Coordinating Parallel Mobile Ambients to Solve SAT Problem in Polynomial Number of Steps

Bogdan Aman, Gabriel Ciobanu

Coordinating Parallel Mobile Ambients to Solve SAT Problem in Polynomial Number of Steps

Bogdan Aman and Gabriel Ciobanu

Romanian Academy, Institute of Computer Science, Iaşi, Romania
"A.I.Cuza" University, Blvd. Carol I no.11, 700506 Iaşi, Romania
gabriel@info.uaic.ro

Abstract. In this paper we present a version of mobile ambients, called parMA, having a weak form of replication and a parallel semantics. We investigate how parMA can solve intractable problems in a polynomial number of computational steps. We use parMA to give a semiuniform solution to a well-known strong NP-complete problem, namely to the Boolean satisfiability problem (SAT).

1 Introduction

Ambient calculus is a formalism for describing distributed and mobile computing [11]. In contrast with the π-calculus [19] where mobility is expressed by communication, the ambient calculus uses an explicit notion of *movement* given by moving actions (*in* and *out*) together with an "opening" action and (local) communication. An ambient is a named location, and it represents the unit of movement. The authors of [11] introduce the mobile ambients as "a paradigm of mobility where computational ambients are hierarchically structured, where agents are confined to ambients and where ambients move under the control of agents". Their initial goal was "to make mobile computation scale-up to widely distributed, intermittently connected and well administered computational environments". The resulting ambient model is elegant and powerful, well suited for expressing issues of mobile computations as working environment, allowing access to information or resources [2, 16]. Many variants have been proposed; among them, we mention mobile safe ambients [17], push and pull ambient calculus [16] and boxed ambients [7]. Despite the fact that the initial motivation of mobile ambients assumes a high degree of parallelism in their evolution, the usual semantics of the proposed variants is the interleaving semantics.

Some results show that mobile ambients have the computational power of Turing machines by encoding into ambient calculus (or fragments of it) some formalisms known to be Turing complete: asynchronous π-calculus [11, 12], counter machines [9] and Turing machines [11, 18]. A link between mobile ambients and π-calculus is established in [13], where it is proven that pure mobile ambients can be embedded into a fragment of the π-calculus, namely in the localized sum-free synchronous monadic π-calculus with matching and mismatching. Other authors relate ambients to security issues and to system biology. Some simulators were

also developed [14] in which the evolution of mobile ambients can be observed easily (see http://www-sop.inria.fr/mimosa/ambicobjs/).

Many formal machine models (e.g., Turing machines) have an infinite number of memory locations. Mobile ambients are computing devices of finite size having a finite description with a fixed amount of initial resources (ambients and processes), that can evolve to a possibly infinite family of mobile ambients obtained by replication in order to solve a (decision) problem. A decision problem X is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called instances) and θ_X is a total boolean function (that is, a predicate) over I_X. Its solvability is defined through the recognition of the language associated with it. Let M be a Turing machine with the working alphabet Γ, L a language over Γ, and the result of any halting computation is *yes* or *no*. If M is a deterministic device, it recognizes or decides L whenever, for any string u over Γ, if $u \in L$, then M accepts u (the result on input u is *yes*), or M rejects u (the result on input u is *no*). If M is a non-deterministic device, it recognizes or decides L whenever, if for any string u over Γ, $u \in L$, and only if there exists a computation of M with input u such that the answer is *yes*.

According to [15], the NP-complete problems are divided into weak (e.g., Knapsack) and strong (e.g., SAT) depending on the size of the input. We show how a parallel version of mobile ambients with a weak form of replication can solve NP-complete problems in a polynomial number of steps. We provide a semiuniform solution to the best known strong NP-complete problem (SAT) in a polynomial number of steps [22]. To give such a solution, we treat mobile ambients as "deciding devices" that respect the following conditions: (1) all computations halt, (2) two special names *yes* and *no* are used, and (3) in a halting configuration a channel *ans* is ready to output one of the names *yes* and *no*; the computation is *accepting* if *yes* is present in the halting configuration, and *rejecting* if *no* is present in the halting configuration on channel *ans*. Mobile ambients respect these conditions if we impose some constraints:

- We use a true concurrent semantics allowing processes to run in parallel. The key rule is
$$P \to P' \text{ and } Q \to Q' \text{ implies } P \mid Q \to P' \mid Q'.$$
This parallel semantics is natural if we recall that in [11] a process is described as "running even when the surrounding ambient is moving", aspect which the interleaving semantics does not reflect properly. Other reasons to consider a parallel semantics are presented in [14] where the authors present a distributed implementation of mobile ambients.
- A restricted form of replication is used by considering a weaker *duplication* operator which only doubles a process; this means that a reduction rule $!P \to P \mid P$ is used instead of the congruence rule $!P \equiv P \mid\ !P$ or instead of the reduction rule $!P \to P \mid\ !P$. This duplication rule is also used by R. Milner in [20]. It helps in controlling all the computations to halt (and so fulfilling condition 1. of deciding devices).
- We use a special symbol Ⓢ that helps in delimiting the computational steps. Ⓢ is a purely technical device that is used in the subsequent for-

malization of the structural operational semantics of parMA; intuitively, ⓈP specifies a process P which is temporarily stalled and so cannot execute any action.

- We use two kinds of action rules: $\rightarrow$ and $\stackrel{\phi}{\Rightarrow}$. The former is an execution of a set of actions, and the latter is used to remove all occurrences of Ⓢ (using a tree parsing algorithm) in order to start a new round of parallel actions.

The differences between parallel and interleaving semantics is underlined also in [5]: "The parallel construct is interpreted in terms of interleaving, as usual in many (timed) process algebras .... Alternatively one could adopt maximal parallelism, which means that at each moment every enabled process of the system is activated". In defining a parallel semantics, we follow the solution used also in [8] where such a semantic is defined in brane calculi [10], a process algebra related to BioAmbients [23].

The paper is organised as follow. Section 2 defines the syntax and semantics of parallel mobile ambients (parMA). In Section 3 we give some notions of complexity, and show how to solve SAT problem in a polynomial number of steps. Section 4 illustrates how mobile ambients compute effectively by considering a SAT instance with three clauses and three variables as an example. Conclusion and references end the paper.

2 Parallel Mobile Ambients

In this section we present a variant of mobile ambients having a parallel semantics. Initially, mobility in ambient calculus involved the authorization to enter or exit certain domains in order to access information; the access to information is controlled at many levels: local computer, local area network, regional area network, wide-area intranet and internet. We consider a framework given by mobile ambients hierarchically structured inside a well-defined environment, where ambients move under the control of agents running inside them.

2.1 Syntax

Table 1 describes the syntax of parMA.

Table 1: *Parallel Mobile Ambients Syntax*

c	channel name	P, Q	::=	processes	
a, b, Env	ambient names		$\mathbf{0}$	inactivity	
x, y	variables		$M.P$	movement	
M	::= capabilities		$a[P]$	ambient	
	in a can enter a		$P \mid Q$	composition	
	out a can exit a		$c\langle a\rangle.P$	output action	
	open a can open a		$c(x).P$	input action	
			$!P$	duplication	
			ⓈP	stalled	

The name Env represents the environment in which the mobile ambients reside, and can appear only once, at the top of the hierarchical structure. Process $\mathbf{0}$ is an inactive process (it does nothing). A movement $M.P$ is provided by the capability M, followed by the execution of process P. An ambient $a[P]$ represents a bounded place labelled by a in which a process P is executed. $P \mid Q$ is a parallel composition of processes P and Q. An output action $c\langle a\rangle.P$ releases a name a on channel c, and then behaves as process P. An input action $c(x).\,P$ captures a name from channel c, and binds it to a variable x within the scope of process P. A weak form of replication, namely the duplication of a process P (producing two parallel copies of process P) is denoted by $!P$. The process $\textcircled{S}P$ is used to state that process P is temporarily "stalled".

2.2 Operational Semantics

The first component of the operational semantics of parMA is the *structural congruence* $\equiv$. It is the smallest congruence such that the equalities from Table 2 hold. Its role is to rearrange a process in order to apply the action rules given in Table 3. The axioms from Table 2 describe the commutativity and associativity of the parallel composition.

Table 2: *Structural Congruence*

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

The set $fn(P)$ of free names of a process P is defined as:

$$fn(P)=\begin{cases} \emptyset & \text{if } P = \mathbf{0} \\ fn(R) \cup \{a\} & \text{if } P = in\ a.R \text{ or } P = out\ a.R \\ & \text{or } P = open\ a.R \text{ or } P = a[R] \\ fn(R) \cup \{a, c\} & \text{if } P = c\langle a\rangle.R \\ fn(R) \cup \{c\} & \text{if } P = c(x).R \\ fn(R) \cup fn(Q) & \text{if } P = R \mid Q \\ fn(R) & \text{if } P = !R \end{cases}$$

Table 3 introduces two kinds of action rules: $P \rightarrow P'$ and $P \stackrel{\phi}{\Rightarrow} P'$. The former is an execution of a set of actions, and the latter is used to remove all occurrences of $\textcircled{S}$ in order to start a new round of parallel actions. The first five rules of Table 3 are one-step reductions for *in, out, open, communication* and *duplication*. In rule **(Com)**, by $P'\{a/x\}$ we denote the substitution by a of each free occurrence of variable x in process P'. The next three rules propagate reductions across ambient nesting and parallel composition. In rule (**Par2**), by $R \not\rightarrow$ is denoted a process R that cannot evolve. Rule (**Struct**) allows the use of structural congruence during reduction. When no rule can be applied in Env, rule (**Step**) is used to delete all occurrences of $\textcircled{S}$ in order to start a new round of transitions. It can be noticed that in rules **(Par2)** and **(Step)** we use negative

premises: an activity is performed based on the absence of actions. This is due to the fact that sequencing the evolution can only be defined using negative premises, as done for sequencing processes [4, **?**,21].

Table 3: *Reduction Rules*

(In)	$\dfrac{a \neq Env \quad b \neq Env}{a[in\ b.\, P \mid Q] \mid b[R] \rightarrow b[a[\textcircled{S}P \mid Q] \mid R]}$
(Out)	$\dfrac{a \neq Env \quad b \neq Env}{a[b[out\ a.\, P \mid Q] \mid R] \rightarrow b[\textcircled{S}P \mid Q] \mid a[R]}$
(Open)	$\dfrac{a \neq Env}{open\ a.\, P \mid a[Q] \rightarrow \textcircled{S}P \mid Q}$
(Com)	$c\langle a\rangle.\, P \mid c(x).\, P' \rightarrow \textcircled{S}P \mid \textcircled{S}P'\{a/x\}$
(Dupl)	$!P \rightarrow \textcircled{S}P \mid \textcircled{S}P$
(Amb)	$\dfrac{P \rightarrow Q}{a[P] \rightarrow a[Q]}$
(Par1)	$\dfrac{P \rightarrow P' \quad Q \rightarrow Q'}{P \mid Q \rightarrow P' \mid Q'}$
(Par2)	$\dfrac{P \rightarrow Q \quad R \not\rightarrow}{P \mid R \rightarrow Q \mid R}$
(Struct)	$\dfrac{P' \equiv P,\ P \rightarrow Q, \quad Q \equiv Q'}{P' \rightarrow Q'}$
(Step)	$\dfrac{Env[P] \not\rightarrow}{P \stackrel{\phi}{\Rightarrow} \phi(P)}$

The rules of Table 3 define execution of processes. A complete computational step in mobile ambients is captured by a derivation of the form

$$Env[P] \rightarrow\!\!\stackrel{\phi}{\Rightarrow} Env[P'].$$

This means that a derivation is a compressed representation of a sequence of individual actions followed by a reinitialization step (removing of all Ⓢ symbols).

2.3 Example

To illustrate the basic components of parMA, we use an example in which several students wish to move from the campus to the university and back, having the possibility to use either a tram or a bus. The scenario involves eight ambients and four processes.

$$Env[campus[student[P_1 \mid P_4] \mid student[P_1 \mid P_4] \mid tram[P_2 \mid P_4] \mid bus[P_3 \mid P_4]] \mid univ[student[P_1 \mid P_4]]]$$

The role of the ambients is suggested by their names. The processes are:

- $P_1 = Q_1 \mid Q_2$
  $Q_1 = in\ tram.c(x).c(x).c(x).out\ tram.Q_1$
  $Q_2 = in\ bus.c(x).c(x).c(x).c(x).out\ bus.Q_2$

- $P_2 = c(x).out\ campus.c(x).in\ univ.P_2'$
  $P_2' = c(x).out\ univ.c(x).in\ campus.P_2$
- $P_3 = c(x).out\ campus.c(x).c(x).in\ univ.P_3'$
  $P_3' = c(x).out\ univ.c(x).c(x).in\ campus.P_3$
- $P_4 = c\langle a\rangle.P_4$

The communication on channel c, that takes place inside ambients, is used to model the fact that the *tram*, *bus* and *students* perform the following actions:

- the first $c(x)$ from P_2, P_2', P_3 and P_3' represents the fact that the *bus* and *tram*, once inside the *campus* or *univ*, are willing to wait for *students* that intend to travel between *campus* and *univ*.
- the others $c(x)$ from P_2, P_2', P_3 and P_3' are used to model the fact that the movement of the *tram* and *bus* between *campus* and *univ* takes a number of steps (equal with the number of input actions on channel c).
- all $c(x)$ from Q_1 and Q_2 are used to prevent the *students* from getting out of the *tram* or *bus* before reaching the desired location.

It can be noticed that both *students* from *campus* can enter either the *bus* or the *tram*. Suppose both choose the *tram*. Then the mobile ambient

$$Env[campus[student[P_1 \mid P_4] \mid student[P_1 \mid P_4] \mid tram[P_2 \mid P_4] \mid bus[P_3 \mid P_4]] \mid univ[student[P_1 \mid P_4]]]$$

evolves to

$$Env[campus[tram[Ⓢout\ campus.c(x).in\ univ.P_2' \mid ⓈP_4 \mid student[Ⓢc(x).c(x).c(x).out\ tram.Q_1 \mid Q_2 \mid ⓈP_4] \mid student[Ⓢc(x).c(x).c(x).out\ tram.Q_1 \mid Q_2 \mid ⓈP_4]] \mid bus[Ⓢout\ campus.c(x).c(x).in\ univ.P_3' \mid ⓈP_4]] \mid univ[student[P_1]]]$$

The *tram* and the *bus* are still inside *campus* since they communicated on channel c in order to permit the willing *students* to get inside them. At this moment it can be noticed that only rule **(Step)** can be applied in order to eliminate the symbols Ⓢ, obtaining the mobile membrane

$$Env[campus[tram[out\ campus.c(x).in\ univ.P_2' \mid P_4 \mid student[c(x).c(x).c(x).out\ tram.Q_1 \mid Q_2 \mid P_4] \mid student[c(x).c(x).c(x).out\ tram.Q_1 \mid Q_2 \mid P_4]] \mid bus[out\ campus.c(x).c(x).in\ univ.P_3' \mid P_4]] \mid univ[student[P_1]]]$$

After three steps, the *tram* is inside *univ* where it waits for the students to exit/enter it, while the *bus* is still between the two *campus* and *univ*, being ready to enter *univ* in the next step. Thus the next mobile ambient is obtained

$$Env[campus[\ ] \mid bus[in\ univ.P_3' \mid P_4] \mid univ[tram[P_2' \mid P_4 \mid student[out\ tram.Q_1 \mid Q_2 \mid P_4] \mid student[out\ tram.Q_1 \mid Q_2 \mid P_4]] \mid student[P_1]]]$$

In the next step the two *students* from the *tram* get out inside *univ*, while the *student* that was waiting enters the *tram* in order to reach *campus*. The obtained mobile ambients is

$$Env[campus[\ ] \mid univ[bus[P_3' \mid P_4] \mid tram[out\ univ.c(x).in\ campus.P_2 \mid P_4 \mid student[c(x).c(x).c(x).out\ tram.Q_1 \mid Q_2 \mid P_4]]$$

$$| \; student[P_1] \; | \; student[P_1]]]$$

The mobile ambient continues to change, but we stop here since we have illustrated the expressive power of the proposed formalism.

3 Solving NP-Complete Problems in Polynomial Steps

As stated in the introduction, we use mobile ambients as deciding devices, in which all computation starting from the initial ambient agree on the result. A family **MA**, a collection of ambients, solves a decision problem if for each instance of the problem there is a member of the family able to decide on the instance. In order to define the notions of (semi)uniformity, we denote:

- for a suitable alphabet O, each instance of the decision problem is encoded as a string v over O;
- $V = \{v_1, \ldots\}$ - the language of encoded instances of the given problem;
- $\mathbf{MA}(v)$ - the member of **MA** which solves the instance v;
- $\mathbf{MA}_n$ - the member of **MA** which solves all instances of length n.

Definition 1. *The family* **MA**

(i) decides V *if for any string* $v \in O^*$*, the mobile ambient* $\mathbf{MA}(v)$ *(or* $\mathbf{MA}_n$ *for all instances* v*,* $|v| = n$*) generates an yes answer whenever* $v \in V$ *and a no answer otherwise;*

(ii) is sound *with respect to* V *when, for any string* $v \in O^*$*, if there exists an accepting computation of* $\mathbf{MA}(v)$ *(*$\mathbf{MA}_n$*), then* $v \in V$*;*

(iii) is complete *with respect to* V *when, for any string* $v \in O^*$*, if* $v \in V$*, then every computation of* $\mathbf{MA}(v)$ *(*$\mathbf{MA}_n$*) is accepting.*

Inspired by the uniformity conditions applied to families of Boolean circuits [6], we imposed similar ones on families of processes. By imposing certain resource restrictions (on number of steps and space) to the function that constructs each member of the family **MA**, it can be ensured that the set of problems decided by the family does not increase. The function is called an

- *uniformity condition* if an instance size is mapped to a mobile ambient that decides all instances of that length;
- *semiuniformity condition* if a single instance is mapped to a mobile ambient that decides that instance.

Definition 2. *If we consider a set of problem instances* $V = \{v_1, v_2, \ldots\}$*, two classes of functions* E, F *and a total function* $t : \mathbb{N} \to \mathbb{N}$*, such that:*

1. *there exist a* F*-uniform family of mobile ambients* $\mathbf{MA} = \{\mathbf{MA}_1, \ldots\}$*; this means that there exist a function* $f \in F$*,* $f : \{1\}^* \to \mathbf{MA}$ *such that* $f(1^n) = \mathbf{MA}_n$*, namely all instances* v_k *of length* n *are solved by* $\mathbf{MA}_n$*, where* $\mathbf{MA}_n$ *can be constructed by a function* $f \in F$*;*

2. *there exists an encoding function* $e \in E$ *such that* $e(v)$ *is the input process of* $\mathbf{MA}_n$*, for* $|v| = n$*;*
3. **MA** *is* t*-efficient:* $\mathbf{MA}_n$ *halts in* $t(n)$ *steps (e.g.,* **MA** *is polynomial efficient if* $t(n)$ *is polynomial in* n *for all* n*);*
4. **MA** *is sound and complete with respect to* V*,*

then we say that the class of problems V *is solved by an* (E, F)*-uniform family of mobile ambients* **MA***, and denote the family by (E,F)-***MA***(t). The set of languages decided by a uniform family of mobile ambients in a polynomial number of steps is defined as (E,F)-***PMA*** $= \bigcup_{k\in\mathbb{N}}$ (E,F)-***MA***(n^k).*

Semiuniformity is a generalization of uniformity, namely

Definition 3. *If we consider a set of problem instances* $V = \{v_1, v_2, \ldots\}$*, a class of functions* H *and a total function* $t : \mathbb{N} \to \mathbb{N}$*, such that:*

1. *there exist a* H*-semiuniform family of mobile ambients* $\mathbf{MA} = \{\mathbf{MA}_{v_1}, \mathbf{MA}_{v_2}, \ldots\}$*; namely, there exist a function* $h \in H$*,* $h : V \to \mathbf{MA}$ *such that* $h(v_i) = \mathbf{MA}_{v_i}$*;*
2. **MA** *is* t*-efficient:* $\mathbf{MA}_n$ *halts in* $t(|v_n|)$ *steps;*
3. **MA** *is sound and complete with respect to* V*,*

then we say that the class of problems V *is solved by an* (H)*-semiuniform family of mobile ambients* **MA***, and denote the family by (H)-***MA***(t). The set of languages decided by a semiuniform family of mobile ambients in a polynomial number of steps is defined as (H)-***PMA*** $= \bigcup_{k\in\mathbb{N}}$ (H)-***MA***(n^k).*

3.1 Boolean Satisfiability Problem

The SAT problem checks the satisfiability of a propositional logic formula in conjunctive normal form (CNF). Let $\{x_1, x_2, \ldots, x_n\}$ be a set of Boolean variables. A formula in CNF is of the form

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

where each $C_i, 1 \leq i \leq m$ is a disjunction of the form

$$C_i = y_1 \vee y_2 \vee \cdots \vee y_r \ (r \leq n),$$

where each y_j is either a variable x_k or its negation $\neg x_k$. In this section we show how, starting from a formula φ, to construct a process P that provides a semiuniform solution to the SAT problem by using mobile ambients with parallel semantics and duplication (for an instance of SAT we construct a mobile ambient which decides it). We start with the process

$$P = P_1 \mid Q_1$$

in which P_1 is used to provide the answer to the problem when placed in parallel with Q_1, a process that generates all possible assignments over the set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables. In what follows we describe how each of these two processes are constructed starting from the φ formula.

– process Q_1 is defined recursively using the processes Q_i $(1 \leq i < n)$ and Q_n. For each variable x_i from the set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables, we construct a process Q_i defined as follows:

$$Q_i = x_i\langle t_i\rangle.x\langle z\rangle.x\langle z\rangle \mid x_i\langle f_i\rangle.x\langle z\rangle.x\langle z\rangle \mid$$
$$|!x_i(y_i).(x(y).x(y).open\ k_i \mid k_i[Q_{i+1}]), \text{ for } 1 \leq i < n$$
$$Q_n = x_n\langle t_n\rangle \mid x_n\langle f_n\rangle\ |!x_n(y_n).A[x_1\langle y_1\rangle \mid \ldots \mid x_n\langle y_n\rangle$$
$$\mid y_1\langle a\rangle \mid \ldots \mid y_1\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q].$$

where:

- process Q contains terms of the form $t_k(b).in\ C_j$ (if x_k appears in C_j) or $f_k(b).in\ C_j$ (if $\neg x_k$ appears in C_j). For example if we consider a 3CNF satisfiability problem with $\varphi = C_1 \wedge C_2 \wedge C_3$ and $X = \{x_1, x_2, x_3\}$, $C_1 = x_1 \vee \neg x_3$, $C_2 = \neg x_1 \vee \neg x_2$ and $C_3 = x_2$ we have
  $$Q = t_1(b).in\ C_1 \mid f_3(b).in\ C_1 \mid f_1(b).in\ C_2 \mid f_2(b).in\ C_2 \mid t_2(b).in\ C_3$$
- each ambient A will contain a different assignment over the set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables; after Q_n is executed, there will be 2^n ambients A, obtained by using the duplication operator ! that proceeds the processes that generate an A ambient.
- $y_j\langle a\rangle \mid \ldots \mid y_j\langle a\rangle$ stands for m parallel processes $y_j\langle a\rangle$, one for each disjunction, and, after all y_j are instantiated, are used to communicate with the processes from Q.
- $x\langle z\rangle.x\langle z\rangle$ are used to introduce a delay that prevents that an ambient k_i containing a Q_{i+1} is not opened to soon and cause unwanted evolutions.

– process P_1 has the form:

$$P_1 = C_1[\ldots[C_m[J[x(y)\ldots x(y) \mid x\langle z\rangle\ldots x\langle z\rangle.K[out\ J]]$$
$$\mid L[in\ A.ans\langle yes\rangle \mid in\ K.ans\langle no\rangle]]]]$$

where:

- to each disjunction C_i, $1 \leq i \leq m$ we associate an ambient C_i;
- the ambients C_i, $1 \leq i \leq m$ are placed one inside the other, forming an ambient structure of depth m. The order in which these ambients are placed is not important (thus the construction is not unique), but for simplicity we consider the ambient
  $$C_1[\ldots[C_m[\ldots]]\ldots]$$
- the previous ambient is used to check if there exists an assignments over the set $\{x_1, x_2, \ldots, x_n\}$ of Boolean variables that respects all these disjunctions. If such an assignment exists, this means that an ambient A containing this assignment, will eventually reach inside ambient C_m (an ambient A enters an ambient C_i if the assignment placed inside A respect the disjunction C_i)
- $x(y)\ldots x(y)$ stands for a $2n + m + 1$ sequence of capabilities $x(y)$, and together with $x\langle z\rangle\ldots x\langle z\rangle$ that stands for a $2n+m+1$ sequence of capabilities $x\langle z\rangle$, introduces a delay equal with the number of steps needed by an ambient A to get inside ambient C_m, before the ambient K exits ambient J. It can be noticed that if an ambient A gets near the ambient L (inside ambient C_m), this ambient enters the ambient A, generates the *yes* answer and prevents K to generate the negative answer (the ambient K cannot interact with the ambient L inside the ambient A).

In what follows we explain how these two processes (P_1 and Q_1) once constructed from the φ formula, can generate an answer to the problem.

Starting from P_1 the evolution in the first $2n+1$ steps is given by the rule $P_i \rightarrow\!\!\stackrel{\phi}{\Rightarrow} P_{i+1}$, where P_{i+1} is obtained from P_i by performing a communication on channel x. In parallel, starting from the process Q_1, are generated 2^n ambients A that contain all possible assignments over the variables $\{x_1, \ldots, x_n\}$, namely each assignment is contained inside an ambient A.

Next we describe in detail the evolution of Q_1. We have two cases.

Case 1: For $1 \leq i \leq n-1$, the evolution of each

$$Q_i = x_i\langle t_i\rangle.x\langle z\rangle.x\langle z\rangle \mid x_i\langle f_i\rangle.x\langle z\rangle.x\langle z\rangle \\ \mid\, !x_i(y_i).(x(y).x(y).open\ k_i \mid k_i[Q_{i+1}])$$

from the 2^{i-1} processes Q_i running in parallel, starts with a duplication, because any other reduction is not possible. The process that duplicates is in fact the process containing the ambient labelled by k_i in which the process Q_{i+1} is placed. This is done because the variables $y_1, \ldots, y_{i-1}$, $i \geq 2$ are already instantiated in the process Q_{i+1}, and we want to create two new copies: one in which y_i is replaced by t_i, and one in which y_i is replaced by f_i, keeping also the already instantiated variables. We obtain the process:

$$Q_i^1 = \mathbf{x_i}\langle \mathbf{t_i}\rangle.x\langle z\rangle.x\langle z\rangle \mid \mathbf{x_i}\langle \mathbf{f_i}\rangle.x\langle z\rangle.x\langle z\rangle \\ \mid \mathbf{x_i}(\mathbf{y_i}).(x(y).x(y).open\ k_i \mid k_i[Q_{i+1}]) \\ \mid \mathbf{x_i}(\mathbf{y_i}).(x(y).x(y).open\ k_i \mid k_i[Q_{i+1}])$$

At this moment Q_i^1 has two input actions on channel x_i, and two output actions on channel x_i that are ready to communicate the values of t_i and f_i. This means that two **(Comm)** rules are applied in parallel, leading to:

$$Q_i^2 = \mathbf{x}\langle \mathbf{z}\rangle.x\langle z\rangle \mid \mathbf{x}\langle \mathbf{z}\rangle.x\langle z\rangle \\ \mid \mathbf{x}(\mathbf{y}).x(y).open\ k_i \mid k_i[Q_{i+1}\{t_i/y_i\}] \\ \mid \mathbf{x}(\mathbf{y}).x(y).open\ k_i \mid k_i[Q_{i+1}\{f_i/y_i\}]$$

After the communications on channels x_i are performed, the communication of t_{i+1} and f_{i+1} inside Q_{i+1} on channels x_{i+1} takes place in two steps. This motivates a delay in opening the ambients k_i, such that the communication from all k_i running in parallel does not get mixed up on channels x_{i+1}, leading to some unwanted assignments. The obtained process is:

$$Q_i^3 = \mathbf{x}\langle \mathbf{z}\rangle \mid \mathbf{x}\langle \mathbf{z}\rangle \mid \mathbf{x}(\mathbf{y}).open\ k_i \mid k_i[Q_{i+1}^1\{t_i/y_i\}] \\ \mid \mathbf{x}(\mathbf{y}).open\ k_i \mid k_i[Q_{i+1}^1\{f_i/y_i\}]$$

The channels x_{i+1} are ready to communicate inside the processes Q_{i+1}, and so the capabilities $open\ k_i$ are released in the next step.

$$Q_i^4 = \mathbf{open\ k_i} \mid \mathbf{k_i}[Q_{i+1}^2\{t_i/y_i\}] \mid \mathbf{open\ k_i} \mid \mathbf{k_i}[Q_{i+1}^2\{f_i/y_i\}]$$

Once the communication inside ambients k_i on channels x_{i+1} has finished, these ambients are opened, thus obtaining

$$Q_i^5 = Q_{i+1}^3\{t_i/y_i\} \mid Q_{i+1}^3\{f_i/y_i\}$$

Since the process Q_i^5 does not contain any capabilities or replication operators, except the ones from Q_{i+1}^3, it means that each Q_i evolves for 5 steps, from which 3 steps are in parallel with the ones from Q_{i+1}.

Case 2: for $i = n$ the evolution of each

$$Q_n = x_n\langle t_n\rangle \mid x_n\langle f_n\rangle$$
$$\mid !x_n(y_n).A[x_1\langle y_1\rangle \mid \ldots x_n\langle y_n\rangle \mid y_1\langle a\rangle \mid \ldots y_1\langle a\rangle \mid \ldots y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q]$$

from the 2^{n-1} processes Q_n running in parallel, starts with a duplication rule, obtaining

$$Q_n^1 = \mathbf{x_n}\langle \mathbf{t_n}\rangle \mid \mathbf{x_n}\langle \mathbf{f_n}\rangle$$
$$\mid \mathbf{x_n}(\mathbf{y_n}).A[x_1\langle y_1\rangle \mid \ldots x_n\langle y_n\rangle \mid y_1\langle a\rangle \mid \ldots y_1\langle a\rangle \mid \ldots y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q]$$
$$\mid \mathbf{x_n}(\mathbf{y_n}).A[x_1\langle y_1\rangle \mid \ldots x_n\langle y_n\rangle \mid y_1\langle a\rangle \mid \ldots y_1\langle a\rangle \mid \ldots y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q]$$

and after communications of t_n or f_n on channels x_n:

$$Q_n^2 = A[x_1\langle y_1\rangle \mid \ldots x_n\langle y_n\rangle \mid y_1\langle a\rangle \mid \ldots y_1\langle a\rangle \mid \ldots y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q]\{t_n/y_n\}$$
$$\mid A[x_1\langle y_1\rangle \mid \ldots x_n\langle y_n\rangle \mid y_1\langle a\rangle \mid \ldots y_1\langle a\rangle \mid \ldots y_n\langle a\rangle \mid \ldots \mid y_n\langle a\rangle \mid Q]\{f_n/y_n\}$$

Since there are only ambients labelled by A and no *open* capabilities, the process Q_n^2 cannot evolve any more. However, in order to cope with the fact that $Q_i^5 = Q_{i+1}^3\{t_i/y_i\} \mid Q_{i+1}^3\{f_i/y_i\}$, for $1 \leq i \leq n-1$, we consider Q_n^2 to be equal with Q_n^3.

Starting from $P_1 \mid Q_1$, after $2n+1$ steps we obtain $P_{2n+2} \mid Q''$, where:

$$Q'' = A[x_1\langle t_1\rangle \mid \ldots \mid x_n\langle t_n\rangle \mid t_1\langle a\rangle \mid \ldots \mid t_1\langle a\rangle \mid \ldots \mid t_n\langle a\rangle \mid \ldots \mid t_n\langle a\rangle \mid Q]$$
$$\mid \ldots \mid A[x_1\langle f_1\rangle \mid \ldots \mid x_n\langle f_n\rangle \mid f_1\langle a\rangle \mid \ldots \mid f_1\langle a\rangle \mid \ldots \mid f_n\langle a\rangle \mid \ldots \mid f_n\langle a\rangle \mid Q]$$

To illustrate how such a process Q'' looks, we give a small example in which $n = m = 2$, we have $2^2 = 4$ ambients generated by Q_1, and process Q'' is:

$$Q'' = A[x_1\langle t_1\rangle \mid x_2\langle t_2\rangle \mid t_1\langle a\rangle \mid t_1\langle a\rangle \mid t_2\langle a\rangle \mid t_2\langle a\rangle \mid Q]$$
$$\mid A[x_1\langle t_1\rangle \mid x_2\langle f_2\rangle \mid t_1\langle a\rangle \mid t_1\langle a\rangle \mid f_2\langle a\rangle \mid f_2\langle a\rangle \mid Q]$$
$$\mid A[x_1\langle f_1\rangle \mid x_2\langle t_2\rangle \mid f_1\langle a\rangle \mid f_1\langle a\rangle \mid t_2\langle a\rangle \mid t_2\langle a\rangle \mid Q]$$
$$\mid A[x_1\langle f_1\rangle \mid x_2\langle f_2\rangle \mid f_1\langle a\rangle \mid f_1\langle a\rangle \mid f_2\langle a\rangle \mid f_2\langle a\rangle \mid Q]$$

As it can be noticed, the four ambients A contain all the Boolean assignments over the variables $\{x_1, x_2\}$, namely $\{t_1, t_2\}$, $\{t_1, f_2\}$, $\{f_1, t_2\}$, $\{f_1, f_2\}$, and each possible assignment t_i and f_i is kept as an output value on channel x_i.

After obtaining all possible assignments, we need to check which one satisfies all the clauses C_j. To do this, we use the processes Q that contain either terms of the form $t_k(b).in\ C_j$ meaning that x_k appears in C_j, or of the form $f_k(b).in\ C_j$ meaning that $\neg x_k$ appears in C_j. To be able to use the capability $in\ C_j$, there should be a $t_k\langle a\rangle$, respectively a $f_k\langle a\rangle$ inside the ambient A, both resulting from the instantiation of y_k. All ambients that satisfy the clause C_j enter in parallel the ambient C_j. If there exist at least one ambient A that contains $in\ C_1 \mid \ldots \mid in\ C_m$, it means that this ambient can go inside ambient C_m, and contains a solution to the SAT problem; in this case the ambient L enters the ambient A placed inside membrane C_m, releasing the *yes* answer on channel *ans* (1 step). Otherwise, ambient K exits ambient J, and so ambient L enters K; thus the *no* answer is send on channel *ans* (2 steps).

We have a deterministic evolution of the mobile ambients, and so no interference (redex overlapping) exists in our solution of SAT problem. This motivates the use of mobile ambients rather than safe mobile ambients [17].

3.2 Analysis

If n is the number of variables $(x_1, \ldots, x_n)$, and m is the number of clauses $(C_1, \ldots, C_m)$, then the number of ambients, capabilities and duplication operators in the initial process is given by the sum of:

- 3 ambients and $4n + 2m + 7$ capabilities in process P_1;
- 10 capabilities, 1 ambient and 1 replication operator in each Q_i, $1 \leq i < n$;
- $n + 3 + mn$ capabilities, 1 ambient and 1 replication operator in Q_n;
- maximum $4m$ capabilities in Q.

Thus, the total size of the initial process is $\mathcal{O}(mn)$. The maximum number of computational steps performed in an execution is equal with $2n+m+3$, a number determined by:

- $2n + 1$ steps to generate all the possible Boolean assignments over a set of variable $\{x_1, \ldots, x_n\}$;
- m steps required by a solution to move inside ambient C_m;
- either 1 step to generate a *yes* answer on channel *ans*, or 2 steps to generate a *no* answer on channel *ans*.

It is straightforward to show that:

- the construction of $P_1 \mid Q_1$ is semiuniform;
- sound and complete: $P_1 \mid Q_1$ says *yes* iff the given SAT instance is satisfiable;
- function H required for the above construction is in **P**.

Proposition 1. *Using* parMA*, NP-complete problems can be solved in a polynomial number of steps.*

4 An Example of How Mobile Ambients Solve 3CNF-SAT

To illustrate how mobility can "compute" and solve hard problems, we consider a 3CNF satisfiability problem with $\varphi = C_1 \wedge C_2 \wedge C_3$ and $X = \{x_1, x_2, x_3\}$, $C_1 = x_1 \vee \neg x_3$, $C_2 = \neg x_1 \vee \neg x_2$ and $C_3 = x_2$. In this case $n = 3$ and $m = 3$. We start with the mobile ambient:

$$P = P_1 \mid Q_1$$

where

$$P_1 = C_1[\ldots[C_3[J[x(y)\ldots x(y) \mid x\langle z\rangle \ldots x\langle z\rangle.K[out\ J]] \mid L[in\ A.ans\langle yes\rangle \mid in\ K.ans\langle no\rangle]]]]$$

with Q_i $(1 \leq i < n)$ and Q_n defined as follows

$$Q_i = x_i\langle t_i\rangle.x\langle z\rangle.x\langle z\rangle \mid x_i\langle f_i\rangle.x\langle z\rangle.x\langle z\rangle \mid$$
$$|!x_i(y_i).(x(y).x(y).open\ k_i \mid k_i[Q_{i+1}]), \text{ for } 1 \leq i < 3$$
$$Q_3 = x_3\langle t_3\rangle \mid x_3\langle f_3\rangle\ |!x_3(y_3).A[x_1\langle y_1\rangle \mid x_2\langle y_2\rangle \mid x_3\langle y_3\rangle$$
$$\mid y_1\langle a\rangle \mid y_1\langle a\rangle \mid y_1\langle a\rangle \mid y_2\langle a\rangle \mid y_2\langle a\rangle \mid y_2\langle a\rangle \mid y_3\langle a\rangle \mid y_3\langle a\rangle \mid y_3\langle a\rangle$$
$$\mid t_1(b).in\ C_1 \mid f_3(b).in\ C_1 \mid f_1(b).in\ C_2 \mid f_2(b).in\ C_2 \mid t_2(b).in\ C_3]$$

The evolution of this term (by applying duplication and communication rules) leads in the first $2n+1 = 2*3+1 = 7$ steps to the generation of all the possible truth assignments over a set of variables $\{x_1, x_2, x_3)$. Since we have described in the previous section how Q_1 evolves to Q_1^5, here we just enumerate the first five obtained ambients. In what follows we bold the capabilities and ambients involved actively in an evolution step.

$$P_1 \mid Q_1 \rightarrow\!\!\stackrel{\phi}{\Rightarrow} \ldots \rightarrow\!\!\stackrel{\phi}{\Rightarrow} P_6 \mid Q_1^5$$

We replace Q_1^5 with $Q_2^3\{t_1/y_1\} \mid Q_2^3\{f_1/y_1\}$ obtaining

$$P_6 \mid Q_2^3\{t_1/y_1\} \mid Q_2^3\{f_1/y_1\}$$

and then we substitute Q_2^3 processes for obtaining

$$\begin{array}{c} P_6 \mid \mathbf{x}\langle\mathbf{z}\rangle \mid \mathbf{x}\langle\mathbf{z}\rangle \mid \mathbf{x}\langle\mathbf{z}\rangle \mid \mathbf{x}\langle\mathbf{z}\rangle \\ \mid \mathbf{x}(\mathbf{y}).open\ k_2 \mid k_2[Q_3^1\{t_1/y_1, t_2/y_2\}] \mid \mathbf{x}(\mathbf{y}).open\ k_2 \mid k_2[Q_3^1\{t_1/y_1, f_2/y_2\}]) \\ \mid \mathbf{x}(\mathbf{y}).open\ k_2 \mid k_2[Q_3^1\{f_1/y_1, t_2/y_2\}] \mid \mathbf{x}(\mathbf{y}).open\ k_2 \mid k_2[Q_3^1\{f_1/y_1, f_2/y_2\}]) \end{array}$$

In the next step the communication on all channels x takes place in parallel, leading to all the possible assignments placed inside ambients A.

$$\begin{array}{l} \rightarrow\!\!\stackrel{\phi}{\Rightarrow} P_7 \mid open\ k_2 \mid k_2[Q_3^2\{t_1/y_1, t_2/y_2\}] \mid open\ k_2 \mid k_2[Q_3^2\{t_1/y_1, f_2/y_2\}] \\ \quad \mid open\ k_2 \mid k_2[Q_3^2\{f_1/y_1, t_2/y_2\}] \mid open\ k_2 \mid k_2[Q_3^2\{f_1/y_1, f_2/y_2\}] \end{array}$$

We replace Q_3^2 processes in order to see how the assignments look.

$$\begin{array}{l} P_7 \mid \mathbf{open\ k_2} \mid \mathbf{k_2}[A\{t_1/y_1, t_2/y_2, t_3/y_3\} \mid A\{t_1/y_1, t_2/y_2, f_3/y_3\}] \\ \quad \mid \mathbf{open\ k_2} \mid \mathbf{k_2}[A\{t_1/y_1, f_2/y_2, t_3/y_3\} \mid A\{t_1/y_1, f_2/y_2, f_3/y_3\}] \\ \quad \mid \mathbf{open\ k_2} \mid \mathbf{k_2}[A\{f_1/y_1, t_2/y_2, t_3/y_3\} \mid A\{f_1/y_1, t_2/y_2, f_3/y_3\}] \\ \quad \mid \mathbf{open\ k_2} \mid \mathbf{k_2}[A\{f_1/y_1, f_2/y_2, t_3/y_3\} \mid A\{f_1/y_1, f_2/y_2, f_3/y_3\}] \end{array}$$

where $A = A[x_1\langle y_1\rangle \mid x_2\langle y_2\rangle \mid x_3\langle y_3\rangle$
$\mid y_1\langle a\rangle \mid y_1\langle a\rangle \mid y_1\langle a\rangle \mid y_2\langle a\rangle \mid y_2\langle a\rangle \mid y_2\langle a\rangle \mid y_3\langle a\rangle \mid y_3\langle a\rangle \mid y_3\langle a\rangle$
$\mid t_1(b).in\ C_1 \mid f_3(b).in\ C_1 \mid f_1(b).in\ C_2 \mid f_2(b).in\ C_2 \mid t_2(b).in\ C_3]$.

From this point forward, the performed steps are:

- Since all the possible assignments are generated, we open in parallel all ambients k_2 such that all ambients A become siblings with the ambient C_1, ready to start the checking stage. Also all possible communications inside ambients A are performed, thus launching the $in\ C_j$ capabilities corresponding to the clauses C_j.
- In the next $m = 3$ steps, the solutions of φ should go inside ambient C_3. First, the solutions of C_1 go inside ambient C_1 in parallel.
- Next, from the solutions of C_1 are selected the solutions of C_2, namely the solutions are all moved inside ambient C_2 in parallel.
- Finally, the solutions of C_3 are selected among the solutions of C_1 and C_2, namely the solutions which move in parallel inside ambient C_3.
- Since we have an ambient A inside ambient C_3, an *yes* answer is released on channel *ans* in the next step. In parallel, ambient K comes out of ambient J, but since ambient L is not present, the *no* answer cannot be sent on channel *ans*.

- Alternatively, if after $2n + m + 1$ steps there is no ambient A inside ambient C_3, then ambient K exits ambient J, and so allowing the ambient L to enter ambient K and to release a *no* answer on channel ans.

5 Conclusion

There are a large number of process calculi used to model complex systems in which interactions and mobility are essential (e.g., [3]). Following this research line, we have previously extended mobile ambients with timers [2] and types [1] in order to study their ability of modelling complex systems in distributed networks. In this paper we use mobile ambients with a parallel semantic (parMA) in order to study their complexity aspects. Thus we provide a semiuniform solution of the SAT problem in a polynomial number of steps by using mobile ambients with a weak form of replication which work according to a parallel semantics.

As far as we know, this is a first attempt to use mobile ambients with parallelism (as they were introduced initially) to create an algorithm that solves an NP-complete problem in a polynomial number of steps. In this way, we show how the mobile ambients can be coordinated to solve problems. There are several topics that could be investigated as further work, including finding other hard problems and complexity classes that can be solved using mobile ambients or related formalisms (process calculi).

Acknowledgements

The work of Bogdan Aman and Gabriel Ciobanu was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0919.

References

1. B. Aman, G. Ciobanu. Mobile Ambients with Timers and Types. *Lecture Notes in Computer Science*, vol.4711, 50–63, 2007.
2. B. Aman, G. Ciobanu. Timed Mobile Ambients for Network Protocols. *Lecture Notes in Computer Science*, vol.5048, 234–250, 2008.
3. B. Aman, G. Ciobanu. *Mobility in Process Calculi and Natural Computation*, Springer, 2011.
4. B. Bloom, S. Istrail, A.R. Meyer. Bisimulation Can't Be Traced: Preliminary Report. *In 15th ACM Symposium on Principles of Programming Languages*, 229–239, 1988.
5. F. Boer, M. Gabbrielli, M. Meo. A Timed Linda Language and its Denotational Semantics. *Fundamenta Informaticae*, vol.63, 2004.
6. A. Borodin. On Relating Time and Space to Size and Depth. *SIAM Journal of Computing*, vol.6, 733–744, 1977.
7. M. Bugliesi, G. Castagna, S. Crafa. Access Control for Mobile Agents: the Calculus of Boxed Ambients. *ACM Transactions on Programming and Systems*, vol.26, 57–124, 2004.

8. N. Busi. On the Computational Power of the Mate/Bud/Drip Brane Calculus: Interleaving vs. Maximal Parallelism. *Lecture Notes in Computer Science*, vol.3850, 144–158, 2006.
9. N. Busi, G. Zavattaro. On the Expressive Power of Movement and Restriction in Pure Mobile Ambients. *Theoretical Computer Science*, vol.322, 477–515, 2004.
10. L. Cardelli. Brane Calculi - Interactions of Biological Membranes. *Lecture Notes in Computer Science*, vol.3082, 257–280, 2004.
11. L. Cardelli, A. Gordon. Mobile Ambients. *Theoretical Computer Science*, vol.240, 177–213, 2000.
12. W. Charatonik, A. Gordon, J.-M. Talbot. Finite-control Mobile Ambients. *Lecture Notes in Computer Science*, vol.2305, 295–313, 2002.
13. G. Ciobanu, V.A. Zakharov. Encoding Mobile Ambients into the π-calculus. *Lecture Notes in Computer Science*, vol.4378, 148–165, 2007.
14. C. Fournet, J.-J. Lévy, A. Schmitt. An Asynchronous, Distributed Implementation of Mobile Ambients. *Lecture Notes in Computer Science*, vol.1872, 348–364, 2000.
15. M. Garey, D. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. Freeman, 1979.
16. D. Hirschkoff, D. Teller, P. Zimmer. Using Ambients to Control Resources. *Lecture Notes in Computer Science*, vol.2421, 21–39, 2002.
17. F. Levi, D. Sangiorgi. Mobile Safe Ambients. *ACM Transactions on Programming and Systems*, vol.25, 1–69, 2003.
18. S. Maffeis, I. Phillips. On the Computational Strength of Pure Ambient Calculi. *Theoretical Computer Science*, vol.330, 501–551, 2005.
19. R. Milner. *Communicating and Mobile Systems : the π-Calculus*. Cambridge University Press, 1999.
20. R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.
21. F. Moller. *Axioms for Concurrency*. PhD Thesis, Department of Computer Science, University of Edinburgh, 1989.
22. C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1995.
23. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science*, vol.325, 141–167, 2004.