (\beta_j)_j$ of "exponents", taken from $\mathbb{N}^* \cup \{\infty\}$, such that $\mathcal{C} \equiv \Pi_{i=1\dots l} \mathcal{C}_i^{\alpha_i}$ and $\mathcal{D} \equiv \Pi_{j=1\dots l'} \mathcal{D}_i^{\beta_j}$ (where $\mathcal{C}^\infty$ denotes $!\mathcal{C}$). Since expansion does not duplicate single terms, we have that for any $j \leq l'$, there exists a unique $i \leq l$ such that $\mathcal{D}_j(\sigma) = \mathcal{C}_i$. Moreover, if we call J_i the set of indices j such that $\mathcal{C}_i \equiv \mathcal{D}_j(\sigma)$, we have that $J_i \neq \emptyset$, and

$$\alpha_i = \sum_{j \in J_i} \beta_j \,.$$

Let us now prove the direct implication of case 3. Using the notations we have introduced, we may write a decomposition of $(\alpha_i)_i$ into two families $(\alpha_i^1), (\alpha_i^2)$ of indices in $\mathbb{N} \cup \{\infty\}$ such that $\mathcal{C}_\epsilon \equiv \Pi_{i=1\dots l} \mathcal{C}_i^{\alpha_i^\epsilon}$, $\epsilon = 1, 2$, with $\alpha_i = \alpha_i^1 + \alpha_i^2$. We then mimick this decomposition on the family $(\beta_j)_j$, and define two families $(\beta_j^1), (\beta_j^2)$ such that $\beta_j = \beta_j^1 + \beta_j^2$ and $\alpha_i^\epsilon = \sum_{j \in J_i} \beta_j^\epsilon$ for any i. We conclude by defining $\mathcal{D}_\epsilon$ as $\Pi_{j=1\dots r'} \mathcal{D}_j^\epsilon$.

To treat case 4, we first remark that in this case every α_i is equal to ∞, so that every J_i contains at least one j such that $\beta_j = \infty$. This is enough to establish case 4 for a single context $\mathcal{C}$ (i.e. with $r = 1$).

In the general case where $\mathcal{C}$ is not single, the sets J_is need not necessarily have the same cardinality. We therefore define:

$$J^\infty = \{j.\, \beta_j = \infty\}\,, \quad r = \max_i \sharp(J_i \cap J^\infty)\,, \quad \text{and} \quad s = \max_{j \notin J^\infty} \beta_j \,.$$

The idea is then to use the structural congruence laws $!\mathcal{C} \equiv !\mathcal{C} \mid !\mathcal{C}$ and $!\mathcal{C} \equiv !\mathcal{C} \mid \mathcal{C}$ in order to "unfold" copies of some components of $\mathcal{D}$, so that cardinalities can match r and s. More precisely, we set $\mathcal{D}_k = \Pi_{i=1\dots l} \mathcal{D}_{k,i}$, the $\mathcal{D}_{k,i}$s being defined as follows: for $k = 2 \dots r$, the pattern of the proof is given in an analogous proof above (Lemma 3.5).

□

We now exploit the definitions introduced above to provide an inductive caracterisation of $\simeq_{\text{bis}}$, based on the notion of active context decomposition. We start by defining the inductive relation $\sim_{\text{ind}}$.

Definition 3.9 (Inductive characterisation) *Given a process P, let $P \sim_{\text{ind}} (.)$ be the (least) predicate defined by induction on $\mathsf{ds}(P)$ as follows:*
if there exist two active contexts $\mathcal{C}$ and $\mathcal{D}$, and two vectors of processes $\widetilde{P}$, $\widetilde{Q}$, such that the following conditions hold:

1. *$P \equiv \mathcal{C}[\widetilde{P}]$ and $Q \equiv \mathcal{D}[\widetilde{Q}]$;*
2. *there is a function σ such that $\mathcal{C} \ \lhd_\sigma \ \mathcal{D}$;*
3. *for any j, $[\,]_{\sigma(j)}$ and $[\,]_j$ are underneath the same capability cap_j. Moreover, there are P'_j, Q'_j such that $P_{\sigma(j)} \stackrel{\mathsf{cap}_j}{\Longrightarrow} P'_j \sim_{\text{ind}} Q_j$, and $Q_j \stackrel{\mathsf{cap}_j}{\Longrightarrow} Q'_j$ and $P_{\sigma(j)} \ \sim_{\text{ind}} \ Q'_j$;*

then $P \ \sim_{\text{ind}} \ Q$.

Fact 1 (Correctness of definition) *Definition 3.9 is inductive.*

Proof: The point to check is that $\mathsf{ds}(P'_j) < \mathsf{ds}(P)$. This is a consequence of $\mathsf{ds}(P_{\sigma(j)}) < \mathsf{ds}(P)$ and $\mathsf{ds}(P'_j) \leq \mathsf{ds}(P_{\sigma(j)})$ (by Lemma 2.4). □

Remarks:

- The fact that $\sim_{\text{ind}}$ is inductive does not imply that it is a decidable relation. Indeed, in clause 3, there is a search for P'_j, Q'_j in the (potentially infinite) set of images of $P_\sigma(j)$ and Q_j. We will show in Section 6 that $\sim_{\text{ind}}$ is actually not decidable in general.
- Intuitively, when defining $\sim_{\text{ind}}$, we have to exploit expansions in order to conform to clause 4 in Lemma 3.5.

The following important result can now be assessed:

Theorem 3.10 *Relations $\simeq_{\text{bis}}$ and $\sim_{\text{ind}}$ coincide on MA.*

Proof: Let us first assume that $P \ \simeq_{\text{bis}} \ Q$. We reason by induction on $\mathsf{ds}(P)$ and a decomposition of P under the form $\mathcal{C}[\widetilde{P}]$ such that $\mathcal{C}$ is normalised. We then proceed by induction on $\mathcal{C}$; using Lemma 3.5 we may find some $\mathcal{C}', \widetilde{Q}$ such that $Q \equiv \mathcal{C}'[\widetilde{Q}]$ and some function σ satisfying the following conditions: (i) $\mathcal{C} \lhd_\sigma \mathcal{C}'$; (ii) for any j, if $[\,]_{\sigma(j)}$ and $[\,]_j$ are underneath the same capability cap_j, then there are P'_j, Q'_j such that $P_{\sigma(j)} \ \stackrel{\mathsf{cap}_j}{\Longrightarrow} \ P'_j \simeq_{\text{bis}} Q_j$, and $Q_j \ \stackrel{\mathsf{cap}_j}{\Longrightarrow} \ Q'_j$ and $P_{\sigma(j)} \ \simeq_{\text{bis}} \ Q'_j$; the function σ results from the construction of the proof of Lemma 3.5, case 4.

Now if $\mathsf{ds}(P) = 0$, this means that $\widetilde{P}, \widetilde{Q}$ are empty and then $P \ \sim_{\text{ind}} \ Q$. Otherwise, we have by induction that $P_{\sigma(j)} \stackrel{\mathsf{cap}_j}{\Longrightarrow} P'_j \sim_{\text{ind}} Q_j$, and $Q_j \stackrel{\mathsf{cap}_j}{\Longrightarrow} Q'_j$ and $P_{\sigma(j)} \ \sim_{\text{ind}} \ Q'_j$, and again $P \ \sim_{\text{ind}} \ Q$. So the first implication is proved.

The other implication follows the same pattern (induction on $\mathsf{ds}(P)$, then on the active context of P), using the results of Lemma 3.8 instead of Lemma 3.5. □

3.3 The subcalculi $\mathbf{MA}_{\mathrm{IF}}$ and $\mathbf{MA}_{\mathrm{IF}}^{\mathrm{syn}}$

We are now ready to introduce two subclasses of MA,, that shall be needed in the statement of our results. These are defined through restrictions on the class of processes P that are allowed after a capability cap in a term of the form $\mathsf{cap}.\,P$.

Definition 3.11 (image-finite processes, $\mathbf{MA}_{\mathrm{IF}}$, $\mathbf{MA}_{\mathrm{IF}}^{\mathrm{syn}}$) *Given a capacity* cap*, a process P is* cap-image finite *if the set $\{P' \mid P \overset{\langle\mathsf{cap}\rangle}{\Longrightarrow} P'\}$, quotiented by $\simeq_{\mathrm{bis}}$, is finite. The classes MA_{IF} and $MA_{\mathrm{IF}}^{\mathrm{syn}}$ are obtained by adding to the grammar of Table 1 the following constraints on the production* $\mathsf{cap}.\,P$*: in MA_{IF}, the continuation P must be a* cap*-image-finite process of MA_{IF}; in $MA_{\mathrm{IF}}^{\mathrm{syn}}$, the continuation P must be finite.*

The inclusions

$$\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}} \subseteq \mathrm{MA}_{\mathrm{IF}} \subseteq \mathrm{MA}$$

are strict. The processes P_0 and P_1 in Example 1 are in $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ and image-finite; however $\mathsf{out}\ n.\,P_0$ and $\mathsf{out}\ n.\,P_1$ are in $\mathrm{MA}_{\mathrm{IF}}$ but not in $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$. The process $P = \mathsf{open}\ n.\,(!\mathsf{open}\ a \mid !a[b[\mathbf{0}]])$ is in MA, but not in $\mathrm{MA}_{\mathrm{IF}}$, because $!\mathsf{open}\ a \mid !a[b[\mathbf{0}]]$ is not image finite.

$\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$, and hence also $\mathrm{MA}_{\mathrm{IF}}$, contains processes that are not image-finite: for instance, the processes used to encode Turing Machines in Section 6.

Let us stress two easy facts about calculi $\mathrm{MA}_{\mathrm{IF}}$ and $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$:

- *Closure under transition:* $\mathrm{MA}_{\mathrm{IF}}$ and $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$are closed under transition, that is given $P \in \mathrm{MA}_{\mathrm{IF}}$ (resp. $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$) and $Q \in \mathrm{MA}$ such that $P \longrightarrow Q$ or $P \overset{\mathsf{cap}}{\to} Q$, then Q also is in $\mathrm{MA}_{\mathrm{IF}}$ (resp. $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$).
- *Stability:* $\mathrm{MA}_{\mathrm{IF}}$ and $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ are stable under $\simeq_{\mathrm{bis}}$, that is given $P \in \mathrm{MA}_{\mathrm{IF}}$ (resp. $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$) and $Q \in \mathrm{MA}$ such that $P \simeq_{\mathrm{bis}} Q$, then Q also is in $\mathrm{MA}_{\mathrm{IF}}$ (resp. $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$).

These results can be established by induction on the sequentiality degree, the key property being condition 4 of $\simeq_{\mathrm{bis}}$. For $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$, stability can also be seen as a direct consequence of Theorem 3.12.

We now extend a result from [San01] on finite processes to $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$.

Theorem 3.12 (Characterisation of $\simeq_{\mathrm{bis}}$ on $\mathbf{MA}_{\mathrm{IF}}^{\mathrm{syn}}$) *Let $P \in MA_{\mathrm{IF}}^{\mathrm{syn}}$. Then for all $Q \in MA$,*

$$P \simeq_{\mathrm{bis}} Q \textit{ iff } P \equiv Q\,.$$

Proof: The proof for finite MA is reported in [San01]. The important fact to notice is that for P, Q *finite*, and any capability cap,

$$P \overset{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\mathrm{bis}} Q \overset{\langle\mathsf{cap}\rangle}{\Longrightarrow} \simeq_{\mathrm{bis}} P \qquad \text{iff} \qquad P \equiv Q\,.$$

This allows us to rewrite clause 2 in the definition of $\sim_{\mathrm{ind}}$ as simply $P_{\sigma(j)} \equiv Q_j$ when considering terms of $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$. This actually corresponds to the active context decomposition of $\equiv$, hence $\sim_{\mathrm{ind}} \;=\; \equiv$. □

The equality $\mathsf{out}\ n.\, P_0 \ \simeq_{\rm bis}\ \mathsf{out}\ n.\, P_1$ of Example 1 shows that Theorem 3.12 cannot be extended to $\mathrm{MA}_{\mathrm{IF}}$. Indeed, the $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ requirement that processes underneath capabilities should be finite seems essential for Thm. 3.12 to hold.

The results we have presented in this section express properties of Mobile Ambients at an operational level. We shall prove in Sec. 5 that $\simeq_{\rm bis}$ (or, equivalently, $\sim_{\rm ind}$) provides in fact an operational characterisation of logical equivalence, $=_L$.

4 Auxiliary formulas

In this section, we introduce some formulas that shall be needed in order to derive the results of Sec. 5. These results are also an illustration of the expressive power of the ambient logic. We successively give characteristaions of capabilites, both ephemeral and persistent, of finiteness, and of the set of free names of a process.

4.1 Formulas for ephemeral capabilities

An MA process can be viewed as a finite labelled tree in which labels can be ambient names, capabilities, replicated ambients, and replicated capabilities. If we can define formulas that describe all these labels, then we will be able to derive the characteristic formulas using standard techniques for image-finite processes with finite tree representation [GS86, SI94]. AL already has formulas $n[\mathcal{A}]$ that talk about ambient labels. We have to construct the formulas for the other labels. Formulas for capabilities are presented in [San01], for finite MA; they are also correct on MA. Their definition is given on Table 4.

$\mathsf{1Comp}$	$\stackrel{\rm def}{=}$	$\neg 0 \ \wedge\ 0 \Vert 0$	(this formula is from [San01])
$\mathsf{1Cap}$	$\stackrel{\rm def}{=}$	$\mathsf{1Comp} \ \wedge\ \neg\exists x.\ x[\top]$	
$\langle \mathsf{in}\ n\rangle.\, \mathcal{A}$	$\stackrel{\rm def}{=}$	$\mathsf{1Cap} \ \wedge\ \forall x.\ \big(n[0] \rhd \Diamond\, n[x[\mathcal{A}]]\big)@x$	
$\langle \mathsf{out}\ n\rangle.\, \mathcal{A}$	$\stackrel{\rm def}{=}$	$\mathsf{1Cap} \ \wedge\ \forall m.\ \big((\Diamond m[\mathcal{A}] \mid n[0])@n\big)@m$	
$\langle \mathsf{open}\ n\rangle.\, \mathcal{A}$	$\stackrel{\rm def}{=}$	$\mathsf{1Cap} \ \wedge\ \forall m.\ \big(\ n[m[0]] \rhd \Diamond\, m[0] \mid \mathcal{A}\ \big)$	
$[\![\mathsf{cap}]\!].\, \mathcal{A}$	$\stackrel{\rm def}{=}$	$\langle \mathsf{cap}\rangle.\, \top \ \wedge\ \neg\langle \mathsf{cap}\rangle.\, \neg\mathcal{A}$	for any capability cap

Table 4: Formulas for (ephemeral) capabilities

Lemma 4.1 (Formulas for capabilities) *For any capability* cap*, formula* $\mathcal{A}$ *and term* P*,*

- $P \models \langle \mathsf{cap}\rangle.\, \mathcal{A}$ *iff there are* P' *and* P'' *such that* $P \equiv \mathsf{cap}.\, P'$ *and* $P' \stackrel{\langle \mathsf{cap}\rangle}{\Longrightarrow} P''$ *with* $P'' \models \mathcal{A}$*.*

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

Proof: See [San01] for the formulas of the form $\langle\mathsf{cap}\rangle.\mathcal{A}$. Once these properties are established, the results for $[\![\mathsf{cap}]\!].\mathcal{A}$ easily follow. □

The formula $\mathsf{1Comp}$ (from [San01]) characterises the single processes. A formula $\langle\mathsf{cap}\rangle.\mathcal{A}$ expresses *possibility* (in Lemma 4.1, *at least one* derivative of P' satisfies $\mathcal{A}$), while a formula $[\![\mathsf{cap}]\!].\mathcal{A}$ expresses *necessity* (*all* derivatives of P' satisfy $\mathcal{A}$). Note that the latter formulas are not the dual of the possibility formulas, as in standard modal logics, because of the spatial aspects of AL. For instance, $[\![\mathsf{in}\ n]\!].\top$ is different from $\neg\langle\mathsf{in}\ n\rangle.\neg\top$: the latter is in fact equivalent to $\top$.

4.2 Characterising finiteness and free names

We now present a formula that is satisfied by all and only finite processes. The representation of replication seems *a priori* unfeasable in the present version of the ambient logic, as it does not provide a recursion operator. We may actually 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.

We thus show that we may reduce the characterisation of finiteness to the existence of a scenario which guarantees reachability of $\mathbf{0}$:

Lemma 4.2 *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 4.3 $P \in MA$ *is finite iff there are* Q, R, n *such that* $n[P \mid Q] \mid R \Longrightarrow \mathbf{0}$.

Proof:

- <u>Let us assume that P is finite</u>. We prove by induction on 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':

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

- $\mathsf{cap} = \mathsf{out}\ m$. Then 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.
- $\mathsf{cap} = \mathsf{open}\ m$. Then we set $Q = m[\mathbf{0}] \mid Q_1$ and $R = R_1$.

The first implication is thus established.

- Let us 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 $\mathbf{0}$.

The lemma is proved. □

Let us now consider the following formula:

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

Proposition 4.4 *For any $P \in MA$, $P \models \phi_{\mathrm{fin}}$ iff P is finite.*

Proof: Direct consequence of the previous lemma. □

We can also define, for any finite set $\mathcal{S}$ of names, a formula satisfied by those processes whose set of free names is precisely $\mathcal{S}$. For this construction we exploit the ability, using the modal formulas for capabilities, to detect unguarded occurrences of names, together with Lemma 4.5. We say that a process P is *flat* if the only process underneath all capabilities and inside all ambients of P is $\mathbf{0}$.

Lemma 4.5 *For all P, n, $n \in \mathrm{fn}(P)$ 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: We first introduce an auxiliary notion which will be useful for the proof.

The *occurrence depth* of a name n in a term is given by a function $\mathsf{depth}_n : \mathcal{P} \longrightarrow \mathbb{N} \cup \{\infty\}$, inductively defined as follows:

- $\mathsf{depth}_n(\mathbf{0}) = \infty$.
- $\mathsf{depth}_n(n[P_1]) = 0$, and for $n \neq m$, $\mathsf{depth}_n(m[P_1]) = \mathsf{depth}_n P_1 + 1$.
- $\mathsf{depth}_n((!)P_1 \mid \ldots \mid (!)P_r) = \min_{1\leq i\leq r} \mathsf{depth}_n(P_i)$.
- $\mathsf{depth}_n(\mathsf{cap}.\,P) = 0$ for $\mathsf{cap} \in \{\mathsf{in}\ n, \mathsf{out}\ n, \mathsf{open}\ n\}$, and $\mathsf{depth}_n(\mathsf{cap}.\,P) = \mathsf{depth}_n(P) + 1$ otherwise.

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:

- first implication: Let us assume that $\mathsf{depth}_n(P) < \infty$. 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 $\mathsf{depth}_n(P) = i + 1$, let us assume for example that $P \equiv \mathsf{in}\ m_1.\,P_1 \mid P_2$ 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 P_1. But $Q_1, R_1, S_1 \mid P_2$ also satisfy these conditions for $P_1 \mid P_2$. So if we set $Q = Q_1$, $R = m_1[\mathbf{0}] \mid \mathsf{open}\ m_1 \mid R_1$ and $S = S_1 \mid P_2$, then Q, R, S can be chosen for P. For the other cases, we proceed as in the proof of Lemma 4.3.

  The first implication is proved.
- second implication: Let us 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 $m[P \mid Q] \mid R \Longrightarrow T$, $n \notin \mathrm{fn}(T)$. The second implication thus follows.

□

We can now define the formula to capture the set of free names of a process; it is given in Table 5 (in the definition of $\mathsf{refers}(\{n_1, \ldots, n_k\})$, we adopt the convention that $\bigwedge_{i=0\ldots k} F_i = \top$ and $\bigvee_{i=0\ldots k} F_i = \bot$ when $k = 0$).

Formula $\mathsf{refers1}(n)$ detects whether name n occurs in a process, while $\mathsf{toplevel}(n)$ detects whether n occurs at top level (i.e. P satisfies this formula iff $\mathsf{depth}_n(P) = 0$).

Proposition 4.6 (Intensionality of name occurrences) *There is a computable function that associates to each finite set $\mathcal{S}$ of names a formula $\mathsf{refers}(\mathcal{S})$ such that for any $P \in MA$*

$$P \models \mathsf{refers}(\mathcal{S}) \qquad \textit{iff} \qquad \mathcal{S} = \mathrm{fn}(P)\,.$$

Proof: Consequence of the previous Lemma. □

$\mathcal{A}^\omega$	$\stackrel{\text{def}}{=}$	$(\mathsf{1Comp} \to \mathcal{A})^\forall$
$m = n$	$\stackrel{\text{def}}{=}$	$(n[\top])@m$ (this formula is from [CG00])
$\mathsf{flat}(m)$	$\stackrel{\text{def}}{=}$	$[\![\mathsf{in}\ m]\!].\,0\ \vee\ [\![\mathsf{out}\ m]\!].\,0\ \vee\ [\![\mathsf{open}\ m]\!].\,0\ \vee m[0]$
$\mathsf{flatcond}(n)$	$\stackrel{\text{def}}{=}$	$\big(\exists m.\ \neg(m = n)\ \wedge\ \mathsf{flat}(m)\ \big)^\omega$
$\mathsf{toplevel}(n)$	$\stackrel{\text{def}}{=}$	$\big(\langle\mathsf{in}\ n\rangle.\top\ \ \vee\ \ \langle\mathsf{out}\ n\rangle.\top\ \ \vee\ \ \langle\mathsf{open}\ n\rangle.\top\ \ \vee\ \ n[\top]\ \big)\ \ \mid\ \top$
$\mathsf{refers1}(n)$	$\stackrel{\text{def}}{=}$	$\forall m.\ \mathsf{flatcond}(n) \blacktriangleright \big(\ \mathsf{flatcond}(n) \blacktriangleright \Diamond\ m[\mathsf{toplevel}(n)]\ \big)@m$
$\mathsf{refers}(\{n_1, \ldots, n_k\})$	$\stackrel{\text{def}}{=}$	$\bigwedge_{i=0\ldots k} \mathsf{refers1}(n_i)\ \wedge\ \forall x.\ \mathsf{refers1}(x)\ \to\ \bigvee_{i=0\ldots k} x = n_i$

Table 5: Formulas for free names

4.3 Formulas for persistence

We now move to the formulas that characterise 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 $\mathcal{A}^\omega \stackrel{\text{def}}{=} (\mathsf{1Comp} \to \mathcal{A})^\forall$. The second part of the definition of $!\mathcal{A}$ addresses persistence, by saying that there are infinitely many processes at top level that satisfy $\mathcal{A}$. The whole definitions are given in Table 6

$\mathsf{Rep}_{\mathsf{in}\ n}(\mathcal{A})$	$\stackrel{\text{def}}{=}$	$\mathcal{A}^\omega\ \ \wedge\ \ \forall m.\ (\neg\mathsf{refers1}\ m)\ \to$ $([\![\mathsf{out}\ n]\!].\,0)^\omega \rhd \big(\ n[0] \rhd \Box\Diamond\ (\ n[m[\mathcal{A}\ \mid\ \top]]\ )\big)@m$
$\mathsf{Rep}_{\mathsf{out}\ n}(\mathcal{A})$	$\stackrel{\text{def}}{=}$	$\mathcal{A}^\omega\ \ \wedge \forall m.\ (\neg\mathsf{refers1}\ m)\ \to$ $([\![\mathsf{in}\ n]\!].\,0)^\omega \rhd \big(\ n[0] \rhd \Box\ \Diamond(\ m[\mathcal{A}\ \mid\ \top] \mid n[0]\ )\big)@m$
$\mathsf{Rep}_{\mathsf{open}\ n}(\mathcal{A})$	$\stackrel{\text{def}}{=}$	$\mathcal{A}^\omega\ \wedge\ (n[0])^\omega \rhd \Box\ (\ \mathcal{A}\ \mid\ \top\ )$
$\mathsf{Rep}_{n[]}(\mathcal{A})$	$\stackrel{\text{def}}{=}$	$(n[\mathcal{A}])^\omega\ \wedge\ ([\![\mathsf{open}\ n]\!].\,0)^\omega \rhd \Box\ (n[\mathcal{A}]\ \mid\ \top\ )$
$!n[\mathcal{A}]$	$\stackrel{\text{def}}{=}$	$\mathsf{Rep}_{n[]}(\mathcal{A})$

Table 6: Formulas for persistent single terms

The interpretation of these formulas can be expressed modulo some hypotheses concerning the sequentiality and depth degree of their models, captured by the following notions.

Definition 4.7 (Selective and expressive formulas) *We say that 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*), we also say that* $\mathcal{A}$ *is* cap-expressive *(resp.* n-expressive*) if all terms satisfying it are of the form* $\mathsf{cap}.\,P$ *(resp.* $n[P]$*).*

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 toplevel in a process, and interpret the formulas introduced above.

Lemma 4.8 (Characterisation of replication of single processes)

1. *Given a capability* cap*, and a sequentially selective and* cap*-expressive formula* $\mathcal{A}$*, define* $!\mathcal{A} \stackrel{\text{def}}{=} \mathsf{Rep}_{\mathsf{cap}}(\mathcal{A})$*. Then* $P \models !\mathcal{A}$ *iff there are* $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ *such that* $P \equiv \Pi_{1\leq i\leq r}!\mathsf{cap}.\, P_i \mid \Pi_{r+1\leq i\leq s}\mathsf{cap}.\, P_i$*, and* $\mathsf{cap}.\, P_i \models \mathcal{A}$ *for all* $1 \leq i \leq s$*, where* $\Pi_{1\leq i\leq t} Q_i$ *abbreviates* $Q_1 \mid \ldots \mid Q_t$*.*

2. *Similarly, for any name* n *and depth selective and* n*-expressive formula* $\mathcal{A}$*, define* $!\mathcal{A} \stackrel{\text{def}}{=} \mathsf{Rep}_{n[]}(\mathcal{A})$*. Then* $P \models !\mathcal{A}$ *iff there are* $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ *such that* $P \equiv \Pi_{1\leq i\leq r}!n[P_i] \mid \Pi_{r+1\leq i\leq s} n[P_i]$*, and* $n[P_i] \models \mathcal{A}$ *for all* $1 \leq i \leq s$*.*

Proof:
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 toplevel in $m[P \mid Q] \mid n[\mathbf{0}]$, ambient n remains at toplevel 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 do 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 now assume that $P \models \mathsf{Rep}_{\mathsf{in}\ n}(\mathcal{A})$. Then according to the first part of the formula, there exist some P_is 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 stricly smaller than $\mathsf{ds}(P)$. 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 $\mathsf{Rep}_{\mathsf{in}\ n}(\mathcal{A})$, and we obtain a contradiction. Hence, at least one of the P_is is replicated, and the reverse implication is proved.

Case 2. Let us now treat case 2, and assume that $P \equiv !n[P_1] \mid \ldots \mid (!)n[P_r]$, with the P_is such that $P_i \models \mathcal{A}$. Then P satisfies $\mathsf{Rep}_{n[]}(\mathcal{A})$ iff for any $Q \equiv \mathsf{open}\ n^\omega$, and any R such that $P \mid Q \Longrightarrow R$, there are R_is 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 $\mathsf{Rep}_{n[]}(\mathcal{A})$. 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. □

5 Characteristic formulas and completeness

Characteristic formulas and completeness for an algebraic characterisation of the logical equivalence are two related problems and the proof techniques required to establish them are similar. In fact, the existence of characteristic formulas is a more demanding result that completeness of $\simeq_{\mathrm{bis}}$ w.r.t. $=_L$, and will actually not be derivable for the whole calculus. In this section, we establish two results, the existence of characteristic formulas for $\mathrm{MA}_{\mathrm{IF}}$ and completeness of $\simeq_{\mathrm{bis}}$ on MA.

5.1 Characteristic formulas

A characteristic formula of a process P is a formula that is satisfied by all and only the processes Q in the relation $\simeq_{\mathrm{bis}}$ with P. This name is justified by the fact that such a characteristic formula will also characterise a class of $=_L$ by correction of $\simeq_{\mathrm{bis}}$ (Lemma 3.3).

Example 2 *We first illustrate the constructions we shall define below on some examples. Let us consider processes $P_1 = !\mathsf{open}\ n.\, n[\mathbf{0}]$, $P_2 = \mathsf{open}\ n \mid n[\mathbf{0}]$ and $P_3 = !\mathsf{open}\ n.\, P_2$, and explain how we construct characteristic formulas for such terms, using the constructions we have introduced in the previous section (Tables 4 and 6).*

A characteristic formula for P_1 is easy to define since the continuation term $n[\mathbf{0}]$ as no reducts. Hence the formula $[\![\mathsf{open}\ n]\!].\, n[0]$, using a formula for necessity*, satisfies the conditions of Lemma 4.8, and a characteristic formula for P_1 is*

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

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

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

However, in P_3 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 4.8 (and is not *a charateristic formula of* $\mathsf{open}\ n.\,P_2$*, although* $\mathcal{F}_2$ *is for* P_2*), so that we need to add the possibility to reduce to* $\mathbf{0}$*, yielding to the formula* $[\![\mathsf{open}\ n]\!].\,(\mathcal{F}_2 \ \vee\ \mathbf{0})$*. But then we also accept the term* $\mathsf{open}\ n$*, which shows why we are led to add a* possibility *condition in the formula. Finally, a characteristic formula for* P_3 *is the following:*

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

To construct a characteristic formula $\mathcal{F}_P$ of a process P, we may suppose (up to $\equiv$) that replication only appears above single terms, in other words that the active context of P is normalized. We then define the characteristic formula $\mathcal{F}_P$ of process P by structural induction as given in Table 7.

$\mathcal{F}_{\mathbf{0}}$	$\stackrel{\text{def}}{=}$	0	$\mathcal{F}_{P\mid Q}$	$\stackrel{\text{def}}{=}$	$\mathcal{F}_P \ \mid\ \mathcal{F}_Q$
$\mathcal{F}_{n[P]}$	$\stackrel{\text{def}}{=}$	$n[\mathcal{F}_P]$	$\mathcal{F}_{\mathsf{cap}.P}$	$\stackrel{\text{def}}{=}$	$\langle \mathsf{cap}\rangle.\,\mathcal{F}_P \ \wedge\ [\![\mathsf{cap}]\!].\,\bigvee_{\{P',\ P \Longrightarrow P'\}/\simeq_{\text{bis}}} \mathcal{F}_{P'}$
$\mathcal{F}_{!n[P]}$	$\stackrel{\text{def}}{=}$	$\mathsf{Rep}_{n[]}(\mathcal{F}_P)$	$\mathcal{F}_{!\mathsf{cap}.P}$	$\stackrel{\text{def}}{=}$	$\mathsf{Rep}_{\mathsf{cap}}(\mathcal{F}_{\mathsf{cap}.P})$

Table 7: Characteristic formulas in MA_{IF}

Theorem 5.1 (Characteristic formulas for MA_{IF}) *For any term* P*, define* $\mathcal{F}_P$ *according to Table 7. Then*

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

Proof: We reason by induction on the sequentiality degree of P, and then by structural induction :

- if $P = \mathbf{0}$, $\mathcal{C} = \mathbf{0}$, and $\mathcal{C}^* = \mathcal{F}_P = 0$. Then $Q \models \mathcal{F}_P$ iff $Q \ \equiv\ \mathbf{0}$, ie $P \ \simeq_{\text{bis}}\ Q$.
- the cases $P = n[P']$, $P = P_1 \mid P_2$, $P = !P'$ are also straightforward (using Lemma 3.5). For example, $Q \models \mathcal{F}_{P_1\mid P_2}$ iff (by induction) there are Q_1, Q_2 such that $Q \ \equiv\ Q_1 \mid Q_2$ and $P_i \ \simeq_{\text{bis}}\ Q_i$ iff (by Lemma 3.5) $P \ \simeq_{\text{bis}}\ Q$.
- let us now consider the case where $P = \mathsf{cap}.\,P'$. Let Q be any term such that $Q \models \mathcal{F}_P$. Then there are some Q', P'' such that $Q \ \equiv\ \mathsf{cap}.\,Q'$, $Q' \ \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow}\ \models \mathcal{F}_{P'}$, $P' \ \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow}\ P''$ and $Q \models \mathcal{F}_{P''}$ (since $Q' \ \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow}\ Q'$). By induction, this shows that $Q' \ \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow}\ \simeq_{\text{bis}}\ P' \ \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow}\ \simeq_{\text{bis}}\ Q'$, so that $P \ \simeq_{\text{bis}}\ Q$.

  Conversely, we have that P satisfies $\mathcal{F}_P$ trivially, and by correction, any Q bisimilar to P satisfies also $\mathcal{F}_P$.

$\square$

Together with correction, this result shows that $\simeq_{\text{bis}} = =_L$ on MA_{IF}, that is $\simeq_{\text{bis}}$ is complete on this sublanguage. Moreover the characteristic formulas for $\simeq_{\text{bis}}$ actually also characterise the equivalence classes induced by the logic. We will see in section 5.3 how much we can extend this result to MA.

5.2 Discussion about model-checking and tautologies

In AL, the construction of characteristic formulas has connections with the decidability of other problems related to the logic, namely model-checking (whether $P \models \mathcal{A}$ holds, for any given process P and formula $\mathcal{A}$) and validity (whether a given formula $\mathcal{A}$ is satisfied by all processes). These problems have been addressed in [CT01, CYO01, CCG02]. In particular, for AL, in [CT01] the undecidability of tautologies is established on a small fragment of the logic, a result that entails the undecidability of model-checking.

Using characteristic formulas, we can derive results similar, albeit weaker, to those in [CT01], proceeding the other way around (they are weaker because undecidability is established on a larger language). Indeed, we have, for all $P, Q, R \in \mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$:

$$P \models \mathcal{F}_Q \wedge \Diamond\mathcal{F}_R \quad \text{iff} \quad P \equiv Q \Longrightarrow R\,.$$

Now, since $\Longrightarrow$ is undecidable on $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ (as will be shown in Section 6), so is the model checking problem.

More generally, the existence of characteristic formulas allows us to consider validity and model-checking to be equivalent decision problems. To see why, first remark that validity can be encoded inside model-checking [CG00], thanks to the $\triangleright$ connective. Conversely, we can encode the model-checking problem inside validity using characteristic formulas as follows (recall that $\mathcal{F}_P$ denotes the characteristic formula of process P):

$$\text{for all } P \in \mathrm{MA}_{\mathrm{IF}},\ \text{for all } \mathcal{A},\quad P \models \mathcal{A} \ \text{ iff } \ \vdash \mathcal{F}_P \to \mathcal{A}\,.$$

In [CT01] and [CYO01], model-checking and validity turn out to be either both decidable or both undecidable, the key issue being the presence of name quantification in the logic. We do not know at present whether characteristic formulas could be derived in the setting of [CYO01].

5.3 Completeness of intensional bisimilarity

We derive the completeness of $\simeq_{\mathrm{bis}}$ by exploiting the formulas introduced above, and two further key properties. The first one essentially says that we can construct characteristic formulas for active contexts (Definition 3.6). The second result expresses the fact that given a fixed active context, we may restrict ourselves to a finite set of processes (Fact 2).

Some useful sets of processes. In order to prove completeness, we need to adapt the reasonings we have made above to some specific sets of terms. The basic ideas are the same, and in fact we only need some additional hypotheses about the processes being used to model formulas. We introduce the corresponding notions:

Definition 5.2 (Sets of processes $\mathcal{E}_P$, $\mathsf{frozen}(P)$ and $\mathsf{cont}(P)$) *For any process P, we define $\mathcal{E}_P \stackrel{\text{def}}{=} \{P',\ \exists\, \mathsf{cap}.\ P \stackrel{\langle\mathsf{cap}\rangle}{\Longrightarrow} P'\}$.*

The set $\mathsf{frozen}(P)$ is the set of the subterms of P that appear after exactly the first level of prefixes, that is for the decomposition $P = \mathcal{C}[\widetilde{P}]$ where $\mathcal{C}$ is an *active context and* $\widetilde{P} = (P_1, \ldots, P_n)$*,* $\mathsf{frozen}(P) = \{P_i\}_{1 \le i \le n}$*.*

The set $\mathsf{cont}(P)$ *is the set of subterms of* P *that appear after any number of prefixes in* P*, in other words:*

$$\mathsf{cont}(P) \stackrel{\text{def}}{=} \mathsf{frozen}(P) \cup \bigcup_{P' \in \mathsf{frozen}(P)} \mathsf{cont}(P').$$

Note that the above equality can be taken as a definition of $\mathsf{cont}(P)$ by induction on the sequentiality degree of P.

If P, P' are two congruent terms, then $\mathsf{frozen}(P) = \mathsf{frozen}(P')$ up to $\equiv$, and ditto for $\mathsf{cont}(P)$ and $\mathsf{cont}(P')$. Hence these sets (in their quotiented version) are uniquely determined by the congruence class of P.

We now present a parametrised (or, in our terminology, *restricted*) version of the results we have proved above.

Definition 5.3 (Restricted characteristic formula, sequentiality and depth selectivity) *Let* $\mathcal{E}$ *be a set of terms,* P *a term and* F *a formula. We will say that* F *is a characteristic formula for* P *on* $\mathcal{E}$ *if for any* $Q \in \mathcal{E}$*,* $Q \models F$ *iff* $Q \simeq_{\text{bis}} P$*. We will also say that* F *is sequentially selective (resp. depth selective) on* $\mathcal{E}$ *if all the terms* P *satisfying* F *and such that* $\mathsf{cont}(P) \subseteq \mathcal{E}$ *have the same sequentiality degree (resp. depth degree).*

We have an analogue of Lemma 4.8 for restricted selectivity, whose proof is the same.

Lemma 5.4 (Characterisation of replication of single processes (restricted)) *Let* $\mathcal{E}$ *be a set of terms,* F *a sequentially selective formula on* $\mathcal{E}$*, and* P *any term such that* $\mathsf{cont}(P) \subseteq \mathcal{E}$*.*

1. *Given a capability* cap*, and a sequentially selective on* $\mathcal{E}$ *and* cap*-expressive formula* $\mathcal{A}$*,* $P \models \mathsf{Rep}_{\mathsf{cap}}(\mathcal{A})$ *iff there are* $r \ge 1, s \ge r, P_i$ *(*$1 \le i \le s$*) such that* $P \equiv \Pi_{1 \le i \le r} !\mathsf{cap}.\, P_i \mid \Pi_{r+1 \le i \le s} \mathsf{cap}.\, P_i$*, and* $\mathsf{cap}.\, P_i \models \mathcal{A}$ *for all* $1 \le i \le s$*.*

2. *Consider a formula* $\mathcal{A}$ *and a name* n *such that* $\mathcal{A}$ *is depth selective on* $\mathcal{E}$ *and* n*-expressive. Then* $P \models \mathsf{Rep}_{n[]}(\mathcal{A})$ *iff there are* $r \ge 1, s \ge r, P_i$ *(*$1 \le i \le s$*) such that* $P \equiv \Pi_{1 \le i \le r} !n[P_i] \mid \Pi_{r+1 \le i \le s} n[P_i]$ *and* $n[P_i] \models \mathcal{A}$ *for all* $1 \le i \le s$*.*

With the definitions above, completeness of $\simeq_{\text{bis}}$ boils down to the existence of a characteristic formula of P on $\{P, Q\}$ for any terms P, Q. We shall exploit this idea in a more refined fashion, to establish completeness by reasoning inductively on the sequentiality degree of terms. We will namely work with the sets $\mathcal{E}_P$ (see Def. 5.3), and prove that for any P, Q, there exists a characteristic formula of P on $\mathcal{E}_Q$ (Lemma 5.8). Then completeness will follow easily.

Fact 2 (Finiteness of $\mathsf{cont}(P)$**)** *For any* $P \in MA$*,* $\mathsf{cont}(P)$ *is finite.*

Let us now explain why the set $\mathsf{cont}(P)$ is interesting. In the sense of the next lemma, the set $\mathsf{cont}(P)$ allows us to "restrict" the future of P, and reason about a finite set of terms.

Lemma 5.5 (Predictive lemma) *Let P, Q be two terms such that $P \longrightarrow Q$ or $P \stackrel{\mathsf{cap}}{\to} Q$ for some cap. Then $\mathsf{cont}(Q) \subseteq \mathsf{cont}(P)$ when quotiented by $\equiv$.*

Remark: The statement of this result requires some care when taking also communications into account, because a prefixed term may lead to several continuations, that differ by some name replacing. However, as our calculus has no name generation mechanism, it is enough to consider that a frozen open term may be closed with an already known name.

We have not been able to define characteristic formulas for all terms of MA. We will however try to tell as much as we can within a single formula. To achieve this, we show that we can capture the active context of a term with a formula. This is done by defining a function that computes, starting from a normalized active context, a *formula context* (i.e. a logical formula with holes in it, along the lines of the definition of plain contexts). We shall then give the interpretation of the construction.

The active contexts we consider are *normalised*, as defined above, in the sense that replication is only applied to single terms (we can suppose this without loss of generality thanks to the $\equiv$ laws we have for replication). The formula context we produce from an active context of this form associates two holes $[]_i^{\Diamond}$ and $[]_i^{\Box}$ to each hole $[]_i$ in the initial context. The formula contexts we define are then to be filled with two vectors of formulas $\widetilde{F^{\Diamond}}$ and $\widetilde{F^{\Box}}$, matching the corresponding holes.

Definition 5.6 (Active context formula) *Consider an active normalised context $\mathcal{C}$. We define its associated context formula $\mathcal{C}^*$ by the equations given below. In particular, we associate to every prefixed hole $[]_i$ of $\mathcal{C}$ a context formula having two holes, that we denote $[]_i^{\Diamond}$ and $[]_i^{\Box}$. Hence the arity of $\mathcal{C}^*$ is twice the arity of $\mathcal{C}$.*

$$
\begin{array}{rcl@{\qquad}rcl}
(\mathbf{0})^* & \stackrel{\text{def}}{=} & 0 & (\mathcal{C}_1 \mid \mathcal{C}_2)^* & \stackrel{\text{def}}{=} & \mathcal{C}_1^* \mid \mathcal{C}_2^* \\
(n[\mathcal{C}])^* & \stackrel{\text{def}}{=} & n[\mathcal{C}^*] & (\mathsf{cap}.\,[]_i)^* & \stackrel{\text{def}}{=} & \langle\mathsf{cap}\rangle.\,[]_i^{\Diamond} \ \wedge\ [\![\mathsf{cap}]\!].\,[]_i^{\Box} \\
(!n[\mathcal{C}])^* & \stackrel{\text{def}}{=} & \mathsf{Rep}_{n[]}(\mathcal{C}^*) & (!\mathsf{cap}.\,[]_i)^* & \stackrel{\text{def}}{=} & \mathsf{Rep}_{\mathsf{cap}}((\mathsf{cap}.\,[]_i)^*)
\end{array}
$$

Lemma 5.7 (Characterisation of active contexts) *Let $\mathcal{C}$ be a normalised active context. Let $\widetilde{F^{\Diamond}}$ and $\widetilde{F^{\Box}}$ be two vectors of formulas (called the* filling formulas*) whose lengths are equal to the arity of $\mathcal{C}^*$. Let us also consider a set of processes $\mathcal{E}$, and assume that the filling formulas are such that for any index i of a hole $\mathsf{cap}_i.\,[]_i$ of $\mathcal{C}$, the formula*

$$F_i \stackrel{\text{def}}{=} \langle\mathsf{cap}_i\rangle.\,F_i^{\Diamond} \ \wedge\ [\![\mathsf{cap}_i]\!].\,F_i^{\Box} \qquad (=\ (\mathsf{cap}_i.\,[]_i)^*[F_i^{\Diamond}, F_i^{\Box}])$$

is sequentially selective on $\mathcal{E}$.

Then for any term P such that $\mathsf{cont}(P) \subseteq \mathcal{E}$, $P \models \mathcal{C}^*[F^{\Diamond}, F^{\Box}]$ iff there exists an active context $\mathcal{C}_1$, a vector of processes $\widetilde{P_1}$, and a function σ such that $P \equiv \mathcal{C}_1[\widetilde{P_1}]$ and :

1. *$\mathcal{C} \lhd_\sigma \mathcal{C}_1$;*
2. *for any index j of a hole $\mathsf{cap}_j.[]_j$ of $\mathcal{C}_1$, $\mathsf{cap}_j.P_j \models F_{\sigma(j)}$*

Proof: Let us consider P such that $\mathsf{cont}(P) \subseteq \mathcal{E}$ and reason by induction on $\mathcal{C}$:

- if $\mathcal{C} = \mathbf{0}$ then we can take $\mathcal{C}_1 = \mathbf{0}$ (and we have $P \equiv \mathbf{0}$).

- if $\mathcal{C} = n[\mathcal{C}_0]$ then $P \models \mathcal{C}^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]$ iff $P \equiv n[P_0]$ and $P_0 \models \mathcal{C}_0^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]$. By induction, we may find $\mathcal{C}_{1,0}$ such that $P_0 \equiv \mathcal{C}_{1,0}[\widetilde{P}]$ for some $\widetilde{P}$ and a function σ having the right properties. Then we may establish the result taking $\mathcal{C}_1 = n[\mathcal{C}_{1,0}]$ and σ.

- if $\mathcal{C} = \mathcal{C}_a \mid \mathcal{C}_b$ then $P \equiv P_a \mid P_b$ with $P_i \models \mathcal{C}_i^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]$ $(i = a, b)$. This gives by induction contexts $\mathcal{C}_{a,1}$ and $\mathcal{C}_{b,1}$ and functions σ_a and σ_b. Without loss of generality, we may suppose the indices of the holes occurring in these contexts to be disjoint, so that we can define σ, a function which extends both σ_a and σ_b. Define then $\mathcal{C}_1 \stackrel{\text{def}}{=} \mathcal{C}_{a,1} \mid \mathcal{C}_{b,1}$. We have $\mathcal{C} = \mathcal{C}_a \mid \mathcal{C}_b \equiv \mathcal{C}_{a,1}(\sigma_a) \mid \mathcal{C}_{b,1}(\sigma_b) = \mathcal{C}_1(\sigma)$, and condition 2 also holds by induction.

- if $\mathcal{C} = \mathsf{cap}_i.[]_i$, then vectors $\widetilde{F^{\Diamond}}$ and $\widetilde{F^{\Box}}$ have only one component. Moreover, $\mathcal{C}^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]$ is exactly F_i, and $P \equiv \mathsf{cap}_i.P'$ with $\mathsf{cap}_i.P' \models F_i$. So we may take $\mathcal{C}_1 = \mathcal{C}$ and just any function.

- if $\mathcal{C} = !\mathsf{cap}_i.[]_i$, then $\mathcal{C}^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]$ is $\mathsf{Rep}_{\mathsf{cap}}(F_i)$ and we may apply Lemma 5.4 with $\mathcal{E}$ (as by hypothesis F_i is sequentially selective on $\mathcal{E}$), so that there are $r \geq 1, s \geq r, P_i$ $(1 \leq i \leq s)$ such that $P \equiv \Pi_{1 \leq i \leq r} !\mathsf{cap}.P_i \mid \Pi_{r+1 \leq i \leq s} \mathsf{cap}.P_i$, and $\mathsf{cap}.P_i \models \mathcal{A}$ for all $1 \leq i \leq s$. Then we set $\mathcal{C}_1 = \Pi_{1 \leq i \leq r} !\mathsf{cap}.[]_i \mid \Pi_{r+1 \leq i \leq s} \mathsf{cap}.[]_i$, and σ can be freely chosen as long as it is constant on these indices. Conditions 1 and 2 are then satisfied.

- if $\mathcal{C} = !n[\mathcal{C}_1]$, we reason on the same way. Just note that the depth selective hypothesis is there expressed by the inductive hypothesis.

$\Box$

We may now join the two last results to obtain the key lemma for completeness.

Lemma 5.8 (Characteristic formulas on $\mathcal{E}_Q$) *For any two terms P, Q of MA, there is a characteristic formula $F_{P,Q}$ for P on $\mathcal{E}_Q$, or in other words:*

- *$P \models F_{P,Q}$*
- *for any Q' and cap such that $Q \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q'$, if $Q' \models F_{P,Q}$, then $Q' \simeq_{\text{bis}} P$.*

To establish this result, we will first establish it for a particular case, the general case requiring an induction that we will be able to set only after that.

Lemma 5.9 (Sequentiality degree constraint lemma) *For any process $P \in MA$, there exists a formula $F_{\mathsf{ds}(P)}$ such that:*

- $P \models F_{\mathsf{ds}(P)}$
- *for any term Q, if $Q \models F_{\mathsf{ds}(P)}$, then $\mathsf{ds}(Q) \geq \mathsf{ds}(P)$.*

Proof: Let us first reason by induction on $\mathsf{ds}(P)$:

- for $\mathsf{ds}(P) = 0$, $F_{\mathsf{ds}(P)} = \top$ is sufficient.
- for $\mathsf{ds}(P) > 0$, let us assume we may find formulas $F_{\mathsf{ds}(P')}$ for any P' such that $\mathsf{ds}(P') < \mathsf{ds}(P)$. Moreover, let us reason by induction on P.
  - the case $P = \mathbf{0}$ is impossible.
  - for $P = P_1 \mid P_2$, there is some i such that $\mathsf{ds}(P) = \mathsf{ds}(P_i)$. Then we may choose $F_{\mathsf{ds}(P)} = F_{\mathsf{ds}(P_i)} \mid \top$. In the same way, let us set $F_{\mathsf{ds}(!P)} = F_{\mathsf{ds}(P)} \mid \top$ and $F_{\mathsf{ds}(n[P])} = n[F_{\mathsf{ds}(P)}]$.
  - for $P = \mathsf{cap}.\, P'$, we use the general induction hypothesis to get some $F_{P'}$. Let us then take $F_{\mathsf{ds}(P)} = \langle \mathsf{cap} \rangle .\, F_{\mathsf{ds}(P')}$. Then $P \models F_{\mathsf{ds}(P)}$ obviously, and for any Q such that $Q \models F_{\mathsf{ds}(P)}$, we deduce (from Lemma 4.1) that there are some Q', Q'' such that $Q \equiv \mathsf{cap}.\, Q'$ and $Q' \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q''$ with $Q'' \models F_{\mathsf{ds}(P')}$. Now by Lemma 2.4, $\mathsf{ds}(Q) - 1 = \mathsf{ds}(Q') \geq \mathsf{ds}(Q'')$, and by induction hypothesis $\mathsf{ds}(Q'') \geq \mathsf{ds}(P') = \mathsf{ds}(P) - 1$, so that finally $\mathsf{ds}(Q) \geq \mathsf{ds}(P)$.

□

Proof:[of Lemma 5.8] Let P, Q be two terms. We reason by induction on $\mathsf{ds}(P)$ to define $F_{P,Q}$:

- if $\mathsf{ds}(P) = 0$, then $P \in \mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ (actually, P is an inactive tree, with no capability) and we may take $F_{P,Q} = \mathcal{F}_P$.
- Let us now assume that $\mathsf{ds}(P) > 0$.
  - if $\mathsf{ds}(P) > \mathsf{ds}(Q)$, we take $F_{P,Q} = F_{\mathsf{ds}(P)}$. Then $P \models F_{P,Q}$. As for any Q' and cap such that $Q \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q'$, $\mathsf{ds}(P) > \mathsf{ds}(Q) \geq \mathsf{ds}(Q')$, $Q' \not\models F_{P,Q}$ (and $Q' \not\simeq_{\mathrm{bis}} P$).
  - if $\mathsf{ds}(P) \leq \mathsf{ds}(Q)$, we want to construct formulas F_{P_i,Q_j} and F_{Q_j,P_i} for $P_i \in \mathsf{cont}(P)$ and $Q_j \in \mathsf{cont}(Q)$. We have by induction formulas F_{P_i,Q_j} for any $P_i \in \mathsf{cont}(P)$ and $Q_j \in \mathsf{cont}(Q)$. Moreover, we may also obtain formulas F_{Q_j,P_i} by induction when $\mathsf{ds}(Q_j) < \mathsf{ds}(P)$. Finally, when $\mathsf{ds}(Q_j) \geq \mathsf{ds}(P) > \mathsf{ds}(P_i)$, we may define $F_{Q_j,P_i} \stackrel{\mathrm{def}}{=} F_{\mathsf{ds}(Q_j)}$ as above.

* <u>construction of the formula</u>
Let us set

$$\mathcal{E} \stackrel{\text{def}}{=} \mathsf{cont}(P) \cup \mathsf{cont}(Q)\,.$$

$\mathcal{E}$ is finite because of Fact 2, and we may therefore define the formulas

$$F_i^{\Diamond} \stackrel{\text{def}}{=} \bigwedge_{R \in \mathcal{E}} F_{P_i,R} \qquad \text{and} \qquad F_i^{\Box} \stackrel{\text{def}}{=} \neg \bigvee_{R \in \mathcal{E} \setminus \mathcal{E}_{P_i}} F_{R,P_i}\,.$$

Then for the decomposition $P = \mathcal{C}[\widetilde{P}]$ for an active context $\mathcal{C}$ we take

$$F_{P,Q} = \mathcal{C}^*[\widetilde{F^{\Diamond}}, \widetilde{F^{\Box}}]\,.$$

* <u>A key result</u>
Let us first consider the formulas $F_i = \langle \mathsf{cap}_i \rangle.\, F_i^{\Diamond} \wedge [\![\mathsf{cap}_i]\!].\, F_i^{\Box}$. We first prove that for any $R \in \mathcal{E}$, $\mathsf{cap}_i.\, R \models F_i$ iff $\mathsf{cap}_i.\, R \simeq_{\text{bis}} \mathsf{cap}_i.\, P_i$. In particular, this will entail that F_i is sequentially selective on $\mathcal{E}$.
   · Let $R \in \mathcal{E}$ be such that $\mathsf{cap}_i.\, R \models F_i$. Then there is some R' such that $R \stackrel{\mathsf{cap}_i}{\Longrightarrow} R'$ and $R' \models F_i^{\Diamond}$. In particular, $R' \models F_{P_i,R}$, and hence (by induction) $R' \simeq_{\text{bis}} P_i$. Moreover, $R \models F_i^{\Box}$, and since $R \in \mathcal{E}$, R is in $\mathcal{E}_{P_i}$. This entails that $R \stackrel{\mathsf{cap}_i}{\Longrightarrow} \simeq_{\text{bis}} P_i \stackrel{\mathsf{cap}_i}{\Longrightarrow} \simeq_{\text{bis}} R$, and by definition of $\simeq_{\text{bis}}$, $\mathsf{cap}_i.\, R \simeq_{\text{bis}} \mathsf{cap}_i.\, P_i$.
   · Let $R \in \mathcal{E}$ be such that $\mathsf{cap}_i.\, R \simeq_{\text{bis}} \mathsf{cap}_i.\, P_i$. We want to prove that $\mathsf{cap}_i.\, R \models F_i$. By correction, we can restrict ourselves to prove this for $R = P_i$. $\mathsf{cap}_i.\, P_i \models \langle \mathsf{cap}_i \rangle.\, F_i^{\Diamond}$ trivially holds. We thus are left with proving $\mathsf{cap}_i.\, P_i \models [\![\mathsf{cap}_i]\!].\, F_i^{\Box}$. Let P' be such that $P_i \stackrel{\mathsf{cap}_i}{\Longrightarrow} P'$ and assume by absurd there exists some $R \in \mathcal{E} \setminus \mathcal{E}_{P_i}$ such that $P' \models F_{R,P_i}$. Then P' would be bisimilar to R, and $P_i \stackrel{\mathsf{cap}_i}{\Longrightarrow} \simeq_{\text{bis}} R$, which means $R \in \mathcal{E}_{P_i}$, hence a contradiction. Therefore, $P' \models F_i^{\Box}$. As this can be proved for any P', $\mathsf{cap}_i.\, P_i \models [\![\mathsf{cap}_i]\!].\, F_i^{\Box}$.

* <u>Putting it all together</u>
We now come to the proof that formula $F_{P,Q}$ introduced above satisfies the expected properties. First, we show that $P \models F_{P,Q}$: as $P = \mathcal{C}[\widetilde{P}]$, this is equivalent by Lemma 5.7 to $\mathsf{cap}_i.\, P_i \models F_i$ for any i, and is a consequence of the previous reasoning.
Consider now a capability cap and a process Q' such that $Q \stackrel{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q'$; by Lemma 5.5, $\mathsf{cont}(Q') \subseteq \mathcal{E}$. Let us moreover assume that $Q' \models F_{P,Q}$. By Lemma 5.7, this means there is a decomposition $Q' \equiv \mathcal{C}'[\widetilde{Q'}]$ and a function σ such that $\mathsf{cap}_j.\, Q'_j \models F_{\sigma(j)}$. As a consequence of the previous reasoning, we have that $\mathsf{cap}_j.\, Q_j \simeq_{\text{bis}} \mathsf{cap}_j.\, P_{\sigma(j)}$. So $Q' \sim_{\text{ind}} P$, which concludes the proof.

$\square$

This lemma allows us to derive the following important result:

Theorem 5.10 (Completeness of $\simeq_{\text{bis}}$) *In MA,* $=_L\ \subseteq\ \simeq_{\text{bis}}$.

Proof: Let P, Q be two terms such that $P \not\simeq_{\text{bis}} Q$. We apply Lemma 5.8 to processes P and Q. We have $P \models F_{P,Q}$. We then have $Q \overset{\langle \mathsf{cap} \rangle}{\Longrightarrow} Q$ (for any cap), and $Q \models F_{P,Q}$ implies $P \simeq_{\text{bis}} Q$. Hence, since by hypothesis $P \not\simeq_{\text{bis}} Q$, $Q \not\models F_{P,Q}$, and $P \neq_L Q$. $\square$

Corollary 5.11 *In MA, relations* $=_L$*,* $\simeq_{\text{bis}}$ *and* $\sim_{\text{ind}}$ *coincide. Further, on* $MA_{\text{IF}}^{\text{syn}}$*, they also coincide with* $\equiv$*.*

6 (Un)decidability of the logical equivalence

Contexts are terms containing one of several occurrences of a hole, written $\{\!|\ |\!\}$; we adopt this notation in order not to confuse the hole with an ambient construct. Given a context $\mathcal{C}$, we write $\mathcal{C}\{\!|\ P\ |\!\}$ for the process obtained by filling the hole with a term P in $\mathcal{C}$.

The definition of the Turing machine encoding below requires the introduction of some macros that will be given as contexts. To abbreviate definitions, we shall sometimes work with *parametrised contexts*, which are context definitions that depend on some values (names, words, or movements of the head of the Turing machine). Some parametrised definitions shall be written $\texttt{foo}(p); P$: such a definition depends on parameters p and P (P being a process), but we adopt this notation to emphasize sequentiality between the process being introduced and P.

6.1 Ribbons

Digits and words We associate to booleans true and false two names tt and ff. We call these names *digits*, and range over digits with d, d'. A word will be the result of a (possibly empty) concatenation of digits. The empty word shall be written ϵ. We range over words with w, w', w_1, w_2. Given a word w consisting in r digits (with $r \geq 1$), we shall sometimes write w as $w^1 \dots w^r$ to refer to w's digits.

Cells and Words		
$\mathtt{cell}(d)\{\!\mid\ \mid\!\}$	$:=$	$cell[d[\mathbf{0}] \mid !\mathsf{open}\ wo \mid \{\!\mid\ \mid\!\}\ ]$
$\mathtt{word}(w)\{\!\mid\ \mid\!\}$	$:=$	$\mathtt{cell}(w^1)\{\!\mid \mathtt{cell}(w^2)\{\!\mid \ldots \mathtt{cell}(w^r)\{\!\mid\ \mid\!\} \ldots \mid\!\} \mid\!\} \qquad (w = w^1 w^2 \ldots w^r)$
Ribbon Extensor		
$\mathtt{deadextcode}$	$:=$	$!\mathsf{open}\ coin.\ \mathsf{open}\ newcell.\ \mathsf{in}\ cell.\ coin[\mathbf{0}] \mid !newcell[\mathtt{cell}(ff)\{\!\mid \mathsf{out}\ ext \mid\!\}]$
$\mathtt{sendstart}$	$:=$	$msg[\mathsf{out}\ ext.\ !\mathsf{out}\ cell \mid \mathsf{out}\ ribbon\_left.\ start[\mathsf{in}\ TM]]$
$\mathtt{ExtensorFrozen}$	$:=$	$ext[\mathtt{deadextcode} \mid \mathsf{open}\ coin.\ \mathtt{sendstart}]$
$\mathtt{ExtensorAlive}$	$:=$	$ext[coin[\mathbf{0}] \mid \mathtt{deadextcode} \mid \mathsf{open}\ coin.\ \mathtt{sendstart}]$
$\mathtt{ExtensorDead}$	$:=$	$ext[\mathtt{deadextcode}]$
Ribbons		
$\mathtt{cleaninst}$	$:=$	$\mathsf{open}\ cleaner.\ \mathsf{open}\ runclean \mid runclean[\mathtt{deadcleancode}]$
$\mathtt{deadcleancode}$	$:=$	$!\mathsf{open}\ ff \mid !\mathsf{open}\ tt \mid !\mathsf{open}\ cell \mid !\mathsf{open}\ wo$
$\mathtt{FrozenRibbon}(w)$	$:=$	$ribbon\_left[\mathtt{cleaninst} \mid \mathtt{word}(w)\{\!\mid \mathtt{ExtensorFrozen} \mid\!\}]$
$\mathtt{GrowingRibbon}(w)$	$:=$	$ribbon\_left[\mathtt{cleaninst} \mid \mathtt{word}(w)\{\!\mid \mathtt{ExtensorAlive} \mid\!\}]$
$\mathtt{WorkRibbon}(w_1, w_2)\{\!\mid\ \mid\!\}$	$:=$	$ribbon\_left[\mathtt{cleaninst}] \mid \mathtt{word}(w_1)\{\!\mid \{\!\mid\ \mid\!\} \mid \mathtt{word}(w_2)\{\!\mid \mathtt{ExtensorDead} \mid\!\} \mid\!\}$
$\mathtt{OldRibbon}$	$:=$	$ribbon\_left[\mathtt{deadcleancode} \mid \mathtt{ExtensorDead}]$

All names used in the definitions above are supposed to be pairwise distinct. In particular, TM is the name we shall use for the ambient containing the Turing Machine (see below).

Fact 3 (Ribbon evolution) *For any word w and $n \in \mathbb{N}$, let $P_n = \mathtt{GrowingRibbon}(w.\,ff^n)$. We have:*

- $P_n \Longrightarrow P_{n+1}$*;*
- $P_n \Longrightarrow R$ *with* $R = \mathtt{WorkRibbon}(\epsilon, w.\,ff^n)\{\!\mid msg[!\mathsf{out}\ cell \mid \mathsf{out}\ ribbon\_left.\ start[\mathsf{in}\ TM]] \mid\!\}$*;*
- *for any term Q along the reduction paths (whatever these are) from P_n to P_{n+1} and from P_n to R, there exists Q' such that $Q \equiv ribbon\_left[Q']$.*

Moreover, for any word w, we have:

$$\mathtt{WorkRibbon}(w, \epsilon)\{\!\mid \mathbf{0} \mid\!\} \mid cleaner[\mathsf{in}\ ribbon\_left] \Longrightarrow \mathtt{OldRibbon}.$$

6.2 Turing Machine

Definition 6.1 ((Ideal) Turing Machine) *We introduce three symbols $\leftarrow$, $\downarrow$ and $\longrightarrow$ for the movements of the head of a Turing machine.*

We represent a Turing machine as a quadruplet $(\mathcal{Q}, q_{start}, q_A, \delta)$ where $\mathcal{Q}$ is a set of states, q_{start} is the initial state, q_A is the accepting state, and $\delta : \mathcal{Q} \times \{ff, tt\} \longrightarrow \mathcal{Q} \times \{ff, tt\} \times \{\leftarrow, \downarrow, \longrightarrow\}$ is the evolution function.

Notation: we shall write

$$(w_1, q, w_2) \rightarrowtail\!\!\!\!\rightarrow (w_1', q', w_2')$$

to denote the fact that the Turing machine in state q with the head on the cell of the last letter of w_1 (which will be referred to as *"the head dividing the ribbon into words* w_1 *and* w_2*"*) evolves in one step of computation into the machine in state q', dividing the ribbon into words w'_1 and w'_2.

Turing Machine Transition

$$
\begin{array}{lcl}
\mathtt{clear}(d); P & := & wo[\,\mathsf{out}\ head.\,\mathsf{open}\ d.\,cl\_ack[\mathsf{in}\ head]\,] \ \mid\ \mathsf{open}\ cl\_ack.\,P \\
\mathtt{write}(d); P & := & wo[\,\mathsf{out}\ head.\,d[\mathbf{0}] \ \mid\ wr\_ack[\mathsf{in}\ head]\,] \ \mid\ \mathsf{open}\ wr\_ack.\,P \\
\mathsf{become}(mo); P & := & mo[\mathsf{out}\ head.\,\mathsf{open}\ head.\,P] \ \mid\ \mathsf{in}\ mo \\
\mathsf{domove}(mv); P & := & \begin{cases} \mathsf{in}\ cell.\,P & \text{if } mv = \leftarrow \\ P & \text{if } mv = \downarrow \\ \mathsf{out}\ cell.\,P & \text{if } mv = \rightarrow \end{cases} \\
\mathtt{transcode}(d_r, q_w, d_w, mv) & := & \mathtt{clear}(d_r); \mathtt{write}(d_w); \mathsf{become}(mo); \mathsf{in}\ TM.\,\mathsf{domove}(mv); \mathsf{open}\ q_w
\end{array}
$$

State

$$
\begin{array}{lcl}
ff \longrightarrow P \ +\ tt \longrightarrow Q & := & coin[\mathsf{in}\ ff.\,\mathsf{out}\ ff.\,P] \ \mid\ coin[\mathsf{in}\ tt.\,\mathsf{out}\ tt.\,Q] \ \mid\ \mathsf{open}\ coin \\
\mathsf{code}(q) & := & !q[head[\,\mathsf{out}\ TM.\,\big(\ ff \longrightarrow \mathtt{transcode}(ff, d_{ff}, q_{ff}, mv_{ff}) \\
 & & \qquad\qquad\qquad\quad +\ tt \longrightarrow \mathtt{transcode}(tt, d_{tt}, q_{tt}, mv_{tt})\big)\,]] \\
 & & \mid\ !coin[\mathsf{in}\ ff.\,\mathsf{out}\ ff.\,\mathtt{transcode}(ff, d_{ff}, q_{ff}, mv_{ff})] \\
 & & \mid\ !coin[\mathsf{in}\ tt.\,\mathsf{out}\ tt.\,\mathtt{transcode}(tt, d_{tt}, q_{tt}, mv_{tt})] \\
\mathsf{code}(q_A) & := & !q_A[\,get\_out[\mathbf{0}]\,]
\end{array}
$$

Turing Machine Behavior after Recognition

$$
\begin{array}{l}
\mathsf{getout} := \\
\quad !\mathsf{open}\ get\_out.\,\mathsf{out}\ cell.\,get\_out[\mathbf{0}] \\
\mid\ !\mathsf{open}\ get\_out.\,\mathsf{out}\ ribbon\_left.\,\big(\ cleaner[\mathsf{out}\ TM.\,\mathsf{in}\ ribbon\_left] \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid\ coin[\mathsf{out}\ TM.\,\mathsf{in}\ ribbon\_left.\,\mathsf{in}\ cell^{\mathsf{length}(w)}.\,\mathsf{in}\ ext] \\
\qquad\qquad\qquad\qquad\qquad\qquad \mid\ \mathsf{open}\ start.\,\mathsf{in}\ ribbon\_left.\,\mathsf{in}\ cell.\,\mathsf{open}\ q_{start}\ \big)
\end{array}
$$

Our Turing Machine encoding is somehow reminiscent of the one presented in [CG98a]. This is also the case for some macros we use. We can however remark that we work here in a language without name restriction, and with a simpler encoding of choice. Note as well that this encoding is parametric over a word w, whose length (denoted $\mathsf{length}(w)$ is used in the definition of `getout`. This particularity is however quite irrelevant since it plays only a role after the end of the execution of the machine, and not inside the core of its simulation. In particular, we will see after some lemmas that we may claim the following fact:

Fact 4 *Any computation may be encoded in* $MA_{\mathrm{IF}}^{\mathrm{syn}}$

Remark: Busi and Zavattaro [BZ02] have independently obtained an encoding of Random Acces Machines in $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$ (although this sublanguage is not explicitely mentioned in the paper).

Definition 6.2 (Turing Machine in Mobile Ambients) *The encoding of a Turing Machine is based on an ambient named* TM*, containing a persistent code* *named* $\texttt{tmsoup}$*:*

$$\texttt{tmsoup} \ := \ \ \mathsf{code}(q_0) \mid \ldots \mid \mathsf{code}(q_n) \ \mid \ \mathsf{getout} \ \mid \ !\mathsf{open}\ mo\,.$$

We define two configurations for the encoding of a Turing Machine. Before being active, the machine is in starting state*, defined by:*

$$\texttt{TMStart} \ := \ \ TM[\ \mathsf{open}\ start.\,\mathsf{in}\ ribbon\_left.\,\mathsf{in}\ cell.\,\mathsf{open}\ q_{start} \ \mid \ \texttt{tmsoup}\ ]\,.$$

Once the computation has started, the Turing machine in state q is represented by the term

$$\texttt{TM}(q) \ := \ \ TM[\,\mathsf{open}\ q \ \mid \ \texttt{tmsoup}]\,.$$

Lemma 6.3 (Turing machine in $\mathbf{MA}_{\mathrm{IF}}^{\mathrm{syn}}$) *All terms used in the encoding of a Turing machine belong to $MA_{\mathrm{IF}}^{\mathrm{syn}}$.*

Definition 6.4 (deterministic evolution relation) *We say that a process P* deterministically evolves *to Q, written $P \ \rightsquigarrow \ Q$, if and only if $P \longrightarrow Q$ and for any Q' s.t. $P \longrightarrow Q'$, either $Q' \not\twoheadrightarrow$ or $Q \equiv Q'$.*

Notation: We shall write $P \rightsquigarrow^k Q$ to say that P deterministically reduces to Q in k steps ($k \geq 1$). We write $P \rightsquigarrow^* Q$ when $P \rightsquigarrow^k Q$ for some k.

The relation $P \rightsquigarrow^* Q$ captures the fact that P can only reduce to Q up to some blocking states. Actually, such blocking states may only appear due to the firing of the "wrong branch" in a choice encoding ($f\!f \longrightarrow \cdots + tt \longrightarrow \ldots$).

Lemma 6.5 (One step of Turing machine simulation) *Let $\mathcal{M}$ be a Turing machine, q one of its non accepting states, and w_1, w_2 two words, with $w_2 \neq \epsilon$. Suppose $(w_1, q, w_2) \rightarrowtail\!\!\!\!\twoheadrightarrow (w_1', q', w_2')$. Then*

$$\texttt{WorkRibbon}(w_1, w_2)\{\!|\ \texttt{TM}(q)\ |\!\} \ \rightsquigarrow^* \ \texttt{WorkRibbon}(w_1', w_2')\{\!|\ \texttt{TM}(q')\ |\!\}\,.$$

We obtain as a corollary:

Proposition 6.6 (Turing machine simulation) *Given a Turing machine $\mathcal{M}$, for any word w and $n \in \mathbb{N}$, the Turing machine $\mathcal{M}$ recognises the word w on the ribbon $w.\,f\!f^n$ iff there exist two words w_1 and w_2 s.t.*

$$\texttt{WorkRibbon}(\epsilon, w.\,f\!f^n)\{\!|\ \texttt{TM}(q_{start})\ |\!\} \ \rightsquigarrow^* \ \texttt{WorkRibbon}(w_1, w_2)\{\!|\ \texttt{TM}(q_A)\ |\!\}\,,$$

where the terms above are given by the encoding of $\mathcal{M}$.

Fact 5 (Acceptation) *Let w_1, w_2 be two words. Then*

$$\texttt{WorkRibbon}(w_1, w_2)\{\!|\ \texttt{TM}(q_A)\ |\!\}$$
$$\Longrightarrow \texttt{OldRibbon} \ \mid \ \texttt{TMStart} \ \mid \ coin[\mathsf{in}\ ribbon\_left.\,\mathsf{in}\ cell^{\mathsf{length}(w)}.\,\mathsf{in}\ ext]$$

where w is the word used in the encoding of the machine.

Remarks:

- As we already mentioned above, our encoding of the Turing machine is at this point dependent from the word w that we want it to recognize.

- We reason here with a $\Longrightarrow$ transition instead of deterministic reduction $\rightsquigarrow$: indeed, we are considering states where the machine has already recognized the word, and we only need to prove that *there exists* some way back to its (*exact*) initial state. This will be exploited later for the proof of undecidability in Sec. 6.3.

6.3 Undecidability of Logical Equivalence

We can now exploit the encoding we have studied to establish undecidability of the equivalence induced by the logic $=_L$.

Lemma 6.7 (Loop lemma) *Given a Turing machine $\mathcal{M}$ and a word w, define the following terms, given from the encoding of $\mathcal{M}$:*

$$Q := !\mathtt{FrozenRibbon}(w) \mid !\mathtt{OldRibbon} \mid !\mathsf{open}\, msg \mid !\mathsf{out}\, cell \mid \mathtt{TMStart},$$
$$P_0 := Q \mid \mathtt{GrowingRibbon}(w) \quad and \quad P_1 := Q \mid \mathtt{GrowingRibbon}(w.\mathit{ff}).$$

Then $P_0 \Longrightarrow P_1$. Conversely, $P_1 \Longrightarrow P_0$ if and only if the word w may be recognized on a finite (but sufficiently long) ribbon of the shape $w.\mathit{ff}^N$, for some $N \in \mathbb{N}$, by the Turing machine $\mathcal{M}$.

Theorem 6.8 (Undecidability of $=_L$) *$=_L$ is an undecidable relation on $\mathcal{P}$.*

Proof: Consider processes P_0 and P_1 from Lemma 6.7. We show that the problem of deciding whether $\mathsf{open}\, n.\, P_0 =_L \mathsf{open}\, n.\, P_1$ is equivalent to decide wether $P_0 \Longrightarrow P_1 \Longrightarrow P_0$. This will be enough, by Lemma 6.7, to prove the undecidalility of $=_L$.

Let us first assume that $\mathsf{open}\, n.\, P_0 =_L \mathsf{open}\, n.\, P_1$. As P_0 is in $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}} \subset \mathrm{MA}_{\mathrm{IF}}$, we may find a characteristic formula $\mathcal{F}_{P_0}$ for P_0. Then, since $\mathsf{open}\, n.\, P_0 \models \langle \mathsf{open}\, n \rangle.\, \mathcal{F}_{P_0}$, also $\mathsf{open}\, n.\, P_1 \models \langle \mathsf{open}\, n \rangle.\, \mathcal{F}_{P_0}$. The interpretation of $\langle \mathsf{open}\, n \rangle$ gives us that there exists P_1' such that $P_1 \Longrightarrow P_1' \simeq_{\mathrm{bis}} P_0$. Now by big step definition of $\simeq_{\mathrm{bis}}$, P_1' and P_0 have the same active contexts. This entails in particular that P_1' exhibits a growing ribbon $\mathtt{GrowingRibbon}(w)$ at toplevel (indeed, such a term has a –discrinating– active context of the form $ribbon\_left[\mathcal{C}_1 \mid w^1[\ldots w^r[ext[coin[\mathbf{0}] \mid \mathcal{C}_2]]\ldots]]$ for some active contexts $\mathcal{C}_1$ and $\mathcal{C}_2$). As in P_1 the growing ribbon is bigger ($w.\mathit{ff}$), this is only possible if a growing ribbon has been erased by the machine and a new one has been created, that is the word has been recognized by the machine. Using Lemma 6.7, this proves that $P_1 \Longrightarrow P_0$, and the first implication is established (since $P_0 \Longrightarrow P_1$ is obvious).

Suppose now that $P_0 \Longrightarrow P_1 \Longrightarrow P_0$. Then, by definition of intensional equivalence, $\mathsf{open}\, n.\, P_0 \simeq_{\mathrm{bis}} \mathsf{open}\, n.\, P_1$, and the converse implication holds by correction (Thm. 3.3). □

7 Extensions

7.1 Adding communication

The syntax of MA in [CG00] also includes communication, i.e., operators $\langle V \rangle$ for the emission of a value, and $(x)P$ for reception. The value V can be a name, a capability, or a path of capabilities (a string of capabilities).

The results we have presented can be extended to MA with communication of names. In the statement of the results, the main visible difference is that on $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$, $=_L$ coincides with $\equiv_{\mathrm{E}}$, the (decidable) relation obtained by adding the eta-equality

$$(x)\big(\langle x\rangle \mid (y)P\big) \quad = \quad (y)P$$

to the axioms of $\equiv$ (a similar result was known for finite MA [San01]). However, the proof techniques have to be adapted in some places, since complications appear in the treatment of the calculus. Here is a list of the most important changes.

- Definition 3.1: add the following two supplementary conditions:

  4' If $P \equiv (x)P'$ then for any n there is some Q' such that $Q \mid \langle n\rangle \Longrightarrow Q'$ and $P' \ \simeq_{\mathrm{bis}}\ Q'$.

  2' If $P \equiv \langle n\rangle$, then $Q \equiv \langle n\rangle$.

- From this definition, one can show that $\equiv_\eta \subseteq \simeq_{\mathrm{bis}}$ (and $\equiv_\eta = \simeq_{\mathrm{bis}}$ on $\mathrm{MA}_{\mathrm{IF}}^{\mathrm{syn}}$). This involves some more difficulties, since we loose the property that $P \simeq_{\mathrm{bis}} Q$ implies $\mathsf{ds}(P) = \mathsf{ds}(Q)$ (for example $(x)(\langle x\rangle \mid (y)P)$ has a sequentiality degree strictly greater than $(y)P$). However, we can recover this property if we assume P and Q to be in η-normal form.

- The idea is then to consider the calculus of terms in normal form, and extend in the end all the notions and results by $\equiv_\eta$.

- Definition 3.9 then only consider terms in η-normal forms, and its point 3 is rewriten as: *"for any j, $[\,]_{\sigma(j)}$ and $[\,]_j$ are underneath the same capability cap_j or input prefix. Moreover,*

  - *if hole $[\,]_j$ is under an input prefix, the guarded subterms are $(x)P_j$ and $(x)Q_j$ for some x, then for any fresh name n there are P'_j, Q'_j such that $P_{\sigma(j)}\{n/x\} \stackrel{\mathsf{cap}_j}{\Longrightarrow} P'_j\{n/x\} \sim_{\mathrm{ind}} Q_j\{n/x\}$, and $Q_j\{n/x\} \stackrel{\mathsf{cap}_j}{\Longrightarrow} Q'_j\{n/x\}$ and $P_{\sigma(j)}\{n/x\} \ \sim_{\mathrm{ind}}\ Q'_j\{n/x\}$.*
  - *otherwise ...* (usual clause)"

  allowing an effective decrease of the sequentiality degree.

- We may obtain similar formulas for messages, input, replicated messages and replicated input as done in [San01].

- Then the completeness may also be derived, but there is one more difficult to override involving a correct definition of $\mathsf{cont}(P)$. The only point to mention is that an input capacity may give several frozen terms, as we need to consider any replacement of the abstracted variable with any name in a message available in the term. For example, the continuation set of the term $u[\langle n_1\rangle \mid \langle n_2\rangle] \mid (x)x[\mathbf{0}]$ should be $\{n_1[\mathbf{0}], n_2[\mathbf{0}]\}$. We could then conserve the important Lemmas 2 and 5.5. However, we would be very near the limit of the tractable case, since adding name restriction in the calculus will provide (together with !) an possibly infinite ressource of names to consider, so that Lemma 2 would not have an analogous.

We believe that also the addition of communication of capabilities can be handled following [San01], but for technical reason we did not explore this way.

7.2 Other extensions

Restriction Adding restriction to our calculus, along the lines of [CG01b], would involve introducing a ν construct for processes, as well as revelation and hiding operators for formulas. This will completely simplify the way to characterise free names of the terms (see [CG01b]). However, for the technical complexity, we did not try to expand our results in such a direction.

One-step reduction The logic proposed in [CC01] develops ideas similar to the AL for the π-calculus. However, a major difference w.r.t. our setting is the presence of a *one step reduction* construct (instead of the $\diamond$ operator), which, together with a recursion operator, gives a "strong" flavour to the induced logical equivalence. We conjecture that in a similar setting (for the Ambient calculus), the problems we address in the present paper would be easier, yielding stronger results (in particular, formulas of the form $!\mathcal{A}$ without any restriction on $\mathcal{A}$) and simpler proofs. However, in such a frame, it would not be possible to derive neither the formulas for free names nor finiteness, nor for persistency, nor characteristic formulas of infinite terms. All of these properties would then be recovered, and probably sometimes in a more direct way, by adding a recursion operator in the logic.

References

[BZ02] N. Busi and G. Zavattaro. On the expressiveness of Movement in Pure Mobile Ambients. submitted, 2002.

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

[CC01] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (Part I). In *Proc. of TACS'01*, LNCS. Springer Verlag, 2001.

[CCG02] C. Calcagno, L. Cardelli, and A. Gordon. Deciding Validity in a Spatial Logic for Trees. submitted for publication, 2002.

[CG98a] L. Cardelli and A. Gordon. Mobile Ambients. In *Proc. of FOSSACS'98*, volume 1378 of *LNCS*, pages 140–155. Springer Verlag, 1998.

[CG98b] 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.

[CG98c] L. Cardelli and A.D. Gordon. Technical annex to [CG98b]. 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.D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. 27th POPL*. ACM Press, 2000.

[CG01a] L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In *Proc. of ESOP'01*, volume 2028 of *LNCS*, pages 1–22. Springer Verlag, 2001. invited paper.

[CG01b] L. Cardelli and A. Gordon. Logical Properties of Name Restriction. In *Proc. of TLCA'01*, volume 2044 of *LNCS*. Springer Verlag, 2001.

[CT01] W. Charatonik and J-M. Talbot. The Decidability of Model Checking Mobile Ambients. In *Proc. of CSL'01*, LNCS. Springer LNCS, 2001.

[CYO01] C. Calcagno, H. Yang, and P. O'Hearn. Computability and Complexity Results for a Spatial Assertion Language for Data Structures. In *Proceedings of FSTTCS '01*, volume 2245 of *LNCS*. Springer Verlag, 2001.

[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.

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

[How96] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.

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

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

[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.