Fundamenta Informaticae 64 (2005) 271–285
IOS Press

Logical Analysis of Biological Systems*

Radu Mardare†* **and Corrado Priami**
Department of Information and Communication Technology
Trento University, Italy
mardare@dit.unitn.it

Abstract. Our paper proposes a technique for performing logical analysis over the calculi for communication and mobility, i.e., Ambient Calculus type of calculi. We show how this analysis can be used in the case of biological models in order to obtain significant information for biologists.

The technique is based on set theoretical models we developed for ambient processes by using the power of Hypersets Theory. These models are further used as possible worlds in a Kripke structure organized for a propositional branching temporal logic.

Providing the temporal logical structure for the accessibility relation between ambient processes, we open the perspective of reusing model checking algorithms developed for temporal logics in analyzing any phenomena that can be described by these calculi.

Keywords: Hypersets, process algebra, ambient calculus, temporal logics, model checking

1. Introduction

Ambient Calculus [11] is a useful tool to construct mathematical models for complex systems because of its facilities in expressing hierarchies of locations and their mobility. It was developed first as a natural extension, with locations, of Π-Calculus [22, 24], in order to provide models for phenomena concerning communication and mobility. Later, it was adapted to model biological systems [28, 9], as an algebraical alternative to the Membrane Computing approach [25].

*Work partially supported by the FET project IST-2001-32072 DEGAS under the pro-active initiative on Global Computing.
†Address for correspondence: Via Sommarive 14, I-38050 POVO, Trento (TN) – Italy
*Also work: Bucharest University, Romania

Even if such a calculus is able to simulate the behavior of some biological systems, for giving to biologists a real useful tool, we need to do more. It could be useful to develop some techniques for making predictions over the systems, such that the results of some expensive biological experiments to be predicted by some computable formal analysis of the mathematical models. Properties such as *the protein will split*, or *there is a possible future where the complexAB precedes the proteinA* are not expressible inside Ambient Calculus. Only a logic built on top of it can describe such phenomena.

In order to answer such requirements Ambient Logic [12, 13] and Spatial Logics [10] were developed. These logics can describe properties of mobile computations as well as the hierarchy of locations and modifications of this hierarchy in time. The main idea of these is to treat processes as spatio-temporal entities, thus two types of modalities have been used – one for assertions about space and the other for assertions about time.

This paper, resuming our results presented in [19, 20], proposes a propositional branching temporal logic, CTL^*, constructed on top of Ambient Calculus, as a more convenient alternative to Spatial Logics. Temporal logics have emerged, lately, in many domains as a good compromise between expressiveness and abstraction. Many of them support useful computational applications as model checking. For the particular cases of CTL or CTL^*, these techniques were developed up to the construction of some tools able to perform such analysis (see, e.g., SMV [4], NuSMV [2], HyTech [1], VIS [5]). The main feature of our logic is that the final state of any computation can be reconstructed by just having information about the initial state and the history of the computation. The spatial structure of a state is fully described by a set of atomical propositions, while the possible states are described using, in addition, a temporal modality. In this respect our approach is different from those used in Ambient Logic, or Spatial Logics, giving us the advantages of simplicity and expressivity that a CTL* logic has with respect to the cited modal logics. Moreover, seems unproblematic to extend the same sort of logic for other calculi in this paradigm, e.g., BioAmbients Calculus [28], or Brane Calculi [9].

Being the peculiarities of the calculi of communication, it was more profitable for us to develop the logic not directly for Ambient Calculus, but for a set theoretical model of it. This model goes further with Peter Aczel's idea of developing a model for Milner's calculus of communication systems CCS [21]. Aczel's try was finalized with the development of Hypersets Theory [6], and we found it useful to use these hypersets to model communication systems with locations, as Ambient Calculus.

The rest of the paper is organized as follows. We introduce first the Ambient Calculus and we discuss some special features of it. In the third section we present a couple of simple case studies coming from biology. They are used to comment on the advantages of applying temporal logics to the Ambient Calculus specification of phenomena related to life sciences. Section four introduces the theoretical underpinning of our logic: *labelled syntax trees* which are the graph presentation of the set theoretical model for the ambients. In Section 5 we define a branching temporal logic for Ambient Calculus, and show how to run simple reachability properties on our case studies. The final section concludes the presentation with some general remarks and a sketch of the future research directions we intend to take.

2. Overview on Ambient Calculus

First, we briefly recall the Ambient Calculus [11] starting with the syntax of ambient processes.

P, Q, R::=	**processes**	M::=	**capabilities**
$(\nu n)P$	restriction	n	name
0	void	$in\ M$	can enter into M
$P\|Q$	composition	$out\ M$	can exit out of M
$!P$	replication	$open\ M$	can open M
$M[P]$	ambient	$M.M'$	path
$M.P$	capability action	ε	null
$(n).P$	input action		
$\langle M\rangle$	output action		

We accept, in addition to the previous definition, that ambient programs can include unspecified processes denoted by capital letters P, Q and R, hereafter named *atomical processes*[1]. Let Π_P be the class of atomical processes and Π_A the class of ambients.

A structural congruence is defined over processes by:

1. $P \equiv P$
2. $P \equiv Q \Rightarrow Q \equiv P$
3. $P \equiv Q, Q \equiv R \Rightarrow P \equiv R$
4. $P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q$
5. $P \equiv Q \Rightarrow P|R \equiv Q|R$
6. $P \equiv Q \Rightarrow !P \equiv !Q$
7. $P \equiv Q \Rightarrow n[P] \equiv n[Q]$
8. $(\nu n)(m[P]) \equiv m[(\nu n)P],\ \ n \neq m$
9. $!(P|Q) \equiv !P|!Q$
10. $!0 \equiv 0$
11. $!!P \equiv !P, !P \equiv P|!P$
12. $P \equiv Q \Rightarrow M.P \equiv M.Q$
13. $P \equiv Q \Rightarrow (n).P \equiv (n).Q$
14. $P \equiv \varepsilon.P$
15. $(M.M').P \equiv M.M'.P$
16. $(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$
17. $(\nu n)0 \equiv 0$
18. $(\nu n)P|Q \equiv P|(\nu n)Q, n \notin fn(P)$
19. $P|0 \equiv P$
20. $(P|Q)|R \equiv P|(Q|R)$
21. $P|Q \equiv Q|P$
22. $(x).P \equiv (y).P(x \leftarrow y)$ if $y \notin fn(P)$

In addition, we identify processes up to renaming of bound names:

23. $(\nu n)P = (\nu m)P(n \leftarrow m)$ if $m \notin fn(P)$

Finally, the operational semantics of the ambient calculus is defined by the rules:

$$n[in\ m.P|Q]|m[R] \rightarrow m[n[P|Q]|R] \qquad P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$$
$$m[n[out\ m.P|Q]|R] \rightarrow n[P|Q]|m[R] \qquad P \rightarrow Q \Rightarrow P|R \rightarrow Q|R$$
$$open\ n.P|n[Q] \rightarrow P|Q \qquad P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$$
$$(n).P|\langle M\rangle \rightarrow P\{n \leftarrow M\} \qquad P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$$

[1]This is a necessary requirement in developing complex analysis, as model checking, for Ambient Calculus because we have to recognize and distinguish over time, unspecified processes inside the target process. For instance, P is an unspecified process in $n[in\ m.P]$

2.1. Handling the new names

Consider the next process:

$$k'[open\ k.k''[Q]]|(\nu n)n[k[out\ n.in\ k'.in\ n.0]|open\ k'.open\ k''.P]|n[R]|open\ n.t[S]$$

Here, (νn) means that the name n inside the scope of (νn) is different from all the other names in the program. In the example, we want to be sure that $out\ n$ and $in\ n$, which are capabilities prefixing the process 0, will never act over $n[R]$ but only over the ambient that was chosen to name the firewall. Vice versa, $open\ n$, the capability of t, will never act over the firewall ambient, but only over $n[R]$.

The intuition is that the name of the ambient chosen to name the firewall should be one that has not been used before. A possible solution could be just to choose a new name $r \in \Lambda$ and to replace n with it in all its occurrences inside the scope of (νn). This solution is locally good, but it will not prevent the name r from ever being used in other processes that we could combine with ours, so that a name conflict would arise. In other words, the renaming solution is not a compositional one. We guarantee compositionality by a trick that resembles de Bruijn indexes for name-free λ-calculus: we accept ordered pairs of natural numbers as possible names of ambients and we use them to completely remove any (νn) occurrence from processes. So, we replace the k^{th} new name (νn) in a process with[2] the pair $\langle k, 1\rangle$. This approach allows us to combine our process with others for which we already constructed the labelled syntax trees. In this way all the names in the second process will receive names as $\langle k, 2\rangle$ meaning that is the k^{th} new name of the second process, and so on, the k^{th} new name of the l^{th} process will receive the name $\langle k, l\rangle$.

This construction is supported by the assumption that inside an ambient process can only occur a finite number of new name operators and that we will combine only a finite number of processes.

According with the above, our example becomes:

$$k'[open\ k.k''[Q]]|\langle 1,1\rangle[k[out\ \langle 1,1\rangle.in\ k'.in\ \langle 1,1\rangle.0]|open\ k'.open\ k''.P] \\ |n[R]|open\ n.t[S]$$

The analysis of the reductions of our process shows that the expected result is still possible without using the new name operator. Indeed:

$$\begin{aligned} &k'[open\ k.k''[Q]]|\langle 1,1\rangle[k[out\ \langle 1,1\rangle.in\ k'.in\ \langle 1,1\rangle.0]|open\ k'.open\ k''.P]|n[R]|open\ n.t[S] \\ &\to^* k'[open\ k.k''[Q]|k[in\ \langle 1,1\rangle.0]]|\langle 1,1\rangle[open\ k'.open\ k''.P]|n[R]|open\ n.t[S] \\ &\to^* k'[k''[Q]|in\ \langle 1,1\rangle.0]|\langle 1,1\rangle[open\ k'.open\ k''.P]|n[R]|open\ n.t[S] \\ &\to^* \langle 1,1\rangle[k'[k''[Q]]|open\ k'.open\ k''.P]|n[R]|open\ n.t[S] \\ &\to^* \langle 1,1\rangle[Q|P]|n[R]|open\ n.t[S] \\ &\to^* \langle 1,1\rangle[Q|P]|R|t[S] \end{aligned}$$

Hereafter we will treat $\langle l, k\rangle$ as being an ambient name, whenever it appears in our processes. This means that the set Λ contains, as a subset, a subset of $\mathbb{N} \times \mathbb{N}$. This modification does not affect the four rules of structural congruence ((Struct Res Res), (Struct Res Par), (Struct Res Amb) and (Struct Zero Res), see [11]). Only it modifies the intentional interpretation of (νn). It will not mean *this name is new inside the scope of our quantifier*, but *replace this name in all its occurrences inside the scope of our quantifier by an unused pair of natural numbers*.

In this way we reduce all the syntax trees of ambient calculus to syntax trees without new name operators.

[2]We will replace in the ambient calculus process, all the occurrences of n inside the scope of (νn), being ambients or capabilities, with $\langle k, 1\rangle$

3. Using Ambient Calculus to model biological phenomena

In the cited literature, many examples of biological phenomena described using Ambient Calculus type of calculi can be found . In this section we will present two such examples, described in [20], that we found relevant for the advantages of a logical analysis.

In the first example we will describe a part of the scene of the interaction between a Virus and a Macrophage. The Macrophage, by its structure, is able to recognize a Virus by its characteristics. Once it is recognized, the Virus is engulfed by Macrophage and is destroyed. We find it appropriate to describe the Macrophage as an ambient named Mac that contains a process $Digest$ able to destroy the virus; the virus is an ambient v that contains inside a process $Infect$. The Macrophage recognizes the virus by the name v and by its structure (i.e., Macrophage knows the names k, k' that define the structure of the virus). Using this information, Macrophage manages to put in parallel the processes $Infect$ and $Digest$ and in this way annihilates the action of the virus. We can describe this action in Ambient Calculus in a way similar with the description of the action of a firewall [19]:

$$Macrophage \stackrel{def}{=} Mac[k[out\ Mac.in\ v.in\ Mac.0]|open\ v.open\ k'.Digest]$$
$$Virus \stackrel{def}{=} v[open\ k.k'[Infect]]$$
$$Virus|Macrophage \equiv$$
$$v[open\ k.k'[Infect]]|Mac[k[out\ Mac.in\ v.in\ Mac.0]|open\ v.open\ k'.Digest]$$
$$\rightarrow^* v[open\ k.k'[Infect]]|k[in\ v.in\ Mac.0]|Mac[open\ v.open\ k'.Digest]$$
$$\rightarrow^* v[open\ k.k'[Infect]|k[in\ Mac.0]]|Mac[open\ v.open\ k'.Digest]$$
$$\rightarrow^* v[k'[Infect]|in\ Mac.0]|Mac[open\ v.open\ k'.Digest]$$
$$\rightarrow^* Mac[v[k'[Infect]]|open\ v.open\ k'.Digest]$$
$$\rightarrow^* Mac[k'[Infect]|open\ k'.Digest]$$
$$\rightarrow^* Mac[Infect|Digest]$$

For this situation, we are interested in the success of our system, in all possible time paths, to achieve the state where the processes $Infect$ and $Digest$ are in parallel inside the ambient Mac (that represent the Macrophage), such that the virus is annihilated. If our system succeeds to do this, we can say that is an appropriate model for the biological phenomenon, otherwise we have to reconsider our approach. Such properties, we will argue further, can be naturally expressed using a temporal logic.

Consider now the second example, the model of *the trimetric GTP binding proteins (G-proteins)* that plays an important role in the signal transduction pathway for numerous hormones and neurotransmitters [3, 7]. It consists of five processes: a regulatory molecule RM, a receptor R, and three domains that are bound together composing the protein α, β and γ. Data sent by RM to R determine a communication between the receptor R and the protein that causes the breakage of the boundary of α, β and γ. We can express this in Ambient Calculus by the following specification:

$$RM \stackrel{def}{=} open\ n.RM,\ R \stackrel{def}{=} n[\langle GTP\rangle|R],$$
$Protein \stackrel{def}{=} (GDP)(\alpha|\beta|\gamma)$, where GDP is a name that appear in α only, bounded by the input prefix
$$RM|R|Protein \equiv open\ n.RM \mid n[\langle GTP\rangle|R] \mid (GDP)(\alpha|\beta|\gamma) \rightarrow$$
$$RM \mid R \mid \langle GTP\rangle \mid (GDP)(\alpha|\beta|\gamma) \rightarrow$$
$$RM|R|(\alpha|\beta|\gamma)(GDP/GTP) \rightarrow$$
$$RM|R|(\alpha)(GDP/GTP)|\beta|\gamma$$

where we denoted the process obtained by substituting GDP with GTP inside α by $(\alpha)(GDP/GTP)$.

In this example we accepted recursive definitions in Ambient Calculus. This pushed us outside the classical syntax of this calculus. Still the model and the logic we will introduce further can handle with such an extension. Anyway, if we really want to describe the model strictly in the classical syntax, we could differentiate the recursion variables by using some indexes.

With respect to the above example we are interested in expressing properties like *for all possible future paths, sometime in the future, we will have the interaction that will generate the split of the protein*. One might also want to express that the protein will not be split before the interaction between R and RM will be performed (*a property will not be satisfied until another one will be*). Both properties above are examples of 'temporal' properties.

4. The labelled syntax trees

In order to define the temporal logic, we reorganize the spatio-temporal information contained by an ambient process. This will be done by defining a special labelling function for the syntax trees of Ambient Calculus.

A syntax tree $S = (S, \rightarrow_S)$ for a process is a graph with $S = \Pi \cup Cap \cup Syn = (\Pi_P \cup \Pi_A) \cup Cap \cup Syn$ where

- Π is a set that contain all the unspecified process nodes (the atomical processes collected in the subset Π_P) and the ambient nodes (collected in the subset Π_A);
- Cap is the set of capability nodes (we include here the input nodes and the nodes of variables over capabilities as well); and
- Syn is the set of syntactical operator nodes (this set contains the parallel operators $|$ and the prefix operators, $\bullet$). We identify the subset $Syn' = \{\bullet_1 \in Syn \mid \bullet_1 \rightarrow_S \mid \} \subseteq Syn$ of the prefix nodes that are immediately followed, in the syntax tree, by the parallel operator because they play an important role in the spatial structure of the ambient process[3].

We consider also the possibility of having circular branches in our trees, when recursive definitions are involved. All the further discussion is including these cases as well.

The intuition behind the construction of a labelled syntax tree is to associate to each node of the syntax tree some labels by two functions: id that gives to each node an identity, and sp that registers the spatial position of the node.

The identity function id associates a label (urelement or $\emptyset$):

1. to each unspecified process and to each ambient; this label will identify the node and will help us further to distinguish between processes that have the same name,
2. to each capability, the identity of the process in front of which this capability is placed,
3. $\emptyset$, to each syntactical node.

[3]These point operators are those that connect a capability with a process formed by a parallel composition of other processes bounded together by brackets, hereafter *complex processes*, as in $c.(P|Q)$

The spatial function sp associates:

1. to each ambient the set of the identities of its children[4], while to unspecified processes associates the id-label,
2. to each capability, a natural number that counts the position of this capability in the chain of capabilities (if any) belonging to the same process,
3. to each syntactical node the spatial function associates 0, except for the nodes in Syn' to which the function sp will associate the set of identities of the processes connected by the main parallel operator in the compound process that this point is prefixing; for example, in the situation $c.(P|Q)$, $sp(\bullet) = \{id(P), id(Q)\}$.

In what follows, we choose to work in Zermelo-Fraenkel system of Set Theory ZFA with the Anti-Foundation Axiom (AFA), as being a fertile field that offers many tools for analyzing structures, as argued in [8]. This approach allows us to describe the spatial structure of ambient processes as equations in set theory, each such equation being then used as atomical proposition in our logic. In this way we will not use a modality in describing the hierarchy of locations, as Spatial Logics does, but only in describing the evolution of the hierarchy in time. Hereafter, we assume a class $\mho$ of urelements, set-theoretical entities which are not sets (they do not have elements) but can be elements of sets. The urelements together with the empty set $\emptyset$ will generate all the sets we will work with (sometimes sets of sets).

Definition 4.1. A set a is *transitive* if all the elements of a set b, which is an element of a, also belong to a: $\forall b \in a$ if $c \in b$ then $c \in a$.
The *transitive closure* of a, denoted by $TC(a)$ is the smallest transitive set including a. The existence of $TC(a)$ could be justified as follows: $TC(a) = \cup\{a, \cup a, \cup \cup a, ...\}$.

Definition 4.2. The *support of a set* a, denoted by $supp(a)$ is $TC(a) \cap \mho$. The elements of $supp(a)$ are the urelements that are *somehow involved* in a.

Definition 4.3. If $a \subseteq \mho$ then $V(a) \stackrel{def}{=} \{b \mid b \text{ is a set and } supp(b) \subseteq a\}$. $V(a)$ is the class of all sets in which the only urelements that are somehow involved are the urelements of a.

Definition 4.4. Let $S_P = (S, \rightarrow_S)$ be the syntax tree associated with the ambient process P. We call *the structure graph* associated with P, the graph obtained by restricting the edge relation of the syntax tree to $\Pi \cup Syn'$, i.e., the graph $T_P = (\Pi \cup Syn', \rightarrow_T)$ defined by: for $n, m \in \Pi \cup Syn'$, $n \rightarrow_T m$ iff ($n \rightarrow_S^* m$ and does not exist $p \in \Pi \cup Syn'$ such that $n \rightarrow_S^* p \rightarrow_S^* m$).

Intuitively, the structure graph of a process is obtained by restricting the edge relation of its syntax tree to Π.

Definition 4.5. A *decoration* of a graph $G = (G, \rightarrow_G)$ is an injective function $e : G \rightarrow V(\mho) \cup \mho$ such that for all $a \in G$ we have:

[4]We use the terms *parent* and *child* about processes, meaning the immediate parent and immediate child in Ambient Calculus processes.

- if does not exist $b \in G$ such that $a \to_G b$, then $e(a) \in \mho$,
- if $\exists b \in G$ such that $a \to_G b$, then $e(a) = \{e(b) |$ for all b such that $a \to_G b\}$.

We now introduce a set of auxiliary functions that are the block definitions for id and sp.

Definition 4.6. Let the next functions be defined on the subsets of nodes of the syntax tree $(S, \to)$ as follows:

- Let $sp_\Pi : \Pi \cup Syn' \to V(\mho) \cup \mho$ be a decoration of the structure graph associated with our syntax tree.
- Let $id_\Pi : \Pi \to \mho$ be an injective function such that $id_\Pi(P) = sp_\Pi(P)$ for all $P \in \Pi_P$. Consider $U_P \stackrel{def}{=} id_\Pi(\Pi_P) \subset \mho$, $U_A \stackrel{def}{=} id_\Pi(\Pi_A) \subset \mho$.
- Let $sp_{Syn} : Syn \to \mho \cup V(\mho) \cup \mathbb{N}$ defined by ($\mathbb{N}$ is the class of natural numbers)

$$sp_{Syn}(s) = \begin{cases} sp_\Pi(s) & \text{iff } s \in Syn', \\ 0 & \text{iff } s \in Syn \setminus Syn'. \end{cases}$$

  Consider $O \stackrel{def}{=} sp_{Syn}(Syn') \subset V(\mho)$.
- Let $id_{Syn} : Syn \to V(\mho) \cup \mho$ defined by

$$id_{Syn}(s) = \emptyset.$$

- Let $sp_{Cap} : Cap \to \mathbb{N}$ such that

$$sp_{Cap}(c) = \begin{cases} 1 & \text{iff } | \to \bullet \to c \text{ or } n \to \bullet \to c \text{ with } n \in \Pi, \\ k+1 & \text{iff } \bullet_1 \to \bullet_2 \to c \text{ and } \bullet_1 \to c' \in Cap \text{ with } sp_{Cap}(c') = k. \end{cases}$$

- Let $id_{Cap} : Cap \to V(\mho) \cup \mho$ defined for $c \in Cap$ such that $\bullet_c \to c$ by

$$id_{Cap}(c) = \begin{cases} id_\Pi(n) & \text{iff } \bullet_c \to n \text{ with } n \in \Pi, \\ id_{Cap}(c') & \text{iff } \bullet_c \to \bullet' \text{ with } \bullet' \to c', \\ sp_{Syn}(\bullet_c) & \text{iff } \bullet_c \in Syn'. \end{cases}$$

Summarizing, we can define the identity function $id : \Pi \cup Cap \cup Syn \to \mho \cup V(\mho)$ and the spatial function $sp : \Pi \cup Cap \cup Syn \to \mho \cup V(\mho) \cup \mathbb{N}$ by:

$$id(s) = \begin{cases} id_\Pi(s) & \text{iff } s \in \Pi, \\ id_{Cap}(s) & \text{iff } s \in Cap, \\ id_{Syn}(s) & \text{iff } s \in Syn, \end{cases} \qquad sp(s) = \begin{cases} sp_\Pi(s) & \text{iff } s \in \Pi, \\ sp_{Cap}(s) & \text{iff } s \in Cap, \\ sp_{Syn}(s) & \text{iff } s \in Syn. \end{cases}$$

Observe that while the range of id is $\mathfrak{V} \cup V(\mathfrak{V})$, the range of sp is $\mathfrak{V} \cup V(\mathfrak{V}) \cup \mathbb{N}$ (we consider here natural numbers as cardinals[5] so that no structure anomaly emerges as long as $\mathbb{N} \subset \mathfrak{V} \cup V(\mathfrak{V})$).

We identify the sets U_A of urelements chosen for ambients, U_P of urelements chosen for atomical processes, and the set of sets of urelements O that contain all the addresses of the elements in Syn.

We now define *labelled syntax tree* for a given syntax tree of an ambient process.

Definition 4.7. Let $S_P = (S, \to)$ be the syntax tree of the ambient process P. We call *the labelled syntax tree* of it the triplet $Sl_P = (S, \to, \phi)$, where ϕ is the function defined on the nodes of the syntax tree, by

$$\phi(s) = \langle id(s), sp(s) \rangle \text{ for all } s \in S.$$

Remark 4.1. It is obvious the central position of the function id in the previous definitions. For a particular ambient process, once we defined the function id, all the construction, up to the labelled syntax tree, can be done inductively on the structure of the ambient process. Because of this, our construction of the labelled syntax tree is unique up to the choice of urelements (i.e., of U_P and U_A).

Definition 4.8. For a given labelled syntax tree $Sl = (S, \to, \phi)$ we define the functions:

- $ur : \Pi \cup Syn' \to U_P \cup U_A \cup O$ by:

$$ur(s) = \begin{cases} id(s) & \text{if } s \in \Pi, \\ sp(s) & \text{if } s \in Syn'. \end{cases}$$

  This function associates to each node of the structure graph the set-theoretical identity defined by the labelled syntax tree.

- Let $e : U_P \cup U_A \cup O \to \mathfrak{V} \cup V(\mathfrak{V})$ be the function defined by

$$e(\nu) = sp(ur^{-1}(\nu)).$$

  It associates to each ambient and compound process the set of addresses of its children.

- $f : U_P \cup U_A \cup O \to \Lambda \cup \Pi$, where Λ is the set of names of ambients of Ambient Calculus, and Π is the set of atomical processes. For each $\nu \in U_P \cup U_A \subset \mathfrak{V}$, $f(\nu)$ is the name of the process with which ν is associated by id[6], and $f(\nu) = \langle 0, 0 \rangle$ if $\nu \in O$. By the function f each urelement (or set of urelements) used as identity will receive the name of the ambient or atomical process that it is pointing to (the sets receive the name $\langle 0, 0 \rangle$).

[5] Informally, we treat 0 as $\emptyset$, 1 as $\{\emptyset\}$, 2 as $\{\emptyset, \{\emptyset\}\}$, 3 as $\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$ and so on.

[6] Informally we could say that, on $U_A \cup U_P$, we have $f = id^{-1}$, but this is not exact for the reason that id is an injective function while f is not. Because if we have two processes named P, then, for both, the value by f will be P, but, by id^{-1}, they point to different nodes in the syntax tree.

- $F : U_P \cup U_A \cup O \to Cap^*$ for each $\nu \in U_P \cup U_A \cup O$, $F(\nu) = \langle c_1, c_2, ...c_k \rangle$ where $c_i \in Cap$ such that $\forall i \in \mathbb{N},\ id(c_i) = \nu,\ sp(c_i) = i$ and does not exist $c_{k+1} \in Cap$ such that $id(c_{k+1}) = \nu$ and $sp(c_{k+1}) = k + 1$. In the case that, for ν we cannot find any such c_i, we define $F(\nu) = \langle \varepsilon, \varepsilon, ... \rangle$, ε being the null capability. We adopt the following enrichment of the relation of equality on capability chains $=_{Cap}$ defined by the next rules [7]:
  - $\langle c_1, c_2, c_3, ...c_n \rangle - \langle c_1 \rangle =_{Cap} \langle c_2, c_3, ...c_n \rangle$,
  - $\langle \varepsilon, c_1, ..., c_k \rangle =_{Cap} \langle c_1, c_2, ..., c_k, \varepsilon \rangle =_{Cap} \langle c_1, ..., c_t, \varepsilon, c_{t+1}, ...c_k \rangle =_{Cap} \langle c_1, c_2, ..., c_k \rangle$,
  - $\langle \varepsilon, \varepsilon, ...\varepsilon \rangle =_{Cap} \emptyset$.

The function F associates with each of these the list of capabilities that exists in front of the process they point to.

Definition 4.9. Let $S = (S, \to, \phi)$ be a labelled syntax tree of the ambient process P. We will call the *canonical labelled syntax tree* associated with P, denoted by $S^+ = (S^+, \to_+, \phi_+)$, the restriction of the labelled syntax tree to the set $S^+ = \{n |\ n \in S,\ f(n) \neq 0\ and\ F(n) \neq \langle \varepsilon, ...\varepsilon \rangle\}$, where 0 is the null process and ε is the null capability.

Further we will analyze only canonical labelled trees (by extension canonical processes), these being those which evolve during the ambient calculus computations, therefore they are those which really matter for our purpose. Always, we consider ambient processes enclosed in a *master ambient* which stays for the environment. Being the reduction rule $P \to Q \Rightarrow n[P] \to n[Q]$, there is no danger in doing this.

Other aspects concerning the definition of the labelled syntax tree for situations that involves the new name operator, the replication operator, or recursive processes can be found in [18]. Also we introduce an algebra of labelled trees in order to analyze their composition.

In [18] we proved that the function that associates to each ambient process the set $\langle U_P, U_A, O, e, f, F \rangle$ is generating a sound model for Ambient Calculus. Moreover, the tuple $\mathcal{E} = \langle U_A \cup O, U_P, e \rangle$ satisfies the requirements of the definition of a *flat system of equations* which can describe a hyperset uniquely up to bisimulation relation, see [8]. For this reason we can interpret the tuple $\langle U_P, U_A, O, e, f, F \rangle = \langle \mathcal{E}, f, F \rangle$ as a labelled flat system of equations. This is the set theoretical model for communication and mobility calculi. Such a result is important from a few points of view. First we are confident in the possibility to extend this model to all Milner's bigraphs [23] that are seen now as the only model of communication systems. Second, replacing the structure of membership relation e in the definition of the flat system of equations with a relation defined by a fuzzy set structure we hope to provide a model for any calculus of communication involving stochastic information, such as [26, 27].

5. The Logic

The logic we construct is a branching propositional temporal logic[8], CTL^*. The requirements for such a construction [14] are to organize a structure $\mathcal{M} = (S_0, \Sigma, \Re, \mathcal{L})$, where S_0 is the initial state of our

[7] These rules are allowed by the syntax of Ambient Calculus together with the rules of structural congruence over processes.
[8] We choose CTL^* because is more expressive then CTL, but a CTL is possible as well

model, Σ is the class of all possible states in our model, $\Re$ is the accessibility relation between states, $\Re \subseteq \Sigma \times \Sigma$, and $\mathcal{L} : \Sigma \to \mathcal{P}(\mathcal{A})$ is a function which associates to each state $S \in \Sigma$ a set of atomical propositions $\mathcal{L}(S) \subseteq \mathcal{P}(\mathcal{A})$ – the set of the atomical propositions true in the state S ($\mathcal{A}$ will be the class of atomical propositions and $\mathcal{P}$ the power-set operator).

We propose to use the ordered sets $S = \langle U_A, U_P, O, e, f, F\rangle$ as states in our logic. The choice of the initial state should depend on the purpose of our analysis. If we are interested in the future of an ambient calculus process P by itself, then its ordered set will be the initial state. But if P will interact with another process Q, or will become a child of an ambient, or both like in $m[P|Q]$, then, even if we have a particular interest in P, the initial state should be the ordered set of $m[P|Q]$. For this purpose we defined computation operations over these ordered sets to be able, starting from the sets constructed for some initial processes, to obtain the sets for other processes constructed on top of these (for more see [18]).

The construction of Σ should be done in such a way to contain all the possible future states of the initial state. For this reason we take

$$\Sigma = \{S_i = \langle U_A^i, U_P^i, O^i, e_i, f_i, F_i\rangle \mid U_A^i = U_A^0,\ U_P^i = U_P^0,\ and\ O_i = O_0\},$$

where $S_0 = \langle U_A^0, U_P^0, O_0, e_0, f_0, F_0\rangle$ is the initial state. The intuition is that no matter how the process will evolve, it is not possible to appear in it new elements than those that already exist in the initial state[9].

Our main idea is to define the atomic propositions such that they express the basic equations that define the spatial relations between parts of our process. So, we could define the set of atomical propositions as:

$$\mathcal{A} = \{x\mathbf{in}y | x \in U_P \cup U_A \cup O\ and\ y \in U_A \cup O\}.$$

In our logic we want $x\mathbf{in}y$ to be just an atomical proposition and x, y just letters. The cardinality of $\mathcal{A}$ is $card(U_P \cup U_A \cup O) \times card(U_A \cup O)$ which depends (polynomially) on the number of atomical processes and ambients in the ambient calculus process S_0.

Further, the interpretation function $\mathcal{L} : \Sigma \to \mathcal{P}(\mathcal{A})$ is defined by:

$$\mathcal{L}(S) = \{x\mathbf{in}y \mid x \in e_y \text{ or } e_x \in e_y\}.$$

As it concerns the accessibility relation $\Re \subseteq \Sigma \times \Sigma$, following the previous intuition we could define it for two states S_0 and S_1, constructed for the processes P_0 and P_1, by $\langle S_0, S_1\rangle \in \Re$ iff $P_0 \to P_1$ (i.e., P_1 can be reached from P_0 in one step of ambient calculus reduction).

5.1. Syntax

Further, we could introduce the syntax of the CTL* logic [14]. We inductively define a class of state formulae (which will be true or false of states) and a class of path formulae[10] (true or false on paths), starting from $\mathcal{A}$. We accept as basic operators the logical operators $\wedge$ and $\neg$, the temporal operators X (*next time*) and $\cup$ (*until*) and the path quantifier E (*for some futures*). We will derive from them all the

[9] We include here also the situations where some ambients were dissolved by consuming, for example, *open* capability; we consider, in this case, that these ambients still exist in our process but they have an "empty position".

[10] A *fullpath* is an infinite sequence $S_0, S_1, ...$ of states such that $(S_i, S_{i+1}) \in \Re$ for all i. We use the convention that if $x = (S_0, S_1, ...)$ denotes a fullpath, then x^i denotes the suffix path $(S_i, S_{i+1}, S_{i+2}, ...)$.

usual propositional logic operators, the temporal operators G (*always*) and F (*sometimes*) and the path quantifier A (*for all futures*).

Syntactical rules:

1. Each atomical proposition $\alpha \text{in} \beta \in AP$ is a state formula.
2. If p, q are state formulae then so are $p \wedge q$, $\neg p$.
3. If p is a path formula then $E\, p$, $A\, p$ are state formulae.

1'. Each state formula is a path formula.

2'. If p, q are path formulae then so are $p \wedge q$, $\neg p$.

3'. If p, q are path formulae then so are Xp, $p \cup q$.

Syntactical conventions:

1. Ap abbreviates $\neg E \neg p$.
2. EFp abbreviates $E(true \cup p)$.
3. AGp abbreviates $\neg EF \neg p$.
4. AFp abbreviates $A(true \cup p)$.
5. EGp abbreviates $\neg AF \neg p$.

5.2. Semantics

Now we define $\models$ inductively. We write $\mathcal{M}, S_0 \models p$ to mean that the state formula p is true at state S_0 in the model $\mathcal{M}$, and $\mathcal{M}, x \models p$ to mean that the path formula p is true for the fullpath x in the structure $\mathcal{M}$. The rules are:

$\mathcal{M}, S_0 \models P$ iff $P \in \mathcal{L}(S_0),$ where $P \in \mathcal{A}$

$\mathcal{M}, S_0 \models p \wedge q$ iff $\mathcal{M}, S_0 \models p$ and $\mathcal{M}, S_0 \models q$

$\mathcal{M}, S_0 \models \neg p$ iff it is not the case that $\mathcal{M}, S_0 \models p$

$\mathcal{M}, S_0 \models Ep$ iff $\exists$ fullpath $x = (S_0, S_1, ...)$ in $\mathcal{M}$ with $\mathcal{M}, x \models p$

$\mathcal{M}, S_0 \models Ap$ iff $\forall$ fullpath $x = (S_0, S_1, ...)$ in $\mathcal{M}$ with $\mathcal{M}, x \models p$

$\mathcal{M}, x \models p$ iff $\mathcal{M}, S_0 \models p$

$\mathcal{M}, x \models p \wedge q$ iff $\mathcal{M}, x \models p$ and $\mathcal{M}, x \models q$

$\mathcal{M}, x \models \neg p$ iff it is not the case that $\mathcal{M}, x \models p$

$\mathcal{M}, x \models p \cup q$ iff $\exists i \left(\mathcal{M}, x^i \models q \text{ and } \forall j \left(j < i \text{ implies } \mathcal{M}, x^j \models p\right)\right)$

$\mathcal{M}, x \models Xp$ iff $\mathcal{M}, x^1 \models p$

Definition 5.1. A state formula p (resp. path formula p) is *valid* provided that for every structure $\mathcal{M}$ and every state S (resp. fullpath x) in $\mathcal{M}$ we have $\mathcal{M}, s \models p$ (resp. $\mathcal{M}, x \models p$). A state formula (resp. path formula) p is *satisfiable* provided that for some structure $\mathcal{M}$ and some states S (resp. fullpath x) in $\mathcal{M}$ we have $\mathcal{M}, S \models p$ (resp. $\mathcal{M}, x \models p$).

In [17] we develop the algorithms for computing the accessibility relation and in [20] we implement them in NuSMV [2] in order to perform model checking.

5.3. The logical analysis of a biological system

Consider the example of the interaction between the Virus and Macrophage discussed before. If the mathematical model chosen to describe the interaction is appropriate, then our system should have the property that, independently of the path of time that it will choose, always we will meet, in the future, the situation $Mac[Infect|Digest]$. Our logic allows us to formulate all these as a logical statement. We have:

$$u[v[open\ k.k'[Infect]]|Mac[k[out\ Mac.in\ v.in\ Mac.0]|open\ v.open\ k'.Digest]] \qquad (5.1)$$

For 5.1 we choose the urelements: α for u, β for Mac, o for 0, κ for k, ν for v, κ' for k', p for $Infect$ and q for $Digest$ with $\alpha, \beta, \nu, \kappa, \kappa', p, q, o \in \mathfrak{V}$. So, $U_A = \{\alpha, \beta, \nu, \kappa, \kappa'\}$, $U_P = \{q, p, o\}$, $O = \emptyset$; f is defined by: $f(\alpha) = u$, $f(\beta) = Mac$, $f(o) = 0$, $f(\kappa) = k$, $f(\nu) = v$, $f(\kappa') = k'$, $f(q) = Infect$, $f(p) = Digest$ and e is defined by:

$$\begin{array}{lcl}
e(\alpha) = \{e(\nu), e(\beta)\} & \Longrightarrow & \left\{ \begin{array}{l} e(\nu) \in e(\alpha) \\ e(\beta) \in e(\alpha) \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} \nu \mathrm{in} \alpha \quad \text{is true} \\ \beta \mathrm{in} \alpha \quad \text{is true} \end{array} \right. \\
e(\nu) = \{e(\kappa')\} & \Longrightarrow & \{\ e(\kappa') \in e(\nu) \Longrightarrow \{\ \kappa' \mathrm{in} \nu \quad \text{is true} \\
e(\beta) = \{e(\kappa), p\} & \Longrightarrow & \left\{ \begin{array}{l} e(\kappa) \in e(\beta) \\ p \in e(\beta) \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} \kappa \mathrm{in} \beta \quad \text{is true} \\ p \mathrm{in} \beta \quad \text{is true} \end{array} \right. \\
e(\kappa') = \{q\} & \Longrightarrow & \{\ q \in e(\kappa') \Longrightarrow \{\ q \mathrm{in} \kappa' \quad \text{is true} \\
e(\kappa) = \{o\} & \Longrightarrow & \{\ o \in e(\kappa) \Longrightarrow \{\ o \mathrm{in} \kappa \quad \text{is true}
\end{array}$$

The property we are interested in could be expressed as

$$Macrophage|Virus \models AF(\beta \mathrm{in} \alpha \bigwedge q \mathrm{in} \beta \bigwedge p \mathrm{in} \beta).$$

It says that for all time paths exists at least a reachable state for which Mac is a child of the master ambient $u = f(\alpha)$, $Infect = f(q)$ and $Digest = f(p)$ are children of the Macrophage ambient $Mac = f(\beta)$. Further, for checking the truth value of this statement, a model checker could be used. Proving that our logical formula is true it finally means that our mathematical model for describing our problem is a correct one. Vice versa, if is not valid, the model checker will give us a counter example that will show the conflict in our model. For such technics applied, the reader is referred to [20].

6. Conclusions

The logic we constructed on top of Ambient Calculus opens the perspective of using model checking algorithms (or software) developed for temporal logics in analyzing mobile computations. In this way we could predict the future of the systems (biological systems) described using the calculus.

Such verification technique used for biological systems can offer biologists a strong instrument for analysis. Expensive biological experiments could be replaced with model checking analysis and can be used very profitably especially in a negative way, for rejecting impossible hypothesis and for orienting the biologists' studies toward logical possible situations. In this way, instead of performing many experiments, the biologists will focus only on those that are as pointed as possible by the mathematical model.

Our ongoing research makes us confident in the possibility to construct such a logic for other calculi used for describing biological systems, e.g., BioAmbients Calculus [28], or Brane Calculi [9] but also for stochastic calculi. Moreover, using the set theoretical model we hope to identify the "common mathematical reality", which both the membrane computing and the process algebraical approach refer to as "the mathematical structure of the biological systems".

References

[1] HyTech: The HYbrid TECHnology Tool, `http://www-cad.eecs.berkeley.edu/ tah/HyTech/`.

[2] NuSMV: A New Symbolic Model Checker, `http://nusmv.irst.itc.it/`.

[3] Receptors Directly Activating Trimetric G Proteins, `http://courses.washington.edu/conj/gprotein/trimericgp.htm`.

[4] The SMV System, `http://www-2.cs.cmu.edu/ modelcheck/smv.html`.

[5] VIS Homepage, `http://www-cad.eecs.berkeley.edu/ vis/`.

[6] P. Aczel *Non-Well-Founded Sets*, CLSI Lecture Notes Number 14 Stanford: CSLI Publication, 1988.

[7] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, *Molecular Biology of the Cell*, Garland Publishing, Inc., 2002.

[8] J. Barwise, L. Moss, *Vicious Circles. On the Mathematics of Non-Wellfounded Phenomena*, CLSI Lecture Notes Number 60 Stanford: CSLI Publication, 1996.

[9] L. Cardelli, Brane Calculi, *Electronic Notes in Theoretical Computer Science* (to appear); available at *http://www.luca.demon.co.uk/*.

[10] L. Cardelli, L. Caires, A Spatial Logic for Concurrency (Part I), *Information and Computation*, 186, 2 (2003), 194–235.

[11] L. Cardelli, A.D. Gordon, Mobile Ambients, *Theoretical Computer Science*, 2000, 177–213.

[12] L. Cardelli, A.D. Gordon, Anytime, Anywhere. Modal Logics for Mobile Ambients, *Proceedings of the 27th ACM Symposium on Principles of Programming Languages*, 2000, 365–377.

[13] L. Cardelli, A.D. Gordon, Ambient Logic, *Mathematical Structures in Computer Science*, to appear; available at `http://www.luca.demon.co.uk/`.

[14] E. A. Emerson, Temporal and Modal Logic, *Handbook of Theoretical Computer Science*, Vol. B: *Formal Models and Sematics*, Elsevier, 1990, 995–1072.

[15] D.M. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Many-Dimensional Modal Logics: Theory and Applications, *Studies in Logic and the Foundations of Mathematics*, 148 (2003).

[16] F. Honsell, M. Forti, *Set Theory with Free Construction Principles*, Annali Scuola Normale Superiore di Pisa, 1983.

[17] R. Mardare, C. Priami, *Computing the Accessibility Relation for Ambient Calculus*, Dipartimento di Informatica e Tlc, University of Trento, 2003; available at *http://www.dit.unitn.it* following the link Publications.

[18] R. Mardare, C. Priami, *A Propositional Branching Temporal Logic for the Ambient Calculus*, Dipartimento di Informatica e Tlc, University of Trento, 2003; available at *http://www.dit.unitn.it* following the link Publications.

[19] R. Mardare, C. Priami, A Logical Spproach to security in the Context of Ambient Calculus, *Electronic Notes in Computer Science* (to appear).

[20] R. Mardare, C. Priami, Model Checking Biological Systems Described Using Ambient Calculus, *Electronic Notes in Computer Science*, to appear.

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

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

[23] R. Milner, O.H. Jensen, Bigraphs and Mobile Processes, *Technical Report* No 580, University of Cambridge, Computer Laboratory, 2004.

[24] J. Parrow, An Introduction to π-Calculus, *Handbook of Process Algebra*, Elsevier, 2000.

[25] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[26] C. Priami, Stochastic Π-Calculus, *Comput. J.*, 6 (1995), 578–589.

[27] C. Priami, A. Regev, E. Shapiro, W. Silverman, Application of Stochastic Name-Passing Calculus to Representation and Simulation of Molecular Processes, *Information Processing Letters*, 80 (2001), 25–31.

[28] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro, BioAmbients: An Abstraction for Biological Compartments, *Theoretical Computer Science* (to appear); available at `http://www.luca.demon.co.uk/`.