The Microsoft Research - University of Trento Centre for Computational and Systems Biology

Technical Report CoSBi 23/2007

A Logical Characterization of Robustness, Mutants and Species in Colonies of Agents

Matteo Cavaliere
The Microsoft Research-University of Trento Centre for Computational and Systems Biology, Trento, Italy
`cavaliere@cosbi.eu`

Radu Mardare
The Microsoft Research-University of Trento Centre for Computational and Systems Biology, Trento, Italy
`mardare@cosbi.eu`

Sean Sedwards
The Microsoft Research-University of Trento Centre for Computational and Systems Biology, Trento, Italy
`sedwards@cosbi.eu`

This is the preliminary version of a paper that will appear in International Journal of Foundations of Computer Science, 19 (5), 2008

A Logical Characterization of Robustness, Mutants and Species in Colonies of Agents

Matteo Cavaliere, Radu Mardare, Sean Sedwards

The Microsoft Research-University of Trento, Italy

September 30, 2009

We study a modeling framework and computational paradigm called Colonies of Synchronizing Agents (CSAs), which abstracts intracellular and intercellular mechanisms of biological tissues. The model is based on a multiset of agents (cells) in a common environment. Each agent has a local contents, stored in the form of a multiset of atomic objects, updated by multiset rewriting rules which may act on individual agents (intracellular action) or synchronize the contents of pairs of agents (intercellular action).

In this paper we investigate dynamic properties of CSAs, by means of temporal logic, and we give a logical characterization of some notions inspired by evolutionary biology such as *robustness*, *mutants* and *species*. We reveal the relation that exists between the concept of robustness for CSAs and the bisimulation relation on colonies. We also present some decidability results for particular cases of robustness.

1 Motivations

Inspired by biological tissues and populations of cells, in our previous work [6, 5] it has been introduced and investigated an abstract distributed model of computation called *Colonies of Synchronizing Agents* (in short, CSAs). The intention is to create a framework to model, analyze and simulate complex biological systems in the context of formal language theory and multiset rewriting.

The model is based on a population of *agents* (e.g., corresponding to *cells* or *molecules*) in a common environment, able to modify their contents and to synchronize with other agents in the same environment. Each agent has a contents represented by a multiset of atomic objects (e.g., corresponding to chemical compounds or the characteristics of individual molecules) with some of the objects classified as terminals (e.g., corresponding to properties or chemicals visible to an external observer). An agent's contents may be modified independently of other agents by means of multiset rewriting rules (called *internal rules*)[1] which can mimic chemistry or other types of *intracellular mechanisms*.

[1]In [6] internal rules are called evolution rules, adopting a standard terminology from the P systems area. We prefer here a more general term.

Moreover, the agents can influence each other by synchronously changing their contents using pairwise *synchronization rules*. This models, in a deliberately abstract way, the various signalling mechanisms and *intercellular mechanisms* present in biological systems. The rules are global, so all agents obey the same rules: the only feature which may distinguish the agents is their contents. Evolutions of CSAs are defined as sequences of transitions obtained by applying the rules to the agents. These transitions thus mark the passage of the system from one configuration to another.

In this paper we investigate *dynamic properties* of CSAs by applying tools from classical fields of computer science, such as formal language, automata theory and temporal logic. In particular, we investigate the *robustness* of properties of CSAs by considering the ability of a CSA to generate a particular *core set of result* despite the failure (i.e., removal/modification) of some of the agents or rules. In a previous work, [6, 5], the core result was defined as a specific configuration in which the colony must halt. In this paper we extend the idea by proposing a more general approach: we introduce a temporal logic (an extension of CTL logic [19, 1]) as a language for specifying properties of CSAs and we define robustness for such properties. CTL is expressive enough to encode both static and dynamic properties and has the advantage of being widely studied and used for model checking analysis by means of software tools, e.g. [21]. We show how robustness can be mapped into model checking and eventually solved by using model checkers.

An other essential element in the definition of robustness is the transformation imposed to the colony. For instance, in [6, 5] the robustness is considered against deletion of agents or of rules of a colony. Formally, these transformations are binary relation on colonies, called *refinement relations*. In this paper we analyze the notion of robustness against any definable refinement relation. In particular, we can define in terms of robustness several concepts inspired by evolutionary biology, such as *mutants* and *species of CSAs*.

2 Formal Language Preliminaries

This Section is a brief introduction to some basic notions of formal language theory needed in the paper. Further information regarding formal language and automata theory is available from the many monographs in this area, starting with [11, 4] and ending with the handbook [18].

Given the set A we denote by $|A|$ its cardinality, by $\mathcal{P}(A)$ its powerset (the set of all subsets of A), and by $\emptyset$ the empty set. We denote by $\mathbb{N}$ the set of natural numbers.

An *alphabet* V is a finite set of symbols. By V^* we denote the set of all strings over V. By V^+ we denote the set of all strings over V excluding the empty string. The empty string is denoted by λ. The *length* of a string v is denoted by $|v|$. The concatenation of two strings $u, v \in V^*$ is written uv. The number of occurrences of the symbol a in the string w is denoted by $|w|_a$.

Each subset of V^* is called a *language*.

The operations (with languages) of union and intersection are denoted $\cup$ and $\cap$, respectively. Concatenation of the languages L_1, L_2 is $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$.

A generative grammar is a finite device generating in a well-specified sense the strings of a language. Chomsky grammars are particular cases of rewriting systems where the operation used in processing the strings is the rewriting (replacement of a substring of the processed string by another substring). A (Chomsky) grammar is a quadruple $G = (N, T, S, P)$ where N and T are disjoint alphabets, N being a set of non-terminals and T a set of terminals, S is the axiom and P is a finite set of productions (rewriting rules). A production is usually written in the form $r : u \mapsto v$ with $u \in (N \cup T)^*$ with u containing at least a non-terminal (so, it cannot be the empty string).

For $x, y \in (N \cup T)^*$ we write $x \mapsto y$ iff $x = x_1ux_2, y = x_1vx_2$ for some $x_1, x_2 \in (N \cup T)^*$ and $u \mapsto v \in P$. One says that x directly derives y. The language generated by G denoted by $L(G)$ is defined by $L(G) = \{x \in T^* \mid S \mapsto^* x\}$, where $\mapsto^*$ denotes the reflexive and transitive closure of $\mapsto$. We also use $\mapsto^+$ for denoting the transitive closure of $\mapsto$. So the language $L(G)$ consists of all terminal strings that can be obtained starting from S by applying iteratively the productions in P.

A grammar is called *regular* if each production is of the form $a \mapsto v$ with $a \in N$ and $v \in T \cup TN \cup \{\lambda\}$. A grammar is called *context-free* if each production is of the form $a \mapsto v$ with $a \in N$.

Languages generated by context-free and regular grammars are called context-free and regular languages, respectively. We denote by CF and REG the families of context-free and regular languages, respectively. Regular languages are those accepted by finite state automata.

In general, when we want to specify a terminal alphabet we add a subscript to the name of the family; e.g., REG_A is the family of all regular languages over the alphabet A.

For a language $L \subseteq V^*$, the set $length(L) = \{|x| \; |x \in L\}$ is called the *length set* of L, denoted by NL.

If FL is an arbitrary family of languages then we denote by NFL the family of length sets of languages in FL (i.e., it is a family of sets of natural numbers). For instance, $NREG$ is the family of length sets of regular languages.

The *Parikh vector* associated with a string $x \in V^*$ with respect to the alphabet $V = \{a_1, a_2, \dots, a_n\}$ is $Ps_V(x) = (|x|_{a_1}, |x|_{a_2}, \dots, |x|_{a_n})$. For $L \subseteq V^*$ we define $Ps_V(L) = \{Ps_V(x) | x \in L\}$. This is called the *Parikh image* of the language L. The null vector is denoted by $\overline{0}$.

If FL is an arbitrary family of languages then we denote by $PsFL$ the family of Parikh images of languages in FL (i.e., it is a family of sets of vectors of natural numbers).

For instance, $PsREG$ is the family of Parikh images of regular languages in REG.

For instance, $V = \{a, b, c\}$ is an alphabet, $x = aaabbbcaa = a^3b^3ca^2$ is a string over V, $L = \{a^nb^n \mid n \geq 1\}$ is a language over V. We have $|x| = 9$, $|x|_a = 5$, $length(L) = \{2n \mid n \geq 1\}$. The Parikh vector of x with respect to V is $Ps_V(x) = (5, 3, 1)$ and for the language L we have $Ps_V(L) = \{(n, n, 0) \mid n \geq 1\}$.

A *multiset* is a set where each element may have a multiplicity. Formally, a multiset over a set V is a map $M : V \mapsto \mathbb{N}$, where $M(a)$ denotes the multiplicity (i.e., number of *occurrences*) of the symbol $a \in V$ in the multiset M. Note that the set V can be infinite.

For instance $M = \{a, b, b, b\}$, also written as $\{(a, 1), (b, 3)\}$, is a multiset with $M(a) = 1$ and $M(b) = 3$.

For multisets M and M' over V, we say that M is *included in* M' ($M \subseteq M'$) if $M(a) \leq M'(a)$ for all $a \in V$. Every multiset includes the *empty multiset*, defined as M where $M(a) = 0$ for all $a \in V$.

The *sum* of multisets M and M' over V is written as the multiset $(M + M')$, defined by $(M +$

$M')(a) = M(a) + M'(a)$ for all $a \in V$. The *difference* between M and M' is written as $(M - M')$ and defined by $(M - M')(a) = max\{0, M(a) - M'(a)\}$ for all $a \in V$. We also say that $(M + M')$ is obtained by *adding* M to M' (or vice versa) while $(M - M')$ is obtained by *removing* M' from M.

For example, given the multisets $M = \{a, b, b, b\}$ and $M' = \{b, b\}$, we can say that M' is included in M, that $(M + M') = \{a, b, b, b, b, b\}$ and that $(M - M') = \{a, b\}$.

The *support* of a multiset M is defined as the set $supp(M) = \{a \in V | M(a) > 0\}$. A multiset with finite support is usually presented as a set of pairs $(x, M(x))$, for $x \in supp(M)$.

The *cardinality* of a multiset M is denoted by $card(M)$ and it indicates the number of objects in the multiset. It is defined in the following way. $card(M)$ is infinite if M has infinite support. If M has finite support then $card(M) = \sum_{a_i \in supp(M)} M(a_i)$, i.e., all the occurrences of the elements in the support are counted.

We denote by $\mathbb{M}(V)$ the set of all possible multisets over V and by $\mathbb{M}_k(V)$ the set of all multisets over V having cardinality k.

For the case that the alphabet V is finite we can use a compact string notation to denote multisets: if $M = \{(a_1, M(a_1)), (a_2, M(a_2)), \ldots, (a_n, M(a_n))\}$ then the string $w = a_1^{M(a_1)} a_2^{M(a_2)} \cdots a_n^{M(a_n)}$ (and all its permutations) precisely identify the symbols in M and their multiplicities. Hence, given a string $w \in V^*$, we can say that it identifies the multiset $\{(a, |w|_a) \mid a \in V\}$. For instance, the string *bab* represents the multiset $\{b, a, b\} = \{(a, 1), (b, 2)\}$ which has cardinality 3. The empty multiset is represented by the empty string, λ.

3 Colonies of Synchronizing Agents

In this section we recall the definitions of colonies of synchronizing agents and of some related concepts as introduced in [5].

A *Colony of Synchronizing Agents* (a CSA) of degree m is a construct $\pi = (A, T, C, R)$.

• A is a finite alphabet of symbols (its elements are called *objects*). $T \subseteq A$ is the set of *terminal objects*.

• An *agent* over A is a multiset over the alphabet A (an agent can be represented by a string $w \in A^*$, since A is finite). C is the *configuration of* π and is a multiset of agents, with $card(C) = m$. [2]

• R is a finite set of *rules* over A. We have *internal rules* of type $u \mapsto v$, with $u \in A^+$ and $v \in A^*$, and *synchronization rules* of the type $\langle u, v\rangle \mapsto \langle u', v'\rangle$ with $u, v \in A^+$ and $u', v' \in A^*$.

We denote by Π the class of all colonies of synchronizing agents.

An occurrence γ of an internal rule $r : u \mapsto v$ can be applied to an agent w by taking a multiset u from w (hence, $u \subseteq w$) and *assigning* it to γ (i.e., assigning the occurrences of the objects in u to γ). The application of an occurrence of rule r to the agent w consists of removing from w the multiset u and then adding the multiset v to the resulting multiset.

An occurrence γ of a synchronization rule $r : \langle u, v\rangle \mapsto \langle u', v'\rangle$ can be applied to the pair of agents w and w' by: (i) taking from w a multiset u (hence, $u \subseteq w$) and *assigning* it to γ; (ii) taking from

[2]Formally, C is a multiset of degree m over the set of all possible agents over A. Hence, $C \in \mathbb{M}_m(\mathbb{M}(A))$.

w' a multiset v (hence, $v \subseteq w'$) and *assigning* it to γ. The application of an occurrence of rule r to the agents w and w' consists of: (i) removing the multiset u from w and then adding the multiset u' to the resulting multiset; (ii) removing the multiset v from w' and then adding the multiset v' to the resulting multiset.

We assume the existence of a *global clock* which marks the passage of units of time for all agents present in the colony.

The *configuration* of the CSA, π, consists of the agents present in the colony at a given time. A single *asynchronous transition* (in short, $asyn$-transition) [3] of π modifies the configuration C of π. The modification is done in exactly one time unit and is obtained by applying the rules in the set R to the agents present in C in an *asynchronous* way. This means that, for each agent w and each pair of agents w' and w'' present in C, the occurrences of the objects of w, w' and w'' are *either* assigned to occurrences of the rules, with the occurrences of the objects and the occurrences of the rules chosen in a non-deterministic way, *or* left unassigned. A single occurrence of an object may only be assigned to a single occurrence of a rule. In other words, in an $asyn$-transition any number of occurrences of rules (zero, one, or more) can be applied to the agents in the configuration C.

An $asyn$-transition applied to $\pi = (A, T, C, R)$ produces a new CSA $\pi' = (A, T, C', R)$. We indicate that π' derives from π as the result of an $asyn$-transition by $\pi \mapsto \pi'$, and we call π' a *future state of* π.

A sequence (possibly infinite) $e = \langle \pi_0, \pi_1, \cdots, \pi_i, \pi_{i+1}, \cdots \rangle$ of CSAs such that $\pi_j \mapsto \pi_{j+1}$ for all $j \geq 0$, is called an *$asyn$-evolution* of π_0. An $asyn$-evolution e of π_0 is said to be *halting* if it halts, that is if it is finite and the configuration of the last CSA of e is a *halting configuration*, (i.e., a configuration containing only agents for which no occurrences of rules from R can be applied). In this case the last CSA of the sequence is called *halted CSA*.

An $asyn$-evolution of a CSA π_0 is called *complete* if it is either halting or infinite. Let $\mathcal{E}^{asyn}(\pi_0)$ be the set of all $asyn$-complete evolutions of π_0. Given an arbitrary $asyn$-complete evolution of π_0, $e = \langle \pi_0, \pi_1, \cdots, \pi_i, \pi_{i+1} \cdots \rangle \in \mathcal{E}^{asyn}(\pi_0)$, a *$j$-suffix evolution of* e, defined for some $j \geq 0$ and denoted by e_j, is the sequence[4] $e_j = \langle \pi_j, \pi_{j+1}, \cdots \rangle$.

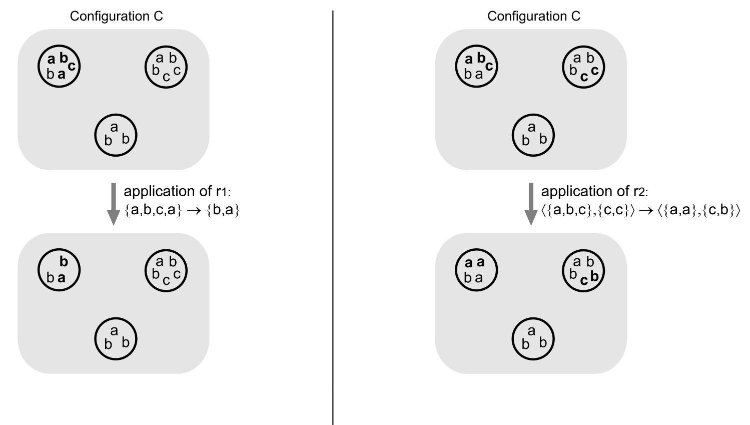
An $asyn$-evolution of π_0 that is halting is called an *$asyn$-computation* of π_0. The *result/output* of an $asyn$-computation of π_0, $\langle \pi_0, \pi_1, \cdots, \pi_h \rangle$ is the *set of vectors of natural numbers* associated with the agents present in the configuration of the halting CSA π_h. Precisely, there is one vector, for each agent w, which describes the multiplicities of terminal objects present in w. More formally, the result of an $asyn$-computation of π_0, $\langle \pi_0, \pi_1, \cdots \pi_h \rangle$ with $\pi_h = (A, T, C_h, R)$, is the set of vectors of natural numbers $\{Ps_T(w) \mid w \text{ is an agent present in } C_h\}$.

Because of the non-determinism in applying the rules, several possible $asyn$-computations of π_0 may exist. Taking the union of all the results for all possible $asyn$-computations of π_0, we get the *set of vectors generated* by π_0, denoted by $Ps_T^{asyn}(\pi_0)$.

We may also consider the total number of objects comprising the agent (the agent's *magnitude*), without considering the internal composition. In this case the *result* of an $asyn$-computation of π_0,

[3] We specify $asyn$-transitions to distinguish them from the synchronous maximal parallel transitions often adopted in models coming from P systems and cellular automata.

[4] Observe that e_j is an $asyn$-complete evolution of π_j.



(a) Internal rule r_1 applied to C | (b) Synchronization rule r_2 applied to C

Figure 1: Alternative application of rules r_1 and r_2 to configuration C from Example 3.1.

$\langle\pi_0, \cdots \pi_h\rangle$ is the set of the lengths of the agents present in the configuration of π_h. More formally, in this case the result of an asyn-computation of π_0, $\langle\pi_0, \cdots \pi_h\rangle$ with $\pi_h = (A, T, C_h, R)$, is the set of numbers $\{|w| \mid w$ is an agent present in $C_h\}$. Again, taking the union of all the results for all possible $asyn$-computations of π_0, we get the *set of numbers generated* by π_0, denoted $N^{asyn}(\pi_0)$.

Example 3.1 *A CSA π with degree 3 is defined in the following way.*
$\pi = (A, T, C, R)$ with $A = \{a, b, c\}$, $T = \{a\}$, $C = \{(abcba, 1), (abbcc, 1), (bab, 1)\}$ = $\{abcba, abbcc, bab\}$.
The rules $R = \{r_1 : abca \mapsto ba,\ r_2 : \langle abc, cc\rangle \mapsto \langle aa, cb\rangle\}$.

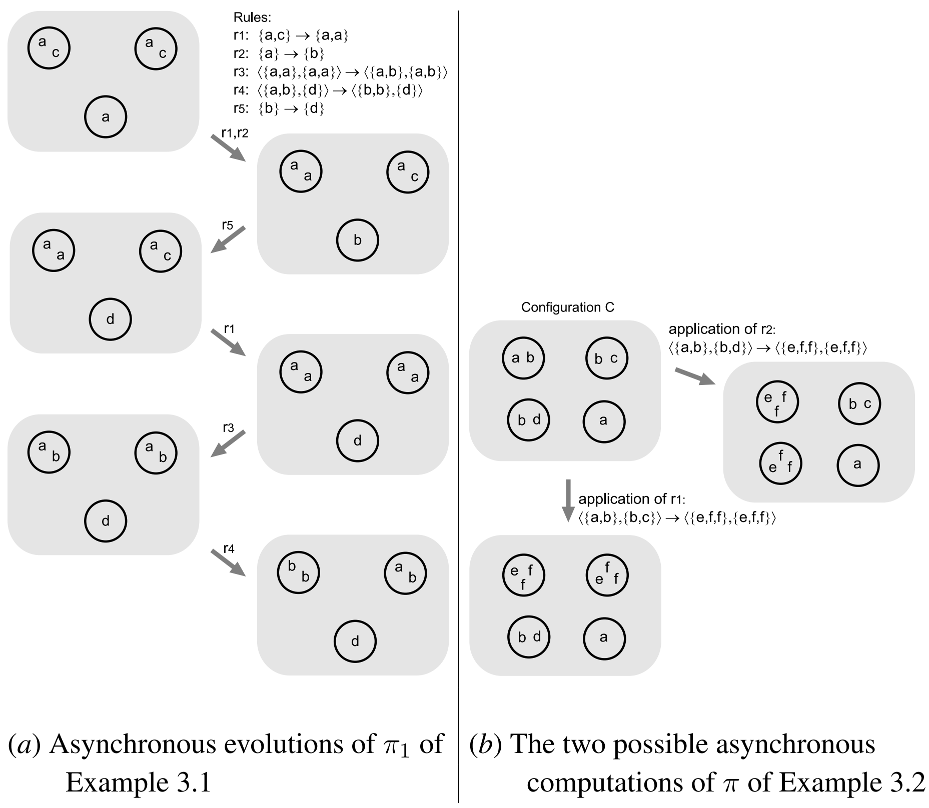
The application of an occurrence of internal rule r_1 to the agent $abcba$ in the configuration C is shown diagrammatically in Figure 1(a). The CSA obtained as a result of this $asyn$-transition is $\pi' = (A, T, C', R)$ with $C' = \{(bab, 1), (abbcc, 1), (bab, 1)\}$ = $\{bab, abbcc, bab\}$.

The application of an occurrence of the synchronization rule r_2 to the pair of agents $abcba$ and $abbcc$ in the configuration C is shown diagrammatically in Figure 1(b). The CSA obtained as a result of this $asyn$-transition is $\pi' = (A, T, C', R)$ with $C' = \{(aaba, 1), (abbcb, 1), (bab, 1)\}$ = $\{aaba, abbcb, bab\}$.

A more complex example of part of an asynchronous evolution is presented in Figure 2(a): $\pi_1 = (A, T, C, R)$ with $C_1 = \{(ac, 2),\ (a, 1)\}$ and rules $R = \{ac \mapsto aa,\ a \mapsto b,\ \langle aa, aa\rangle \mapsto \langle ab, ab\rangle,\ \langle ab, d\rangle \mapsto \langle bb, d\rangle,\ b \mapsto d\}$. A possible evolution can be $\langle\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \cdots\rangle$, where $\pi_i = (A, T, C_i, R)$ with $C_2 = \{(aa, 1), (ac, 1), (b, 1)\}$, $C_3 = \{(aa, 1), (ac, 1), (d, 1)\}$, $C_4 = \{(aa, 2), (d, 1)\}$, $C_5 = \{(ab, 2), (d, 1)\}$, $C_6 = \{(bb, 1), (ab, 1), (d, 1)\}$ etc.

In the next Example we show how the output/result of a CSA is obtained.

Example 3.2 *Consider a CSA $\pi = (A, T, C, R)$ with $A = \{a, b, c, d, e, f\}$, $T = \{e, f\}$, $C = \{(ab, 1), (bc, 1), (bd, 1), (a, 1)\}$. The rules in R are*
$\{r_1 : \langle ab, bc\rangle \mapsto \langle eff, eff\rangle, r_2 : \langle ab, bd\rangle \mapsto \langle eff, eff\rangle\}$.


Rules:
r1: {a,c} → {a,a}
r2: {a} → {b}
r3: ⟨{a,a},{a,a}⟩ → ⟨{a,b},{a,b}⟩
r4: ⟨{a,b},{d}⟩ → ⟨{b,b},{d}⟩
r5: {b} → {d}
r1,r2
r5
r1
r3
r4
Configuration C
application of r2:
⟨{a,b},{b,d}⟩ → ⟨{e,f,f},{e,f,f}⟩
application of r1:
⟨{a,b},{b,c}⟩ → ⟨{e,f,f},{e,f,f}⟩

(a) Asynchronous evolutions of π_1 of Example 3.1

(b) The two possible asynchronous computations of π of Example 3.2

Figure 2: Asynchronous evolutions and computations.

There are only two possible asynchronous computations *of* π *and these are represented diagrammatically in Figure 2(b).*

We have that $Ps_T^{asyn}(\pi) = \{(1,2), \overline{0}\}$.

The first $asyn$*-computation has a halting configuration with the agent (in two copies)* eff *whose associated Parikh vector (with respect to* T*) is* $(1,2)$ *and the agents* bd *and* a*, whose associated Parikh vectors (with respect to* T*) are null vectors* $\overline{0}$ *(these agents do not contain any terminal object from* T*). Then the result of this* $asyn$*-computation is the set of vectors* $\{(1,2)\} \cup \{(1,2)\} \cup \{\overline{0}\} \cup \{\overline{0}\} = \{(1,2), \overline{0}\}$ *with each vector describing the multiplicities of the terminal objects in the agents in the halting configuration.*

The second halting $asyn$*-computation has a halting configuration with the agent (in two copies)* eff *whose associated Parikh vector (with respect to* T*) is* $(1,2)$ *and the agents* bc *and* a*, whose associated Parikh vectors (with respect to* T*) are null vectors. Then, also in this case, the result of the* $asyn$*-computation is the set of vectors* $\{(1,2), \overline{0}\}$.

Taking the union of the results for the (two) possible $asyn$*-computations of* π *we get* $Ps_T^{asyn}(\pi) = \{(1,2), \overline{0}\} \cup \{(1,2), \overline{0}\} = \{(1,2), \overline{0}\}$.

We can also collect the result in terms of magnitude (size) of the agents, thus collecting $N^{asyn}(\pi)$*. In this case we obtain* $N^{asyn}(\pi) = \{3,2,1\}$*. In fact, both* $asyn$*-computations halts with agents of size*

$3, 2$ *and* 1 *(counting their objects). Then for both* $asyn$*-computations the result is the set of numbers* $\{3, 3, 2, 1\} = \{3, 2, 1\}$ *with each number being the magnitude of an agent in the halting configuration.*

Taking the union of the results for the (two) possible $asyn$*-computations of* π *we obtain* $N^{asyn}(\pi) = \{3, 2, 1\} \cup \{3, 2, 1\} = \{3, 2, 1\}$.

4 Dynamic properties of CSAs

In this Section we investigate dynamic properties of CSAs and we define the notion of *robustness*. This notion will allow us to characterize the notions of *species of colonies* and *mutants for species* that are inspired by evolutionary biology.

4.1 A specification language for properties of colonies

In this subsection we introduce a language that allows us to formally specify properties of CSAs used to define robustness conditions. The language developed is a version of *computational tree logic (CTL temporal logic)*. Temporal logics are the most used logics in model-checking analysis: efficient algorithms and tools having already been developed for them, e.g. NuSMV [21]. They are devised with operators for expressing and quantifying on possible evolutions or configurations of systems. For instance, for an arbitrary system it is possible to specify properties such as *'for any possible evolution, ϕ is fulfilled'*, *'there exists an evolution such that ϕ is not true'*, *'in the next state ϕ will be satisfied'*, *'eventually ϕ will be satisfied'* and *'ϕ happens until ψ is satisfied'*, where ϕ and ψ properties of the system. An introduction to the basic notions and results of temporal logics can be found in [1, 19].

Hereafter, we show how to use these operators to formally specify and verify complex properties of CSAs, such as *'the agent will always eventually reach a certain configuration'*, or *'rule r is not applicable until rule r' is used'*, etc. The specification language is constructed, starting from a given set of properties called *atomic properties*, by combining them with boolean and temporal operators. For example, if ϕ is a property, then its logical negation, $\neg\phi$, is also a property which will be satisfied by any CSA that does not have the property ϕ. Similarly we can consider disjunction, conjunctions or other boolean combination of properties.

The temporal properties are more powerful, expressing properties of future states of a colony. As the $asyn$-evolution of a colony is nondeterministic, the future of such a system is usually branching, e.g., at any step of the evolution a colony can have more than one possible future. Consequently, for expressing properties of future states it is necessary to combine quantifiers over possible evolutions of a CSA with quantifiers over possible states of an evolution, such as in *"there is an evolution e of π where eventually ϕ will be satisfied"*. The language $\mathcal{L}_S$ introduced in the next definition contains *configuration formulas* that will be evaluated against CSAs, and *evolution formulas* to be evaluated against evolutions of CSAs.

Definition 4.1 (Syntax) *Let S be a set of* atomic properties. *The language $\mathcal{L}_S$, constructed for S, contains* configuration formulas *and* evolution formulas *which are defined, inductively, as follows:*

- *any atomic property $s \in S$ is a configuration formula of $\mathcal{L}_S$;*
- *if ϕ, ψ are configuration formulas of $\mathcal{L}_S$, then $\neg\phi$ and $\phi \wedge \psi$ are configuration formulas of $\mathcal{L}_S$;*
- *if ϕ is an evolution formula of $\mathcal{L}_S$ then $E\phi$ is a configuration formula of $\mathcal{L}_S$;*
- *if ϕ, ψ are configuration formulas of $\mathcal{L}_S$ then $\phi U \psi$ is an evolution formulas of $\mathcal{L}_S$.*

These atomic operators can be combined in order to derive more complex operators.

1. $\bot = \phi \wedge (\neg\phi)$	2. $\top = \neg\bot$	3. $\phi \vee \psi = \neg((\neg\phi) \wedge (\neg\psi))$
4. $\phi \rightarrow \psi = (\neg\phi) \vee \psi$	5. $X\phi = \bot U \phi$	6. $A\phi = \neg(E\neg\phi)$
7. $F\phi = \top U \phi$	8. $G\phi = \neg(F\neg\phi)$	

The meanings of the operators are the following ones.

The boolean operators $\vee$ (or), $\rightarrow$ (implies), $\top$ (true) and $\bot$ (false) have the meaning from classic propositional logic, as our language is an extension of propositional logic.

The operator E, in the formula $E\phi$ (where ϕ is an evolution formula), is the *existential quantifier* over the possible evolutions of a colony. Thus, $E\phi$ is read *"there exists an evolution of the analyzed colony that has the property ϕ"*. The (derived) operator A is the universal quantifier over evolutions, defined as the dual of E. Used together with an evolution formula ϕ, A defines the configuration formula $A\phi$ that characterizes a CSA π when every possible evolution of π has the property ϕ.

Two configuration formulas ϕ, ψ can be combined by mean of U (*Until*) operator generating the evolution formula $\phi U \psi$, read *"ϕ until ψ"*. An evolution of a colony π has the property $\phi U \psi$ if a future state of π, during this evolution, has the property ψ while all its previous states satisfy ϕ.

The formula $X\phi$, where ϕ is a configuration formula, is an evolution formula satisfied by an evolution of a colony π when ϕ is satisfied at the next state in this evolution. Thus, $X\phi$ is read *"at the next state ϕ is satisfied"*.

The evolution formula $F\phi$ characterizes an evolution e of a CSA π if there is a future state π' of π in this evolution, such that π' has the property ϕ. For this reason $F\phi$ is read *"finally (eventually) ϕ"*. G is the dual of F, and $G\phi$, read *"globally ϕ"* is a property of an evolution e of π when all the elements of e have the property ϕ.

Combining A, E quantifiers with evolution formulas, we obtain complex configuration formulas. For instance the configuration formula $E\phi U \psi$ (*"exists an evolution where ϕ until ψ"*) characterizes the CSA π iff there exists at least one evolution e of π such that, during this evolution, the colony will eventually reach a state which satisfies ψ while all the previous states satisfy ψ.

We now formalize all the intuitions about the expressiveness of these operators. We start by introducing the notion of *model*, which is used for defining the *satisfiability relation* that associates a property ϕ (expressed by a configuration formula in $\mathcal{L}_S$) to a CSA π.

Definition 4.2 (Models) *A* model over *S is a pair $\mathcal{M} = \langle M, i \rangle$ where $M \subseteq \Pi$ is a set of CSAs such that if $\pi \in M$ and $\pi \mapsto^+ \pi'$, then $\pi' \in M$, and $i : M \rightarrow \mathcal{P}(S)$ is the* interpretation function *that associates to each colony from M a set of atomic properties.*

The reason for which we introduce models is that, in some real applications we might be interested in studying the properties of a predefined set of colonies. The closure conditions imposed to M with respect to the transitive closure of $\mapsto$ guarantees that once a CSA is present in the chosen model, all its possible evolution are present too. As consequence, if $\mathcal{M} = \langle M, i\rangle$ is a model and $\pi \in M$, then all the CSAs present in an arbitrary complete $asyn$-evolution $e \in \mathcal{E}^{asyn}(\pi)$ are also in M, and this is denoted by the short notation $e > \mathcal{M}$.

The next definition introduces the satisfiability relation that defines when a property (expressed in $\mathcal{L}_S$) is satisfied by a colony in a given model.

Definition 4.3 (Semantics) *The satisfiability relation for $\mathcal{L}_S$ against a model $\mathcal{M} = \langle M, i\rangle$ is defined, for $\pi \in M$ and for $e > \mathcal{M}$, inductively by:*

$\mathcal{M}, \pi \models s$, for some $s \in S$, iff $\pi \in i(s)$.

$\mathcal{M}, \pi \models \neg\phi$ iff $\mathcal{M}, \pi \not\models \phi$.

$\mathcal{M}, \pi \models \phi \wedge \psi$ iff $\mathcal{M}, \pi \models \phi$ and $\mathcal{M}, \pi \models \psi$.

$\mathcal{M}, \pi \models E\phi$ iff there exists $e \in \mathcal{E}^{asyn}_{\pi}$ such that $\mathcal{M}, e \models \phi$.

$\mathcal{M}, e \models \phi U \psi$ for $e = \langle \pi_0, \pi_1, \dots \pi_k, \dots\rangle$, iff there exists $i \geq 0$ such that $\mathcal{M}, \pi_i \models \psi$ and for all $j \leq i$ $\mathcal{M}, \pi_j \models \phi$.

Consequently, the semantics of the derived formulas are the following.

$\mathcal{M}, \pi \models \phi \vee \psi$ iff $\mathcal{M}, \pi \models \phi$ or $\mathcal{M}, \pi \models \psi$.

$\mathcal{M}, \pi \models \phi \rightarrow \psi$ iff $\mathcal{M}, \pi \models \phi$ implies $\mathcal{M}, \pi \models \psi$.

$\mathcal{M}, \pi \models \top$ always.

$\mathcal{M}, \pi \models \bot$ never.

$\mathcal{M}, e \models X\phi$ for $e = \langle \pi_0, \pi_1, \dots \pi_k, \dots\rangle$, iff $\mathcal{M}, \pi_1 \models \phi$.

$\mathcal{M}, \pi \models A\phi$ iff for any $e \in E^{asyn}_{\pi}$ we have $\mathcal{M}, e \models \phi$.

$\mathcal{M}, e \models F\phi$ for $e = \langle \pi_0, \pi_1, \dots \pi_k, \dots\rangle$, iff there exists $i \geq 0$ such that $\mathcal{M}, e_i \models \phi$.

$\mathcal{M}, e \models G\phi$ for $e = \langle \pi_0, \pi_1, \dots \pi_k, \dots\rangle$, iff for any $i \geq 0$ we have $\mathcal{M}, e_i \models \phi$.

We now introduce some concepts from model theory that will be useful in defining the robustness and the related concepts.

Definition 4.4 (Validity and satisfiability) *A configuration formula ϕ (evolution formula ϕ) from $\mathcal{L}_S$ is* valid*, denoted by $\models \phi$, iff for every model $\mathcal{M} = \langle M, i\rangle$ and any $\pi \in M$ (any $e > M$, resp.) we have $\mathcal{M}, \pi \models \phi$ ($\mathcal{M}, e \models \phi$, resp.). A configuration formula ϕ (evolution formula ϕ) is* satisfiable *iff there exist a model $\mathcal{M} = \langle M, i\rangle$ and a colony $\pi \in M$ (an evolution $e > M$, resp.) such that $\mathcal{M}, \pi \models \phi$ ($\mathcal{M}, e \models \phi$, resp.).*

A configuration formula ϕ (evolution formula ϕ) from $\mathcal{L}_S$ is valid in the model *$\mathcal{M} = \langle M, i\rangle$, denoted by $\models_{\mathcal{M}} \phi$, iff for every $\pi \in M$ (for every $e > M$, resp.) we have $\mathcal{M}, \pi \models \phi$ ($\mathcal{M}, e \models \phi$, resp.). A configuration formula ϕ (evolution formula ϕ) is* satisfiable in the model *$\mathcal{M} = \langle M, i\rangle$ iff there exists a colony $\pi \in M$ (an evolution $e > M$, resp.) such that $\mathcal{M}, \pi \models \phi$ ($\mathcal{M}, e \models \phi$, resp.).*

The concept of satisfiability characterizes a property that is consistent, i.e., it is not trivially false, in the sense that there exists at least a colony satisfying such a property. On the other hand, validity stays for universally true, i.e., any colony satisfy the property. Both concepts can be made "local" to the model. Thus satisfiability in a model characterizes a property for which there exists at least one colony in that model that has the property. Consequently, not any consistent property is a satisfiability property with respect to a given model. In the same way, validity with respect to a model can be stated about a property only if all the colonies in the model have this property. In this way, the valid properties in a model can be used to characterize the colonies included in the model and thus to differentiate them from the other colonies.

A *model-checking problem* is defined as the problem to decide, for an arbitrary model $\mathcal{M} = \langle M, i \rangle$ over S, an arbitrary colony $\pi \in M$ (or an arbitrary evolution $e > \mathcal{M}$), and an arbitrary property $\phi \in \mathcal{L}_S$, whether or not $\mathcal{M}, \pi \models \phi$ ($\mathcal{M}, e \models \phi$, respectively). Informally this means to decide whether or not the colony π satisfies the property ϕ in the model $\mathcal{M}$ (if the evolution e satisfies the property ϕ in $\mathcal{M}$, respectively).

Theorem 4.1 (Decidability) *For a finite set of atomic properties S, the satisfiability, validity and model-checking problems for $\mathcal{L}_S$ against the semantics defined on CSAs are decidable.*

Proof The result derives from the fact that CTL logic is decidable (see, e.g., [19, 1]) for a finite set of atomic propositions. □

The previous theorem does not take in account the complexity issues related to the construction of the model $\mathcal{M}$, to the membership problems in $\mathcal{M}$, or to the costs of checking satisfiability for an atomic property $s \in S$. Theorem 4.1 states that, if $\mathcal{M}$ is given (in a descriptive way), and if we assume that satisfiability of atomic properties is decidable, than validity, satisfiability and model-checking problems are also decidable. In our case, as CSAs are given in a grammar-like form, additional complexity issues might rise, and in the next sections we will underline some of these problems.

5 Robustness

We are now ready to define and to investigate robustness of properties of CSAs against perturbations or modifications of some of the features of the colonies. We generalize the concept of robustness introduced in [5]. We are interested in studying the relationship between a given class of properties and the structure of a colony having these properties. In particular, we want to characterize when a given class of properties Φ is preserved under some structural modification of the colonies, such as removing the ability to act of some agents (i.e., cells) or replacing them with different agents, or removing/modifying some of the evolution rules (i.e., intra or intercellular actions).

In an abstract way, we assume a class S of properties, the *atomic properties*, that generate, by logical constructs, the properties to be tested for invariance against modifications of CSAs. Robustness is defined for a given model $\mathcal{M} = \langle M, i \rangle$ over S (the interpretation function i associates to a colony the set of atomic properties that it satisfies), against some modifications of the colonies in M. In [5] the modifications analyzed were the elimination of one or more agents from the configuration of a colony, and the elimination of some evolution rules. In this paper we propose a more general approach. For this reason, we introduce a *refinement relation* on the model, that is a relation on M. This relation defines the modifications against which the robustness of some properties will be defined. For instance, we can define the refinement relation by deleting/modifying some agents, or by deleting/modifying some of the production rules. Each of these refinements relations defines a specific robustness.

Definition 5.1 (Robustness) *Let S be a set of* atomic properties, *$\Phi \subseteq \mathcal{L}_S$, $\mathcal{M} = \langle M, i\rangle$ a model over S, and $\mathcal{R} \subseteq M \times M$ a relation on M called* refinement.
$\mathcal{M}$ is robust *for Φ against the refinement relation $\mathcal{R}$ if for any $\phi \in \Phi$ and for any $\pi, \pi' \in M$ such that $(\pi, \pi') \in \mathcal{R}$, $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$.*
A model $\mathcal{M}$ is globally robust *against $\mathcal{R}$ if it is robust for $\mathcal{L}_S$ against $\mathcal{R}$.*

Notice that the global robustness ensures the invariability against refinements for any definable temporal property.

We propose some characterization results for robustness.

Proposition 5.1 *For an arbitrary model $\mathcal{M} = \langle M, i\rangle$ over S, an arbitrary refinement relation $\mathcal{R} \subseteq M \times M$, and an arbitrary set of properties $\Phi \subseteq \mathcal{L}_S$, the following assertions are equivalent.*

1. *$\mathcal{M}$ is robust for Φ against $\mathcal{R}$.*
2. *For any $\phi \in \Phi$ and for any $(\pi, \pi') \in \mathcal{R}$, $\mathcal{M}, \pi \not\models \phi$ iff $\mathcal{M}, \pi' \not\models \phi$.*
3. *For any $\phi \in \Phi$ and for any $(\pi, \pi') \in \mathcal{R}$, if $\mathcal{M}, \pi \models \phi$, then $\mathcal{M}, \pi' \models \phi$ and if $\mathcal{M}, \pi \not\models \phi$, then $\mathcal{M}, \pi' \not\models \phi$.*

Proof $(1 \implies 2)$ Suppose that $\mathcal{M}$ is robust for Φ against $\mathcal{R}$, that $(\pi, \pi') \in \mathcal{R}$, and for some $\phi \in \Phi$, $\mathcal{M}, \pi \not\models \phi$. We have to prove that $\mathcal{M}, \pi' \not\models \phi$. Suppose that $\mathcal{M}, \pi' \models \phi$. As $(\pi, \pi') \in \mathcal{R}$ and as $\mathcal{M}$ is robust against $\mathcal{R}$, we derive $\mathcal{M}, \pi \models \phi$ that is in contradiction with the first hypothesis. Consequently we cannot have $\mathcal{M}, \pi' \models \phi$, i.e., $\mathcal{M}, \pi' \not\models \phi$. Similarly can be proved that if $\mathcal{M}, \pi' \not\models \phi$, then $\mathcal{M}, \pi \not\models \phi$ that completes this part of the proof.

$(2 \implies 3)$ Suppose that for any $\phi \in \Phi$ and for any $(\pi, \pi') \in \mathcal{R}$, $\mathcal{M}, \pi \not\models \phi$ iff $\mathcal{M}, \pi' \not\models \phi$. From here we derive that if $\mathcal{M}, \pi \not\models \phi$, then $\mathcal{M}, \pi' \not\models \phi$, so second part of (3) is proved. We prove now that if $\mathcal{M}, \pi \models \phi$, then $\mathcal{M}, \pi' \models \phi$. Suppose that $\mathcal{M}, \pi \models \phi$ and that $\mathcal{M}, \pi' \not\models \phi$. But from $\mathcal{M}, \pi' \not\models \phi$ we derive, using (2), $\mathcal{M}, \pi \not\models \phi$, which contradicts our hypothesis. Hence, $\mathcal{M}, \pi \models \phi$ implies $\mathcal{M}, \pi' \models \phi$.

$(3 \implies 1)$ Suppose that for any $\phi \in \Phi$ and for any $(\pi, \pi') \in \mathcal{R}$, if $\mathcal{M}, \pi \models \phi$, then $\mathcal{M}, \pi' \models \phi$ and if $\mathcal{M}, \pi \not\models \phi$, then $\mathcal{M}, \pi' \not\models \phi$. We have to prove that $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$. From the hypothesis (3) we have that if $\mathcal{M}, \pi \models \phi$, then $\mathcal{M}, \pi' \models \phi$. We have to prove that if $\mathcal{M}, \pi' \models \phi$, then $\mathcal{M}, \pi \models \phi$. If we suppose that this is not true, i.e., that $\mathcal{M}, \pi' \models \phi$ and $\mathcal{M}, \pi \not\models \phi$, we derive, using (3), that $\mathcal{M}, \pi' \not\models \phi$ - contradiction. □

Proposition 5.1 underlines a difference with respect to our previous work, [5], where robustness was defined only by the one-way implication *"if π has the property ϕ, then any refinement of it, π',* *has the property ϕ".* This was rather incomplete as, intuitively, robustness for a property has to be sensitive to negation. The robustness introduced in this paper is taking into account all the possible logical combination of the considered properties. For instance if $\mathcal{M}$ is robust for ϕ and for ψ against the refinement $\mathcal{R}$, then $\mathcal{M}$ is robust also for $\phi \wedge \psi$ or for $\neg\phi$. Playing with negation is less intuitive, but, indeed robustness for ϕ is equivalent with robustness for $\neg\phi$ as robustness means that *"π and π' either have both the property ϕ, or neither of them have the property ϕ".* The following results summarize this intuition.

Definition 5.2 (Boolean closure) *Let S be a set of atomic properties and $\Phi \subseteq \mathcal{L}_S$. The boolean closure of Φ, denoted by $\overline{\Phi}$, is the smallest set of formulas such that $\Phi \subseteq \overline{\Phi}$ and if $\phi, \psi \in \overline{\Phi}$, then $\neg\phi, \phi \wedge \psi \in \overline{\Phi}$.*

Proposition 5.2 *A model $\mathcal{M} = \langle M, i \rangle$ is robust for Φ against $\mathcal{R}$ iff $\mathcal{M}$ is robust for $\overline{\Phi}$ against $\mathcal{R}$, where $\overline{\Phi}$ is the boolean closure of Φ.*

Proof If $\mathcal{M}$ is robust for $\overline{\Phi}$ against $\mathcal{R}$, then $\mathcal{M}$ is robust for Φ against $\mathcal{R}$, as $\Phi \subseteq \overline{\Phi}$. We prove the reverse implication. Let $\phi, \psi \in \Phi$ and let $\pi, \pi' \in M$ be two arbitrary colonies such that $(\pi, \pi') \in \mathcal{R}$. As $\mathcal{M}$ is robust for Φ against $\mathcal{R}$, we have that $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$ and $\mathcal{M}, \pi \models \psi$ iff $\mathcal{M}, \pi' \models \psi$. Then, by the defined semantics, we also have $\mathcal{M}, \pi \models \neg\phi$ iff $\mathcal{M}, \pi' \models \neg\phi$ and $\mathcal{M}, \pi \models \phi \wedge \psi$ iff $\mathcal{M}, \pi' \models \phi \wedge \psi$. Since $\overline{\Phi}$ is the boolean closure of Φ, then for any $\rho \in \overline{\Phi}$ we have $\mathcal{M}, \pi \models \rho$ iff $\mathcal{M}, \pi' \models \rho$. □

The next theorem states that a model is robust for all its "universal truths" against any possible refinement.

Proposition 5.3 *Let $\mathcal{M} = \langle M, i \rangle$ be a model over S. $\mathcal{M}$ is robust for the class of formulas in $\mathcal{L}_S$ that are validities of $\mathcal{M}$, against any definable refinement.*

Proof A validity of $\mathcal{M}$ is a formula ϕ such that for any $\pi \in M$ we have $\mathcal{M}, \pi \models \phi$. Hence, if $\mathcal{R} \subseteq M \times M$ is a refinement and $(\pi, \pi') \in \mathcal{R}$, then $\mathcal{M}, \pi \models \phi$ and $\mathcal{M}, \pi' \models \phi$. Hence, $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$. □

From the previous two propositions we can derive a corollary that somehow describes the class of validities as a "propagation mechanism" for the class of robust properties.

Corollary 5.1 (Propagation) *If a model $\mathcal{M} = \langle M, i \rangle$ over S is robust for ϕ against a refinement $\mathcal{R}$, and if $\models_M \phi \to \psi$ (or $\models \phi \to \psi$), then $\mathcal{M}$ is robust for ψ against $\mathcal{R}$.*

Proof We prove that for an arbitrary pair of colonies $(\pi, \pi') \in \mathcal{R}$ we have $\mathcal{M}, \pi \models \psi$ iff $\mathcal{M}, \pi' \models \psi$. From $\models_M \phi \to \psi$, as well as from $\models \phi \to \psi$, we derive $\mathcal{M}, \pi \models \phi \to \psi$ and $\mathcal{M}, \pi' \models \phi \to \psi$, which implies that $\mathcal{M}, \pi \models \phi \to \psi$ iff $\mathcal{M}, \pi' \models \phi \to \psi$. As $\mathcal{M}$ is robust for ϕ, we also have $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$. Combining these results we derive that $\mathcal{M}, \pi \models \psi$ iff $\mathcal{M}, \pi' \models \psi$. □

The next definition exploits the previous propositions to introduce formally the *core* of a set of properties. In this way, we can generalize the notion of *core result* (introduced in [5]).

Definition 5.3 *A* core *of a set of properties* $\Phi \subseteq \mathcal{L}_S$ *is a set* $\Psi \subseteq \Phi$ *such that* $\overline{\Phi} = \overline{\Psi}$ *and for any* $\Theta \subsetneq \Psi$, $\overline{\Theta} \subsetneq \overline{\Psi}$.

Note that the core of a set of properties is not necessary unique.

The next theorem confirms the intuition that a core of a set of properties is indeed a minimal sub-class (with respect to inclusion) that has to be checked for robustness in order to infer the robustness for the entire class of properties. Consequently, checking robustness for a set of properties can be as efficient as checking robustness for the smallest core.

Theorem 5.4 *Let* $\mathcal{M} = \langle M, i \rangle$ *be a model over* S, $\Phi \subseteq \mathcal{L}_S$ *a set of properties and* $\Psi \subseteq \Phi$ *a core of* Φ. $\mathcal{M}$ *is robust for* Φ *against the refinement* $\mathcal{R}$ *iff* $\mathcal{M}$ *is robust for* Ψ *against* $\mathcal{R}$.

Proof The result follows from Definition 5.3 and Proposition 5.2. □

6 Robust properties: Case studies

This section is dedicated to the analysis of some specific cases of robustness properties of CSAs. In particular, we analyze the case of agent-restriction (where the refinement is defined by removing some agents of the colony), and the case of rule-restriction (where the refinement is based on removing rules of the colony).

The case of agent-restriction is interesting as it formalizes the overall behavior of the colony for the case in which some of the agents cease to work. Therefore, using the presented definitions, a property is robust against agent-restriction of a colony only if is a property that still characterizes the colony after some agents are eliminated.

On the other hand, we also investigate the case of rule-restriction that can be used to represent possible *"mutations"* of the rules used by a colony. In this respect, following the presented definition, a property is robust against rule-restriction if it characterizes the colony and its possible *"mutants"*.

Eventually, we will take advantage of the use of robustness to formally define the concept of species of CSA.

6.1 Agent-Restriction

Let $\pi = (A, T, C, R)$ be an arbitrary CSA. We say that π' is an *agent-restriction* of π if $\pi' = (A, T, C', R)$ with $C' \subseteq C$. This means that π' is a CSA where some of the agents originally present in π no longer "work", i.e., as though they are absent from the colony.

We use Definition 5.1 to define the notion of robustness against agent-restriction for properties such as *"part of the result produced by a CSA belongs to a given core set S from* $PsREG$*"*.

For doing that, we introduce the model $\mathcal{M} = \langle M, i \rangle$, where M is the smallest set of CSAs that contains the colonies of interest, and which is closed under $asyn$-evolutions; $i : M \to \mathcal{P}(S)$ is the interpretation function defined by $i(\pi) = Ps_T^{asyn}(\pi) \cap S$ if π is a halted CSA, and $i(\pi) = \emptyset$ else. All these can be written, in terms of satisfiability relation for $\pi \in \mathcal{M}$ and $s \in S$, as

$\mathcal{M}, \pi \models s$ iff π is halted and $s \in Ps_T^{asyn}(\pi)$.

We also define the refinement relation $\mathcal{R} \subseteq M \times M$ by $(\pi, \pi') \in \mathcal{R}$ iff π' is an agent-restriction of π.

Then the robustness for the property *"part of the result produced by a CSA belongs to a given set S from $PsREG$"* against agent-restriction, can be formulated as the robustness of $\mathcal{M}$ for Φ against $\mathcal{R}$, where

$$\Phi = \{EF(s \wedge AX\bot) \mid s \in S\}.$$

Indeed, $\mathcal{M}, \pi \models EF(s \wedge AX\bot)$ for some $s \in S$ iff there exists an evolution $e \in \mathcal{E}^{asyn}(\pi)$ such that $\mathcal{M}, e \models F(s \wedge AX\bot)$, iff $e = \langle \pi_0 = \pi, \pi_1, \cdots \pi_i, \cdots \rangle$ and there exists $j \geq 0$ such that $\mathcal{M}, \pi_j \models (s \wedge AX\bot)$, i.e., $\mathcal{M}, \pi_j \models AX\bot$ and $\mathcal{M}, \pi_j \models s$. However, $\mathcal{M}, \pi_j \models AX\bot$ means that for any evolution $e' = \langle \pi_j = \pi'_0, \pi'_1 \ldots \rangle$ we have $\mathcal{M}, \pi'_1 \models \bot$, which is not possible. Concluding, we have that $\mathcal{M}, \pi_j \models AX\bot$ specifies the fact that π_j is a halted CSA. Note also that $\mathcal{M}, \pi_j \models s$ is is true iff $s \in Ps_T^{asyn}(\pi)$. Summing up, $\mathcal{M}, \pi \models EF(s \wedge AX\bot)$ expresses the fact that there exists a halting evolution of π where the halting configuration satisfies s.

Consequently, if π' is an agent-restriction of π, then the fact that they will eventually produce the same vectors from S can be expressed as:

for any $s \in S$, $\mathcal{M}, \pi \models EF(s \wedge AX\bot)$ iff $\mathcal{M}, \pi' \models EF(s \wedge AX\bot)$, or equivalently, $\mathcal{M}$ is robust for Φ against agent-restriction.

The problem of checking whether or not "part of the result produced by a CSA belongs to a given set S from $PsREG$" is called *S-reachability problem of $\mathcal{M}$ against agent-restriction.*

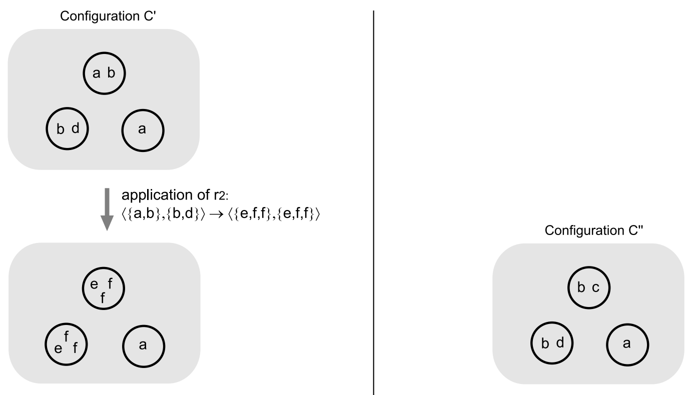
Example 6.1 *Consider π as given in Example 3.2. Suppose we fix the core set $\{(1,2)\}$. π is robust when an occurrence of agent bc is deleted from its initial configuration. In fact, if we consider $\pi' = (A, T, C', R)$ with $C' = \{(ab, 1), (bd, 1), (a, 1)\}$ we have that $Ps_T^{asyn}(\pi') = \{(1,2), (0,0)\}$, which still contains the defined core set. The single computation of π' is represented in Figure 3(a).*

On the other hand, π is not robust when an occurrence of ab is deleted from its initial configuration. In fact, if we consider $\pi'' = (A, T, C'', R)$ with $C'' = \{(bd, 1), (bc, 1), (a, 1)\}$ we have that $Ps_T^{asyn}(\pi'') = \{\overline{0}\}$, which does not contain the core set. The single computation of π'', i.e., the one halting in the initial configuration (no rule can be applied), is represented in Figure 3(b).

It is interesting to investigate whether or not the robustness in the case of S-reachability is decidable and, if possible, to evaluate the computational cost. Hereafter we present an undecidability result.

Theorem 6.1 *For an arbitrary set S of vectors from $PsREG_T$ and an arbitrary model $\mathcal{M}$ over S, the S-reachability problem of $\mathcal{M}$ against agent-restriction is undecidable.*

Proof Basically, we have to prove that checking whether or not $\mathcal{M}$ is robust for $\Phi = \{EF(s \wedge AX\bot) \mid s \in S\}$ against agent-restriction is undecidable. Checking for robustness of $\mathcal{M}$ for $\Phi =$



(a) π': agent bc removed from C | (b) π'': agent ab removed from C

Figure 3: The robustness and lack of robustness of (a) π' and (b) π'' from Example 6.1 when agents bc and ab, respectively, are removed from C.

$\{EF(s \wedge AX\bot) \mid s \in S\}$ against agent-restriction is equivalent, using the semantics introduced in Definition 4.3, with checking for $Ps_T^{asyn}(\pi) \cap S = Ps_T^{asyn}(\pi') \cap S$ for any $(\pi, \pi') \in \mathcal{R}$. But this is undecidable, because it is undecidable to check for $Ps_T^{asyn}(\pi) \cap S \subseteq Ps_T^{asyn}(\pi')$, as already shown in [5]. □

Informally, Theorem 6.1 says that there is no algorithm to solve S-reachability problem for an arbitrary model against agent-restriction. This result depends critically on the fact that the core set corresponds to a specific internal contents that the agents must have in the halting configurations.

In fact, when we consider *weaker* core sets the problem becomes decidable. For instance, suppose we only are interested in the *magnitude* of the agents in the halting configurations. This means that we collect, for a CSA π, the set of numbers $N^{asyn}(\pi)$. In this case the robustness problem can be rephrased in the following manner.

Consider an arbitrary set N from $NREG$. We introduce, as before, the model $\mathcal{M} = \langle M, i \rangle$ over N, where M is the smallest set of CSAs that contains the colonies we are interested in, and which is closed under $asyn$-evolutions; $i : M \to \mathcal{P}(N)$ is the interpretation function defined by $i(\pi) = N \cap N^{asyn}(\pi)$ if π is a halted CSA, and $i(\pi) = \emptyset$, else. All these can be alternatively written in terms of satisfiability relation, for $\pi \in \mathcal{M}$ and $n \in N$, as

$\mathcal{M}, \pi \models n$ iff π is halted and $n \in N^{asyn}(\pi)$.

We also define the refinement relation $\mathcal{R} \subseteq M \times M$ by $(\pi, \pi') \in \mathcal{R}$ iff π' is an agent-restriction of π.

Then, the robustness for the property *"the magnitude of the result produced by a CSA belongs to a given set N of numbers from $NREG$"* against agent-restriction, can be formulated as the robustness of $\mathcal{M}$ for Φ against $\mathcal{R}$, where

$$\Psi = \{EF(n \wedge AX\bot) \mid n \in N\}.$$

One can verify that, as shown in the case of the set Φ, Ψ specifies the intended property. We call this problem *N-magnitude problem of $\mathcal{M}$ against agent-restriction.*

Theorem 6.2 *For an arbitrary set N from $NREG$ and an arbitrary model $\mathcal{M}$ over N, the N-magnitude problem of $\mathcal{M}$ against agent-restriction is decidable.*

Proof Basically, we have to prove that checking whether or not $\mathcal{M}$ is robust for $\Psi = \{EF(n \wedge AX\bot) \mid n \in N\}$ against agent-restriction is decidable. But checking for robustness of $\mathcal{M}$ for $\Psi = \{EF(n \wedge AX\bot) \mid n \in N\}$ against agent-restriction is equivalent with checking whether or not $N^{asyn}(\pi) \cap S = N^{asyn}(\pi') \cap S$ for arbitrary $(\pi, \pi') \in \mathcal{R}$. In [5] it is shown that $N^{asyn}(\pi) \cap S \subseteq N^{asyn}(\pi')$ is decidable and, using similar arguments we can derive the decidability of checking whether or not $N^{asyn}(\pi) \cap S = N^{asyn}(\pi') \cap S$ is true. Hence, the N-magnitude problem is decidable. □

Informally, the above result says that it is possible to check whether or not a CSA is robust against arbitrary deletion of agents, subject to the core result being defined in terms of magnitudes of agents.

6.2 Rule-restrictions, Rule-modifications and Mutants

Similarly with the case of agent-restriction, we analyze the case where in the refinement relation the second colony is obtained from the first colony by modifying or removing some of the rules. This type of refinement can be seen as describing a "mutant" of a CSA. As in the previous sections, we will stress on some decidability results obtained for particular class of properties.

Consider a CSA $\pi = (A, T, C, R)$. The CSA $\pi' = (A, T, C, R')$ is a *mutant*[5] of π. Obviously a colony has several mutants. A refinement relation that associates a colony and one of its possible mutants is called *rule-modification*. A special case of mutants are the *rule-restriction mutants*, which are mutants obtained by only removing some of the rules (possibly all) of a colony. This represents the case when some of the rules do not work, as if, once again, they are absent from the colony.

The refinement relation defined on a model by the rule-modification/restriction introduces specific cases of robustness. As done in the previous subsection, we can study whether or not a moled $\mathcal{M}$ is robust for properties such as *"part of the result produced by a CSA belongs to a given set S from $PsREG$"* (S-reachability problem of $\mathcal{M}$) or *"the magnitude of the agents in the halting configurations belong to a given subset N of $NREG$"* (N-magnitude problem of $\mathcal{M}$), against rule-modification/restrictions. As before, the S-reachability problem of $\mathcal{M}$ is equivalent with checking for the robustness of $\mathcal{M}$ for $\Phi = \{EF(s \wedge AX\bot) \mid s \in S\}$, while the N-magnitude problem is equivalent with checking for the robustness of $\mathcal{M}$ for $\Psi = \{EF(n \wedge AX\bot) \mid n \in N\}$.

Theorem 6.3 *For an arbitrary set S from $PsREG_T$ and an arbitrary model $\mathcal{M}$ over S, the S-reachability problem of $\mathcal{M}$ is undecidable against rule-restriction.*

[5]Note that there is no relation predefined between the set of rules R of π and the set of rules R' of its mutant π'. A particular case that is studied in this subsection is the case of rule-restriction where $R' \subset R$

Proof In [5] it has been shown that it is undecidable whether or not, for an arbitrary CSA, π, with arbitrary terminal alphabet T, arbitrary rule-restriction π' of π, and arbitrary set S from $PsREG_T$, we have $Ps_T^{asyn}(\pi) \cap S \subseteq Ps_T^{asyn}(\pi')$. Moreover, to check whether or not $Ps_T^{asyn}(\pi) \cap S = Ps_T^{asyn}(\pi') \cap S$ is equivalent, using the semantics of $\mathcal{L}_S$ as introduced in Definition 4.3, with checking the robustness of $\mathcal{M}$ for $\Phi = \{EF(s \wedge AX\bot) \mid s \in S\}$ against rule-restriction. Thus, the result of the theorem follows from the fact that $Ps_T^{asyn}(\pi) \cap S \subseteq Ps_T^{asyn}(\pi')$ is undecidable. $\square$

The previous result says that given an arbitrary set $S \subseteq PsREG_T$, and an arbitrary relation that associates to a colony one of its mutants, we cannot provide an algorithm to decide the S-reachability problem. The result can be generalized to the case of refinements defined by rule-modifications.

Corollary 6.1 *For an arbitrary set S from $PsREG_T$ and an arbitrary model $\mathcal{M}$ over S, S-reachability problem of $\mathcal{M}$ is undecidable against rule-modification.*

Proof The result derives from Theorem 6.3 due to the fact that rule-restriction is a particular case of rule-modification and the S-reachability problem is already undecidable for rule-restriction. $\square$

However, using the same ideas as those in Theorem 6.2 we can get the following result.

Theorem 6.4 *For an arbitrary set N from $NREG$ and an arbitrary model $\mathcal{M}$ over N, the N-magnitude problem of $\mathcal{M}$ against rule-restriction is decidable.*

Proof It is known to be decidable whether or not, for an arbitrary CSA, π, arbitrary rule restriction π' of π and arbitrary set S from $NREG$, $N(\pi) \cap S \subseteq N(\pi')$, [5]. This implies the decidability of whether or not $N(\pi) \cap S = N(\pi') \cap S$. However, $N(\pi) \cap S = N(\pi') \cap S$ iff $\mathcal{M}$ is robust for $\Psi = \{EF(n \wedge AX\bot) \mid n \in N\}$ against rule-restriction. Then the theorem follows. $\square$

In other words, there exists an algorithm to solve the N-magnitude problem of a model $\mathcal{M}$ against rule-restriction.

6.3 Global Robustness, Bisimulations and Species

Usually, in biology, a species is defined by a set of characteristic properties that are shared by all the components of a species and only by them. Moreover, once an individual has been defined to be part of a species, such an individual will always belong to that species. Inspired by this description, we introduce in this section the concept of *species of CSAs*, and we use for this the notion of global robustness defined for the characteristic properties of a species.

We first recall the definition of global robustness.

Definition 6.1 (Global Robustness) *Let S be a set of atomic properties, $\mathcal{M} = \langle M, i\rangle$ a model over S, and $\mathcal{R} \subseteq M \times M$ a refinement on M. $\mathcal{M}$ is* globally robust against $\mathcal{R}$ *if it is robust for $\mathcal{L}_S$ against $\mathcal{R}$.*

Given a model $\mathcal{M}$ over a set S of atomic properties and a refinement $\mathcal{R}$ on $\mathcal{M}$, we say that $\mathcal{R}$ *defines species on* $\mathcal{M}$ when $\mathcal{M}$ is globally robust against $\mathcal{R}$.

Notice that global robustness, once established against a couple (π, π') of CSAs, it is preserved in the future, i.e., the future states of π are robust against (some) future states of π' and reverse. This observation give hints on the fact that global robustness can be described as a *bisimulation*.

Definition 6.2 (Bisimulation on CSAs) *Consider a model $\mathcal{M} = \langle M, i\rangle$ over a set S of atomic properties. The S-*bisimulation relation *on $\mathcal{M}$ is the smallest equivalence relation $\sim_S \subseteq M \times M$ such that if $\pi_1 \sim_S \pi_2$, then*

- *$i(\pi_1) = i(\pi_2)$;*
- *if $\pi_1 \mapsto \pi_1'$, then there exists $\pi_2' \in M$ such that $\pi_2 \mapsto \pi_2'$ and $\pi_1' \sim_S \pi_2'$;*
- *if $\pi_2 \mapsto \pi_2'$, then there exists $\pi_1' \in M$ such that $\pi_1 \mapsto \pi_1'$ and $\pi_1' \sim_S \pi_2'$;*

Now we can prove that global robustness for $\mathcal{L}_S$ can be characterized as the S-bisimulation on $\mathcal{M}$. In this way, we provide a "dynamic" description of robustness.

Theorem 6.5 (Characterization of Global Robustness) *Consider a set S of atomic properties, a model $\mathcal{M} = \langle M, i\rangle$ over S and a refinement $\mathcal{R}$ on $\mathcal{M}$. $\mathcal{M}$ is globally robust against $\mathcal{R}$ iff $\mathcal{R}$ is the S-bisimulation relation on $\mathcal{M}$.*

Proof We have to prove the following assertions.

1. The relation $\sim \subseteq M \times M$ defined by $\pi \sim \pi'$ iff for any $\phi \in \mathcal{L}_S$, $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$ is an S-bisimulation.
2. If $\sim \subseteq M \times M$ is an S-bisimulation and $\pi \sim \pi'$ then for any $\phi \in \mathcal{L}_S$ we have $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$.

(1) Suppose that $\pi \sim \pi'$.

We prove that $i(\pi) = i(\pi')$. We have $s \in i(\pi)$ iff $\mathcal{M}, \pi \models s$, iff $\mathcal{M}, \pi' \models s$, iff $s \in i(\pi')$. Hence, $s \in i(\pi)$ iff $s \in i(\pi')$, i.e., $i(\pi) = i(\pi')$.

We prove that if $\pi \mapsto \pi_1$, then $\pi' \mapsto \pi_1'$ and $\pi_1 \sim \pi_1'$. If $\pi \mapsto \pi_1$, then $\mathcal{M}, \pi \models EXT$, that implies $\mathcal{M}, \pi' \models EXT$, i.e., there exists π_1' such that $\pi \mapsto \pi_1'$.

For proving that $\pi_1 \sim \pi_1'$, consider all π_1' such that $\pi' \mapsto \pi_1'$ (the successors of π'). As π' has a finite set of rules, we have a finite number of successors of π' (we assumed that there are k successors). We have to prove that one of these, π_k', has the property that for any $\phi \in \mathcal{L}_S$ we have $\mathcal{M}, \pi_1 \models \phi$ iff $\mathcal{M}, \pi_k' \models \phi$.

Let π_1' be a successor of π', i.e., $\pi' \mapsto \pi_1'$. And suppose that there exists a formula $\phi_1 \in \mathcal{L}_S$ such that $\mathcal{M}, \pi_1 \models \phi_1$ and $\mathcal{M}, \pi_1' \not\models \phi_1$. From $\mathcal{M}, \pi_1 \models \phi_1$, we derive that $\mathcal{M}, \pi \models EX\phi_1$, that is equivalent with $\mathcal{M}, \pi' \models EX\phi_1$. Consequently, there exists a successor π_2' of π' that satisfies ϕ_1. Obviously, $\pi_1' \neq \pi_2'$.

Suppose that there exists a formula $\phi_2 \in \mathcal{L}_S$ such that $\mathcal{M}, \pi_1 \models \phi_2$ and $\mathcal{M}, \pi'_2 \not\models \phi_2$. Then $\mathcal{M}, \pi \models EX(\phi_1 \wedge \phi_2)$, implying that $\mathcal{M}, \pi' \models EX(\phi_1 \wedge \phi_2)$, i.e., there exists one successor π'_3 of π' that satisfies ϕ_1 and ϕ_2. Consequently, $\pi_3 \notin \{\pi'_1, \pi'_2\}$.

This argument can be repeated for maximum k steps such that, eventually, we find a successor π'_k of π' with the property that for any $\phi \in \mathcal{L}_S$, $\mathcal{M}, \pi_1 \models \phi$ iff $\mathcal{M}, \pi'_k \models \phi$.

Similarly, can be proved that if $\pi' \mapsto \pi'_1$, then $\pi \mapsto \pi_1$ such that $\pi_1 \sim \pi'_1$.

It is trivial to verify that $\sim$ is, indeed, the equivalence relation that defines the S-bisimulation.

(2) We prove, by induction on the complexity of a formula in $\mathcal{L}_S$ (the complexity of a formula is given in the standard way, by the number of logical operators that appear in the syntax of the formula), that if $\pi \sim \pi'$, then for any $\phi \in \mathcal{L}_S$ we have $\mathcal{M}, \pi \models \phi$ iff $\mathcal{M}, \pi' \models \phi$.

(i) The case $\phi = s \in S$. $\mathcal{M}, \pi \models s$ iff $s \in i(\pi)$. Because $\sim$ is a bisimulation and $\pi \sim \pi'$, we have $i(\pi) = i(\pi')$. Hence, $s \in i(\pi')$, which is equivalent with $\mathcal{M}, \pi' \models s$.

(ii) The case $\phi = \neg\psi$. $\mathcal{M}, \pi \models \neg\psi$ iff $\mathcal{M}, \pi \not\models \psi$. Suppose that $\mathcal{M}, \pi' \models \psi$. Then, the inductive hypothesis gives $\mathcal{M}, \pi \models \psi$ that is not possible. Consequently, $\mathcal{M}, \pi' \not\models \psi$, i.e., $\mathcal{M}, \pi' \models \neg\psi$.

(iii) The case $\phi = \phi_1 \wedge \phi_2$. $\mathcal{M}, \pi \models \phi_1 \wedge \phi_2$ iff $\mathcal{M}, \pi \models \phi_1$ and $\mathcal{M}, \pi \models \phi_2$. Using the inductive hypothesis we derive $\mathcal{M}, \pi' \models \phi_1$ and $\mathcal{M}, \pi' \models \phi_2$, that imply $\mathcal{M}, \pi' \models \phi_1 \wedge \phi_2$.

(iv) The case $\phi = E\phi_1 U \phi_2$. $\mathcal{M}, \pi \models E\phi_1 U \phi_2$ iff there exists $e = \langle \pi, \pi_1, \cdots, \pi_k, \cdots \rangle \in \mathcal{E}^{asyn}(\pi)$ such that there exists $i \geq 1$ with $\mathcal{M}, \pi_i \models \phi_2$ and for all $j \leq i$, $\mathcal{M}, \pi_j \models \phi_1$. Because $\sim$ is a bisimulation and $\pi \sim \pi'$, there exists $e' = \langle \pi', \pi'_1, \cdots, \pi'_k, \cdots \rangle \in \mathcal{E}^{asyn}(\pi')$ such that for each $l \geq 1$, $\pi_l \sim \pi'_l$. Further, using the inductive hypothesis, we obtain that there exists $i \geq 1$ with $\mathcal{M}, \pi'_i \models \phi_2$ and for all $j \leq i$, $\mathcal{M}, \pi'_j \models \phi_1$, i.e. $\mathcal{M}, \pi' \models E\phi_1 U \phi_2$. □

Corollary 6.2 *Let S be a set of atomic properties and $\mathcal{M}$ a model over S. A refinement $\mathcal{R}$ defines species on $\mathcal{M}$ iff $\mathcal{R}$ is an S-bisimulation.*

Proof Direct consequence of Theorem 6.5. □

This result shows that checking whether or not a refinement defines species over a model is equivalent with checking for bisimulation conditions.

Corollary 6.3 *Let S be a set of atomic properties and $\mathcal{M}$ a model over S. If a refinement $\mathcal{R}$ defines species on $\mathcal{M}$, then $\mathcal{R}$ is an equivalence relation.*

Proof Consequence of the fact that bisimulation is an equivalence. □

The fact that $\mathcal{R}$ defines species on $\mathcal{M}$ only if $\mathcal{R}$ is the S-bisimulation relation, implies that the species are actually equivalence classes with respect to relation $\mathcal{R}$. This allows us to propose the following definition of species.

Definition 6.3 (Species) *Let S be a set of atomic properties, $\mathcal{M} = \langle M, i \rangle$ a model over S and $\mathcal{R} \subseteq M \times M$ a refinement that defines species on $\mathcal{M}$. A* species *is an equivalence class defined by $\mathcal{R}$ over M.*

7 Conclusions and Prospects

In this paper we have analyzed the robustness of Colonies of Synchronizing Agents, a computability model introduced in [6, 5]. We have defined a way of encoding properties of CSAs by means of a temporal logic. Robustness of a model is introduced for temporal formulas and analyzed against refinement relations between colonies. Examples of refinements are deleting/modifying agents of a colony or deleting/modifying rules of a colony. By modifying rules of the colonies we obtain mutants of the colony. A special case of robustness is the global robustness, which is preserved by the $asyn$-evolutions of the colonies. The global robustness have been used to introduce the concept of species of CSAs, and it has been proved to be equivalent with the notion of bisimulation on CSAs.

Several enhancements to this approach are already in prospect, especially in the extension of the investigated model. Primary among these is the addition of *space* to the colony. Precisely, each agent will have a triple of co-ordinates corresponding to its position in Euclidean space and the rules will be similarly endowed with the ability to modify an agent's position. A further extension of this idea is to give each agent an *orientation*, i.e. a rotation relative to the spatial axes, which may also be modified by the application of rules.

The idea is to make the application of a rule dependent on either an absolute position (thus directly simulating a chemical gradient) or on the relative distance between agents in the case of synchronization. Moreover, in the case of the application of a synchronization rule, the ensuing translation and rotation of the two agents may be defined *relative to each other*. In this way it will be possible to simulate reaction-diffusion effects, movement and local environments.

Some additional biologically-inspired primitives are also planned, such as agent *division* (one agent becomes two) and agent *death* (deletion from the colony). These primitives can simulate, for example, the effects of mitosis, apoptosis and morphogenesis. In combination with the existing primitives, it will be possible (and is planned) to model, for example, many aspects of the complex multi-scale behaviour of the immune system.

With the addition of the features just mentioned, it will also be interesting to extend the investigation and proofs given above to identify further classes of CSAs demonstrating robustness and having decidable properties. It is hoped that this approach will then provide insight in challenging areas of systems and evolutionary biology.

References

[1] M. Ben-Ari, A. Pnueli, Z. Manna, The Temporal Logic of Branching Time, *Acta Inf.*, 20, 1983.

[2] F. Bernardini, R. Brijder, G. Rozenberg, C. Zandron, Multiset-Based Self-Assembly of Graphs, *Fundamenta Informaticae*, 75, 2007.

[3] F. Bernardini, M. Gheorghe, Population P Systems, *Journal of Universal Computer Science*, 10, 5, 2004.

[4] C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa eds., *Multiset Processing: Mathematical, Computer Science, and Molecular Computing Point of View*, LNCS 2235, Springer-Verlag, 2001.

[5] M. Cavaliere, R. Mardare, S. Sedwards, A Multiset-Based Model of Synchronizing Agents: Computability and Robustness, Theoretical Computer Science, To Appear.

[6] M. Cavaliere, R. Mardare, S. Sedwards, Colonies of Synchronizing Agents, Technical Report CoSBi 11/2007. Available at www.cosbi.eu/Rpty_Tech.php.

[7] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

[8] A. Ilachinski, *Cellular Automata - A Discrete Universe*, World Scientific Publishing, 2001.

[9] R. Freund, Gh. Păun, O.H. Ibarra, H.-C.Yen, Matrix Languages, Register Machines, Vector Addition Systems, *Proc. Third Brainstorming on Membrane Computing*, Sevilla, 2005, RGCN Report 01/2005. Available at *www.gcn.us.es*

[10] S. Greibach, Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7, 3, 1978.

[11] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[12] J. Kelemen, A. Kelemenová, Gh. Păun, Preview of P Colonies - A Biochemically Inspired Computing Model, *Proceedings of Workshop on Artificial Chemistry*, ALIFE9, Boston, USA, 2004.

[13] J. Kelemen, Gh. Păun, Robustness of Decentralized Knowledge Systems: A Grammar-Theoretic Point of View, *Journal Expt. Theor. Artificial Intelligence*, 12, 2000.

[14] C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodrǵuez-Patón, Tissue P Systems, *Theoretical Computer Science*, 296, 2, 2003.

[15] J. Mata, M. Cohn, Cellular Automata-Based Modelling Program: Synthetic Immune Systems, *Immunol Rev*, 207, 2007.

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

[17] Gh. Păun, Introduction to Membrane Computing, in *Applications of Membrane Computing*, G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez, eds., Springer-Verlag, Berlin, 2006.

[18] G. Rozenberg, A. Salomaa, eds., in *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.

[19] J. Van Benthem, Temporal logic, in *Handbook of Logic in Artificial Intelligence and Logic Programming: Epistemic and Temporal reasoning*, Oxford University Press, 1995.

[20] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[21] http://nusmv.irst.itc.it/

[22] http://psystems.disco.unimib.it