A generic model based on automata for multi-agent systems

Davy Capera [a] Jean Fanchon [b] Jean-Pierre Georgé [a] Valérie Camps [a]

[a] *Institut de Recherche en Informatique de Toulouse IRIT, 118, route de Narbonne F-31062 Toulouse Cedex*

[b] *Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS, 7, avenue du Colonel Roche F-31077 Toulouse Cedex*

Abstract

The objective of this article is the presentation of a model based upon a sort of extended automata product, dedicated to multi-agent systems. We detail the definition and notation enabling the description of a large range of multi-agent systems, as well as the different concepts and notions usually present in this domain. This work is part of a project called MASE whose objectives are to develop a model for adaptive multi-agent systems based upon cooperative self-organisation with emergent functionality, so as to allow their study in a formal way. We are particularly interested by automated proofs aspects concerning properties stemming from the dynamic of this kind of systems, more specifically concerning the question of the convergence.

1 Introduction

Multi-agents systems [22] [24] are made up of numerous autonomous entities, called agents. Usually, these agents have to acheive an individual objective while they are interacting each others in a common environment. Obviously, designers of multi-agent systems want to obtain specific global behaviours from these local activities. This global phenomenum is characterised as emergent if the way the system exhibits the desired properties is not coded in the agents and, as a consequence, if it only exists thanks a continuous collective adaptation process. Emergent phenomena correspond to spontaneous appearance of a global relevant behaviour from local interactions only, when the local rules of interaction are, a priori, independant and are executed whithout any reference to the global activity. Our objective is to study such emergent phenomena in multi-agent systems by modeling them and their environment using a general model, then by developping this general model to focus on the study of multi-agent systems with emergent fonctionnalities and based on cooperative self-organisation. Facing this objective, two main questions deserve to be raised:

- Is it possible to define properties the agents must have in order to obtain a specific global property?
- What (and how) can we deduce at the global level from the lower level known properties?

Several specific models exist covering a wide range of multi-agent systems: process algebra [20] [5], cellular or communicating automata, synchronous or asynchronous, graph relabeling or graph rewriting systems etc. We propose a general model to deal with numerous types of multi-agent systems. This model can be instanciated to fit with various type of system according to their characteristics. Besides the possibility to allow several kinds of communication and synchronisation, this model provide a way to characterise, classify and compare different forms of emergence, cooperation, non cooperation etc. Considering that we are interested in emergent properties, our model is defined in a bottom-up way, that is to say we made up the model from the definition of agents behaviour and their interactions with other agents and with the environment. Thus, the model is not based on a global transition system but it is made up from the definition of local transitions from which a global transition system is obtained.

In this article, the model we aim to present is based upon a product of extended automata [2]. Elementary components describe agents using extended automata (by adding variables) and enhanced by knowledge and neighbourhood of agents. The set of global configurations (or global states) is composed by arrays of these local configurations. The transition relation models the system dynamics: transitions are defined by pre and post-conditions that describe the transformation from a starting conf to a another. A finite set of such rules generates all the reachable transitions and configurations. Thus, the behaviour of the system is described by a transition system ST = (states, actions, transitions, initial and/or endding states) [2].

Elementary components of the model are presented in the section 2. The third section is dedicated to the definition of the transition system which describe the system operational semantics thanks writing of rules that generate the transition relation. The way this model can be applied to multi-agent systems is shown in the section 4. We emphasize the interactions modelling and the presentation of three agent models (generic agent, BDI agent, cooperative agent). The use of the model for a simulation of foraging ants is presented in the section 5. Then, we discuss in section 6 the model suitability relating to its application for multi-agent modelling and from the point of view of concepts expressiveness capabilities.

2 Elementary components

Modelling of multi-agent systems and their behaviours begins by the definition of a set I of agent indentifiers. This set is defined by a integer range $I = [n] = \{1, ...n\}$. Development principles of the model are based on the following choices:

1. Each agent is represented by an automata. While an agent is processing a behaviour, it is in a particular state.
2. Agent states are extended by the knowledge the agent has about otherselves and of itself. Moreover, a neighbourhood relation is associated to each state. This extended states are the local configurations.
3. The global behaviour of the system is defined from local components: a global configuration is an array of local configurations and dynamics of the system is described by transitions that must be compatible with local transitions of each agent.

2.1 Agents and automata

From the set of identifiers I, a family of automata $A = \{A_i = (Q_i, \Sigma_i, \rightarrow_i), i \in I\}$ indexed by I represents the agents. For each automata A_i, are defined:

1. Q_i, a set of elementary states with which are associated a set of variables Var_i.
2. Σ_i, a set of elementary actions (including the nil action which is noted $*$).
3. $\rightarrow_i \subseteq Q_i \times \Sigma_i \times Q_i$, a transition relation noted $q \xrightarrow{a}_i q'$ expressing the switching from the state q to q' by the elementary action a. For each state, a nil transition is defined: $q \xrightarrow{*}_i q$.

Adding a set of variables to states is a way to reduce their number and, as a consequence, to simplify their writing. For instance, definition of a state $Wait$ with a ranged waiting time $[t_{min}, t_{max}]$ can be noted $Wait[t]$, that avoids multiple states definitions like $WaitT_{Min}$, $WaitT_{Min+1}$, ..., $WaitT_{Max}$.

2.2 Elementary Configurations

For each agent, the set of its elementary (or local) configurations is $S_i \subseteq Q_i \times \Pi_{j \in I} R_i(j) \times 2^I$, in which elements for a given agent i are triples $s_i = (q_i, r_i[], v_i)$, composed as following:

1. $q_i \in Q_i$ is an elementary state;

2. $r_i[]$ is an array of local views, where for each $j \in I$ we have $r_i[j] \in R_i(j)$, the set $R_i(j)$ with the empty view $\perp \in R_i(j)$ an agent i possesses about an agent j;

3. $v_i \subseteq I$ is a set of identifiers: neighbours of agent i in that configuration.

For each agent (or type of agent), it is necessary to define what it can know and believe about other agents. For instance, let's consider a set of situated homogeneous agents that have a set of states q_i to which are associated coordinates of their positions ($q[x, y]$). A way to store data about positions could consist in adding yields $coord_x$ and $coord_y$ into local views of agents. Thus, $r_i[j].coord_x$ contains the x value of the agent j position. $r_i[i].coord_x$ and $r_i[i].coord_y$ express the knowledge of agent i about its own position. That knowledge could be inaccurate or false - thus we called them beliefs -, i.e. $r_i[i].coord_x \neq q_i[x]$ for instance.

Restriction to a subset of agents Local configuration restriction $s_i = (q_i, r_i[], v_i)$ to a subset $J \subseteq I$ such that $i \in J$ (noted s_i/J) is the triple $s_i/J = (q_i, r_i/J[], v_i \cap J)$ in which the array of local views $r_i/J[j]$ is only defined for agents $j \in J$ with $r_i/J[j] = r_i[j]$. That notation is useful to define application conditions of the rules presented in section 2.3.

2.3 System and global configurations

The set of global configurations (or global states) is the product $\Omega = \Pi_{i \in I} S_i$ of local configurations sets. An element $\overrightarrow{s} \in \Omega$ can also be written $\overrightarrow{s} = (s_1, s_2....s_n)$ with $|I| = n$.

Bringing together the first three yields of local configurations $s_i = (q_i, r_i[], v_i)$, $\overrightarrow{s} \in \Omega$ can also be written $\overrightarrow{s} = (\overrightarrow{q}, \overrightarrow{r}, v)$ with $\overrightarrow{q} = (q_1, q_2....q_n)$, $\overrightarrow{r} = (r_1[], r_2[]....r_n[])$, and $v : I \longmapsto 2^I$ links each identifier with the set of its neighbours.

Thus, global configurations can be defined by $\Omega = \Pi_{i \in I} Q_i \times (\Pi_{i \in I} \Pi_{j \neq i} R_j) \times V$ in which $V = \{I \longmapsto 2^I\}$ is the set of neighbourhood relations. It is to be noted that an application $v : I \longmapsto 2^I$ defines a simple oriented graph $G_v = (I, v)$ or G if there is no ambiguity.

Then, the set Ω represents all possible states of the system relating to definitions of agents which are made it up.

Global transitions To express multi-agent system dynamics, we need to define transitions that enable to move from a global configuration to another. Thus, we define Act the set of global actions $Act = \cup_{J \subseteq I} \prod_J \Sigma_i$. An element of Act is an array of elementary actions $(a_j)_{j \in J}$ for a subset $J \subseteq I$.

The global transition relation is:

$$\longrightarrow \subseteq \Omega \times Act \times \Omega$$

A triple of this relation is noted:

$$(s_1, s_2....s_n) \xrightarrow{(a_j)_{j \in J}} (s'_1, s'_2....s'_n)$$

in which $J \subseteq I$ is called the subset of active agents (those that involve in the transition).

Such a transition can also be written:

$$(\overrightarrow{q}, \overrightarrow{r}, v) \xrightarrow{(a_j)_{j \in J}} (\overrightarrow{q'}, \overrightarrow{r'}, v')$$

or:

$$\overrightarrow{s} \xrightarrow{a} \overrightarrow{s'}$$

where $a = (a_i)_{i \in J}$.

The underlying graph of a transition is the subgraph from G_v generated by vertices of J, $G_v/J = (J, v/J)$.

Consistency between global transitions and elementary ones. We want to restrict global transitions to the only ones which are consistent with the elementary transitions at the agent level. Thus, for a global transition $(s_1, s_2....s_n) \xrightarrow{(a_j)_{j \in J}} (s'_1, s'_2....s'_n)$, we require that for all $i \in J$, if $s_i = (q_i, r_i[], v_i)$ and $s'_i = (q'_i, r'_i[], v'_i)$, then there is a local transition $q_i \xrightarrow{a_i}_i q'_i$, and $s_i = s'_i$ if $i \notin J$.

Let $\Pi_{i \in I} A_i$ be the automaton $\Pi_{i \in I} A_i = (\Pi_{i \in I} Q_i, Act, \longrightarrow_I)$, where the transition relation $\longrightarrow_I$ is defined by $(q_1, q_2....q_n) \xrightarrow{(a_j)_{j \in J}}_I (q'_1, q'_2....q'_n)$ if and only if for all $i \in J$, there is a local transition $q_i \xrightarrow{a_i}_i q'_i$, and $q_j = q'_j$ if $j \notin J$. That automaton is called free product of automata $(A_i)_{i \in I}$ and only contains, by definition, transitions which are consistent with elementary transitions.

Thus, the consistent condition is expressed by:

$$(\overrightarrow{q}, \overrightarrow{r}, v) \xrightarrow{(a_j)_{j\in J}} (\overrightarrow{q'}, \overrightarrow{r'}, v')$$

only if $\overrightarrow{q} \xrightarrow{(a_j)_{j\in J}}_I \overrightarrow{q'}$, i.e. the global transition relation is consistent with the transition relation of the free product.

Projection of a configuration onto a subset of agents Let $J = (i_1, i_2....i_m) \subseteq I$ be a subset of identifiers, the projection of a configuration $\overrightarrow{s} = (s_1, s_2....s_n)$ onto J is a configuration of an agent system that is noted $\overrightarrow{s}/J = (s_{i_1}/J, s_{i_2}/J....s_{i_m}/J)$ in which each elementary configuration s_{i_k}/J is the restriction of s_{i_k} to J defined in section 2.2.

If we write $\overrightarrow{s} = (\overrightarrow{q}, \overrightarrow{r}, v)$ with $\overrightarrow{q} = (q_1, q_2....q_n)$, $\overrightarrow{r} = (r_1[], r_2[]....r_n[])$, and $v : I \longmapsto 2^I$, then $\overrightarrow{s}/J$ can be written:

$$\overrightarrow{s}/J = (\overrightarrow{q}/J, \overrightarrow{r}/J, v/J)$$

where $\overrightarrow{q}/J = (q_{i_1}, q_{i_2}....q_{i_m})$, $\overrightarrow{r}/J = (r_{i_1}/J[], r_{i_2}/J[]....r_{i_m}/J[])$, and $v/J : J \longmapsto 2^J$ is such that for all $i \in J$, we have $v/J(i) = v(i) \cap J$.

3 Transition Relation

To manage the large number of configurations and transitions, which is exponential in the number of local configurations, we consider transition rules built up from predicates in order to specify the transitions firable from a given configuration. By iteration these rules generate the set of configurations and transitions reachable from a given initial one.

3.1 Transition Rules

A rule is defined by predicates specifying properties of the source configuration and/or the action to be executed and/or the target configuration. These predicates may have a fixed size or a variable size.

We note $Q = \cup_{i\in I} Q_i$, $\Sigma = \cup_{i\in I}\Sigma_i$, and the set of agent views is denoted R, with $\cup_{i,j\in I} R_i(j) = R$. The domain of values of the local configurations of a system with n agents is denoted

$$S(n) = Q \times R^n \times 2^{[n]}$$

Fixed size rules. Let k be an integer, a rule of size k is a boolean function:

$$Trans_\theta : S(k)^k \times \Sigma^k \times S(k)^k \longrightarrow \{True, False\}$$

or a pair $\theta = (Precond_\theta, Postcond_\theta)$ of functions:

$$Precond_\theta : S(k)^k \times \Sigma^k \longrightarrow \{True, False\}$$
$$Postcond_\theta : S(k)^k \times S(k)^k \longrightarrow \{True, False\}$$

Variable size rules We have to consider rules for which the number of involved agents is an integer variable. In this case, predicates $Trans_\theta$ or $(Precond_\theta, Postcond_\theta)$ may also depend on this variable.

A variable size rule is a function:

$$Trans_\theta : \cup_{k\in\aleph} S(k)^k \times \Sigma^k \times S(k)^k \longrightarrow \{True, False\}$$

or a pair $\theta = (Precond_\theta, Postcond_\theta)$ of functions:

$$Precond_\theta : \cup_{k\in\aleph} S(k)^k \times \Sigma^k \longrightarrow \{True, False\}$$
$$Postcond_\theta : \cup_{n\in\aleph} S(k)^k \times S(k)^k \longrightarrow \{True, False\}$$

Global rules We can define rules θ which apply directly to configurations $(s_1, s_2....s_n) \in \Omega$ with:

$$Precond_\theta((s_i)_{i\in I}, (a_i)_{i\in I}) = True$$

a new configuration being defined by $(s'_1, s'_2....s'_n) \in \Omega$:

$$Postcond_\theta((s_i)_{i\in I}, (s'_i)_{i\in I} = True$$

In that case we write $(s_1, s_2....s_n) \xrightarrow{(a_i)_{i\in I}}_\theta (s'_1, s'_2....s'_n)$.

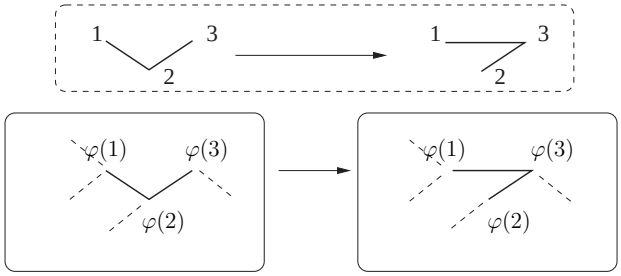

Figure 1: Example of transition induced by a rule

We thus have $\overrightarrow{s} \xrightarrow{a}_\theta \overrightarrow{s'}$ if and only if:

$$Precond_\theta(\overrightarrow{s}, a) \ = \ True$$

and

$$Postcond_\theta(\overrightarrow{s}, \overrightarrow{s'}) \ = \ True$$

3.2 Application of a rule

A rule $Trans_\theta$ applies to a configuration $\overrightarrow{s} = (s_1, s_2....s_n) \in \Omega$ yielding the configuration $\overrightarrow{s'} = (s'_1, s'_2....s'_n)$ if there is a one-to-one mapping $\varphi : [k] \longmapsto I$ with $J = \varphi([k])$ such that:

$$Trans_\theta((\overrightarrow{s}/J)), (a_i)_{i\in J}, (\overrightarrow{s}'_i/J)) = True$$

and such that furthermore for all $i \in I$, $s_{\varphi(i)}/(I-J) = s'_{\varphi(i)}/(I-J)$ and for all $i \in I-J$, $s_i = s'_i$.

We say that the rule θ applies at the occurrence φ. In that case the rule induces the global transition:

$$(s_1, s_2....s_n) \xrightarrow{(a_i)_{i\in J}}_\theta (s'_1, s'_2....s'_n)$$

A rule $(Precond_\theta, Postcond_\theta)$ applies to a configuration $\overrightarrow{s} = (s_1, s_2....s_n) \in \Omega$ if there is a one-to-one mapping $\varphi : [k] \longmapsto I$ with $J = \varphi([k])$ such that $Precond_\theta((\overrightarrow{s}/J), (a_i)_{i\in J}) = True$. The new configuration $(s'_1, s'_2....s'_n) \in \Omega$ being defined by:

$$Postcond_\theta((\overrightarrow{s}/J), (\overrightarrow{s}'/J)) = True$$

and for all $i \in I$ $s_{\varphi(i)}/(I-J) = s'_{\varphi(i)}/(I-J)$ and for all $i \in I-J$, $s_i = s'_i$ The use of a rule is depicted in 1.

In that case we note $(s_1, s_2....s_n) \xrightarrow{(a_i)_{i\in J}}_\theta (s'_1, s'_2....s'_n)$.

Types of conditions The predicates of a rule may concern one or more fields of a configuration $\overrightarrow{s} = (\overrightarrow{q}, \overrightarrow{r}, v)$, and express constraints on the local states, the knowledge or the neighbourhood relation. For instance we can define a predicate

$$Precond_\theta : 2^{[k]\times[k]} \longrightarrow \{True, False\}$$

which applies to binary relations between sets with k elements, and which is satisfied for the configuration $\overrightarrow{s} = (\overrightarrow{q}, \overrightarrow{r}, v)$ by the mapping $\varphi : [k] \longmapsto I$ if the neighbourhood relation v restricted to $\varphi([k])$ satisfies $Precond_\theta$, i.e. $Precond_\theta(v/_{\varphi([k])}) = True$.

3.3 Transition relation induced by a set of rules

Let Θ be a set of rules as defined above. We note $\longrightarrow_\Theta$ the global transition relation $\longrightarrow_\Theta \subseteq \Omega \times Act \times \Omega$ defined by $\overrightarrow{s} \xrightarrow{a}_\Theta \overrightarrow{s'}$ if and only if $\exists \theta \in \Theta : \overrightarrow{s} \xrightarrow{a}_\theta \overrightarrow{s'}$.

We write $\overrightarrow{s} \longrightarrow_\Theta \overrightarrow{s'}$ if and only if $\exists a \in Act : \overrightarrow{s} \xrightarrow{a}_\Theta \overrightarrow{s'}$.

We let $\Theta(\overrightarrow{s})$ be the set of configurations directly reachable from $\overrightarrow{s}$: $\Theta(\overrightarrow{s}) = \{\overrightarrow{s'} \in \Omega : \overrightarrow{s} \longrightarrow_\Theta \overrightarrow{s'}\}$ and $\Theta^*(\overrightarrow{s})$ the set of configurations reachable from $\overrightarrow{s}$ by a sequence of transitions.

Let $X \subseteq \Omega$, the sets of configurations $\Theta^{i+1}(X) = \Theta(\Theta^i(X)) \cup \Theta^i(X)$ for $i \in \aleph$ yield the definition of $\Theta^*(X) = \cup_{i\in\aleph}\Theta^i(X)$

The transition relation induced by the set of rules Θ, the set of agents $\{A_i = (Q_i, \Sigma_i, \rightarrow_i), i \in I\}$ and the initial configuration $\overrightarrow{s_{in}} \in \Pi_{i\in I} S_i$ is $\longrightarrow_\Theta / \Theta^*(\overrightarrow{s_{in}}) \times Act \times \Theta^*(\overrightarrow{s_{in}})$.

3.4 Transition systems

At this point we have defined I the set of agents identifiers componing the system (except environnement denoted Env corresponding to the identifier 0), $\{A_i\}$ which defines the associated family of automata , R representing the knowledge domain and Θ containing the transition rules. From these objects and an initial configuration, the transition system ST with:

$$ST = (\Omega, Act, \longrightarrow_\Theta, \overrightarrow{s}_{in})$$

constitutes the operational semantics (behaviour) of the system.

4 A multi-agent systems dedicated model

In this section, we show that the model can be used to express a large number of mecanisms and properties of multi-agent systems. Our aim is to tackle a panel as large as possible rather than to enable modelling of a specific class of systems.

4.1 The main components

For all multi-agent systems, it is necessary to describe agents behaviour. Then, tools and notations are needed to manage these behaviours relating to agents dynamics and to the whole system dynamics. As these considerations were taken into account while the model has been built, these basic definitions and mecanisms are expressed in an easy way.

A multi-agent system SMA is defined by $SMA = (I, \{A_i\}, R, \Theta, \Gamma)$ Elements I, $\{A_i\}$, R and Θ have been explained in the last sections. Predicate Γ is a consistency one that enables to define global predicate to restrict accessible global configurations or to modify conditions of rules application. Γ is mainly usefull to simplify some properties writing.

For instance, there are several ways to model physical rule such as "two solid entities can't share a common space". If we consider multi-robot system involved in a real physical environment, our model has to handle that law (collisions detection, non ubiquity etc.). If local rules are defined to avoid such situations (at automaton level), some problems occur when rules are applied on a concurrent/asynchronous way. Using Γ, we can tackle this problem by a predicate definition that restricts consistent global configurations to configurations in which there is not two (or more) robots in the same place.

4.2 Interactions

Interactions are modelled using transitions in which two (or more) entities are active. If we consider the modelling of a direct synchronous communication between two agents, that can be written using a fixed-size rule:

$$(q_i, r_i[], v_i), (q_j, r_j[], v_j) \xrightarrow{(\text{i!j, j?i})} (q'_i, r'_i[], v'_i), (q'_j, r'_j[], v'_j)$$

with the boolean function related to this specific exchange. That rule is applicable to any pair (i, j) which fits with the preconditions and postconditions of the rule.

Undirect communications (pheromone deposit, *Blackboard* etc.) are modelled using rules in which environment involves. An ant represented by an automaton i performs a pheromone deposit using a fixed-size rule like

$$(q_{Env}, r_{Env}[], v_{Env}), (q_i, r_i[], v_i) \xrightarrow{(\text{Deposit})} (q'_{Env}, r'_{Env}[], v_{Env}), (q'_i, r'_i[], v_i)$$

.

Interactions which context-dependent "operating field" are tackled thanks variable-size rules. For instance, broadcasting a message to the neighbourhood of a given agent i (whisper) can be defined using a variable-size rule whose conditions concern the neighbourhood v_i: active agents will be whose identifier j is such that $j \in v_i$.

In cellular automata, an agent modifies its state relating to its neighbours and its own state. In this case, transitions are defined by pairs $\tau = [(q_j)_{j\in J}, q']$ where $(q_j)_{j\in J}$ is a n-uplet of states and q' a new state. A given pair τ defines global transitions:

$$(s_1, ..., s_k, ...s_n) \xrightarrow{Upd_\tau(k)} (s_1, ..., s'_k, ..., s_n)$$

where $Upd_\tau(k)$ is an action supported by $J = \{k\} \cup v_k$ and generated by an agent k and its neighbours, with $s_j = (q_j, \perp, v_j)$ far all $j \in J$, and $s'_k = (q', \perp, v_k)$.

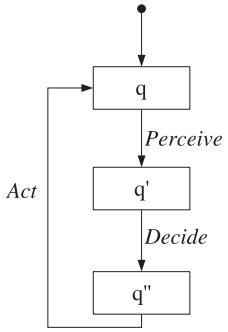

Figure 2: State graph of an agent

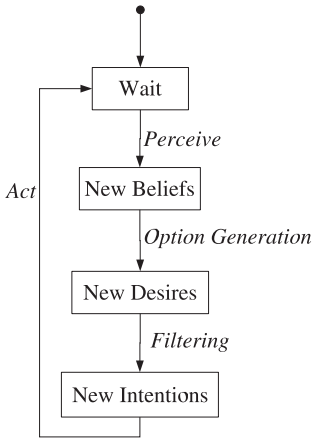

Figure 3: State grpah of a *BDI* agent

4.3 Some Classic Agent Automata

We also explored the definition of some classic automata for the agents. This provides a base to write automata for a specific system deriving from them.

Basic Agent A basic automata can easily be defined considering standard Perceive-Decide-Act life cycle. Figure 2 shows the resulting state graph.

The upper black dot and the transition toward the first state q represent the agent creation. The three following states are labelled by elementary actions *Perceive*, *Decide* and *Act*.

***BDI* Agent** Figure 3 shows a state graph which defines an agent based on the Belief-Desire-Intention model [21]. This graph is obtained by refining the decision process (*Decide*) of the basic agent. The simple (*Decide*) transition is rewritten to describe the options generation (*Option Generation*) and the intentions filtrering (*Filtering*). The first phase updates agent desires using beliefs of the state *New Beliefs* and the old desires. The second phase selects intentions relating to beliefs and tanking into account the new desires generate by the last phase. That process brings the agent in the *New Intentions* state and, then the agent can act.

Cooperative Agent Here we are interested in modelling cooperative agents as they are defined in the AMAS theory [10]. These cooperative agents have to focus on detection and solving of non cooperative situations. These non cooperative situations can be classified in three categories according to the solving actions they can trigger:

1. Regulation: agent modifies its internal parameters (representations and/or characteristics);

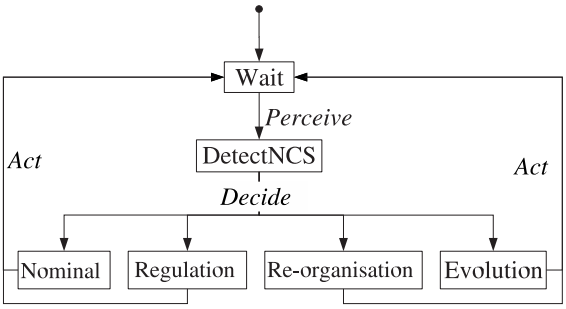

Figure 4: State graph of a cooperative agent

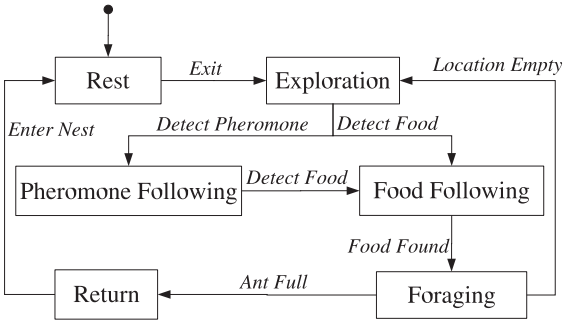

Figure 5: State graph on an ant agent

2. Re-organisation: agent creates or deletes a "work connection" it has with another agent (agents organisation topology is changed);

3. Evolution: agent adds a new agent into the system or it disappears from the system.

"Unfolding" the graph to add states related to non cooperative situations processing, we can define the generic state graph for an automaton whose general behaviour is compliant to the AMAS cooperative agents.

The resulting graph is presented in figure 4. The decision-related phase is enriched with several states and transitions. After the perception phase (action labelled *Perceive*), the agent is in the *DetectNCS* state during which it analyses non cooperative situations. If there is none such a situation, the agent processes the transition ($q_{DetectNCS} \xrightarrow{\text{Decide}} q_{Nominal}$) to execute its nominal behaviour. Else, agent forks for the state that fits with the detecting non cooperative situation (*Regulation*, *Re-organisation*, *Evolution*). For a given problem, this graph can be "unfolded" again in order to describe more precisely some states.

5 Illustrating example

We are detailing here a model of a multi-agent system simulating the foraging activity of an ant hill. This model aims not at being exhaustive but complete enough to concretely present each element of our generic model. In this example, ants are homogeneous agents which are exploring the territory in search of food zones so as to bring it back to the nest. The state graph of an agent is thus described in figure 5.

In accordance to the generic model, we define the multi-agent system as $ANT = (I, \{A_i\}, R, \Theta, \Gamma)$ with:

1. I : the set of identifying symbols for the agents (ant and environment). The ant agents are simply numbered. $I = \{1, ...n\} \cup Env$ with n the number of foraging ants and Env the environment;

2. $\{A_i\}$: the set of automata of the agents. All the ants being homogeneous, there is only one class of elementary automata to represent their behaviour, the one in figure 5. We do not present the automaton of the environment here. For $i \in 1...n$, $A_i = (Q_i, \Sigma_i, \rightarrow_i)$ with:

   (a) Q_i : the set of elementary states with associated variables representing some characteristics of an agent. The states are q_{Rest}, $q_{Exploration}$, $q_{PheromoneFollowing}$, $q_{FoodFollowing}$, $q_{Forraging}$, q_{Return}. For example, $q[PheromoneDrop]$ is an associated variable to all the states. It is indicating the quantity of pheromone dropped by the agent. Thus, when the agent changes from state $q_{Exploration}$ to state $q_{FoodFollowing}$, it drops more pheromone: $q_{FoodFollowing}[PheromoneDrop] = q_{Exploration}[PheromoneDrop] + incr$, $incr$ being the increment value. Another associated variable is the position of the agent, noted $q[Position]$;

   (b) Σ_i : the set of elementary actions containing the actions allowing to change from one elementary state to another, for example a_{Exit}, $a_{DetectFood}$,... (cf. figure5), as well as other actions in relation to the associated variables or to elementary configurations, for example $a_{IncrPheromone}$;

   (c) $\rightarrow_i$: the transition relations describing the sate graph, for example
   $$q_{Exploration} \xrightarrow{a_{DetectFood}}_i q_{FoodFollowing} \quad ;$$

3. R : the set of views of the agents, i.e. the representations of the agents concerning the other agents or the environment. For example $r_i[Env][Pheromone]$ represents the quantity of pheromone detected by the agent i in the environment Env;

4. S_i : the set of elementary configurations. With Q_i, R_i and by adding the neighbourhood of an agent v_i, we can describe an elementary configuration of an agent in the system as following: $s_i = (q_i, r_i[], v_i)$. This represents the whole life cycle of an agent in the system.

5. Θ : the set of transitions rules enabling the evolution of the elementary configurations. For example, to represent the detection and following of food, we note it:
   $$(q_{Exploration}, r_i[], v_i) \xrightarrow{(a_{DetectFood})} (q_{FoodFollowing}, r_i[], v_i)$$
   $Precond : r_i[Env][Food] = true$
   $Postcond : q_{FoodFollowing}[PheromoneDrop] = q_{Exploration}[PheromoneDrop] + incr$;

6. Γ : the coherence predicate. If we want to prevent two agents from being at the same coordinates at the same time, we can note it:
   $$\Gamma = (q_i[Position] \neq q_j[Position]).$$

6 Concepts and notions

There are numerous concepts and notions used in the domain of Multi-Agent Systems that we can not explore exhaustively in this paper. Nevertheless, we can provide some elements (sometimes in an informal way) concerning the expression of some important ones among them.

Environment. We can define a model of the environment [23] with a particular automaton of the systems (typically, the automaton A_0 also noted Env by commodity). For certain applications in which the environment has to be distributed, nothing prevents from translating this distribution by associating different automata to the different "parts" of the environment. The accessibility of these "parts" by the agents are given by the neighbourhoods of the agents. The dynamic nature of the environment can be identified by the existence of transitions in which **only** the agent corresponding to the environment is modified (or rules for which $Trans_\theta$ concerns exclusively local configurations of Env).

Autonomy. The question of autonomy [7] is a delicate one to treat. Indeed, to judge that a system (or an agent) is autonomous or not can be strongly subjective. The autonomy of decision of an agent could be translated by the fact that no elementary transition labelled *Decide* (after matching the model with the *simple agent* model type) is used by rules with conditions concerning local configurations other than the considered agent.

Reorganisation, self-organisation. Reorganisation processes are translated in our model by the application of transitions modifying the neighbourhoods of the agents. The analysis of the decision processes leading to modifications of the neighbourhoods should allow to distinguish self-organising systems from others.

Cognitive agent, reactive agent. Both of these agents types can be described in the model. Being able to identify that the automaton associated to an agent describes a reactive or cognitive (or deliberative) architecture could be achieved by defining abstraction operations (or "folding") to match the state graph of an automaton to state graphs typical of these architectures.

Cooperation and functional adequacy. Cooperation in the scope of adaptive multi-agent systems [6] will be a major centre of interests in our work. This study will need to associate an evaluation function of cooperation (or non-cooperation) to the local configurations (based upon the representations $r_i[]$) to define the notion of a cooperative agent. To define the notion of a system being in an internal cooperative state, the association will be at the level of the multi-agent system (at the level of the set of local configuration associated to the agents, the environment being excluded). The notion of functional adequacy of a multi-agent system should be measurable using an evaluation of the global configurations (they contain the environment and the multi-agent system, thus also representing the interaction of the multi-agent system and the environment). The definition of a cooperative system (and not only the definition of an internally cooperative system) will have to use a projection on the sole local configurations having the environment in their neighbourhood and an evaluation of the cooperation of these local configurations relative to the environment (evaluation of the "frontier" between the multi-agent system and the environment).

Emergence. Studies related to Emergence can focus on the detection of specific attributes of the states of the system (global configurations) and on the analysis of the dynamics responsible for the apparition of these phenomena by studying the behaviours (a sequence of transitions) producing them. The definition of Emergence for artificial systems as given in [15] could also provide means to distringuish emergent systems from those who are not.

Reliability, robustness, convergence. These notions can be defined by properties related to the accessibility of specific sets of configurations and the sequence of transitions leading to them. The use of modal and temporal logics in this framework would allow the use of automatic proof tools.

Interoperability, adaptivity. We can define such systems with the model but it seems unlikely that we will be able to formalise these properties.

Openness. The current model does not allow the expression of appearance or disappearance of agents in a strictly true manner. Nevertheless, by adding an *Out* state to the graph of the automata, it is possible to simulate these mechanisms: when an automaton is inactive, it is considered as not being part of the system any more, leaving this particular state corresponds to the creation of the agent and entering it as a removal. The strong constraint is that we have to maintain the "ghost" automata in the model even when they are not used (and we have to plan enough of them as an "agent reserve"). Still, since we use rules based on conditions, the systematic addition of tests excluding automata in *Out* state is enough.

Implementation/Simulation. It is possible to define these notions by transition systems morphisms. A morphism of a system $ST = (\Omega, Act, \longrightarrow, \overrightarrow{s}_{in})$ to $ST' = (\Omega', Act', \longrightarrow', \overrightarrow{s}'_{in})$ is a couple of applications (σ, ρ): $\sigma : \Omega \longrightarrow \Omega'$, $\rho : Act \longrightarrow Act'$ such as:

- $\sigma(\overrightarrow{s}_{in}) = \overrightarrow{s}'_{in}$ and
- If $\rho(a)$ is not the "null" global action : $\overrightarrow{s}_0 \xrightarrow{a} \overrightarrow{s}_1 \implies \sigma(\overrightarrow{s}_0) \xrightarrow{\rho(a)} \sigma(\overrightarrow{s}_1)$
  If $\rho(a)$ is the "null" global action: $\overrightarrow{s}_0 \xrightarrow{a} \overrightarrow{s}_1 \implies \sigma(\overrightarrow{s}_0) = \sigma(\overrightarrow{s}_1)$.

It can be noted that several different implementations of the same system can exist. The interesting part here is that it provides a simple means to establish that two transitions

systems are describing the same system but with a different "precision" or different processes. This can be of importance for the validation of the adequacy of a multi-agent system being proposed to solve a specific problem: we can prove the existence of a couple (σ, ρ) showing that the model of the system is truly implementing the model of the proble.

7 Analysis and conclusion

We presented in this paper the definitions and notations which describe a general model based on a form of extended automata product dedicated to a wide range of multi-agent systems.

Current agent and multi-agent models are either informal or considering specific aspects. Some are agent centred like the BDI model [1], others focus on interaction, influence [14], dependences [8] [18] or organisations and roles [19]... The majority of formal models are based on logic and presuppose that the memory of an agent is infinite, as well as that the agent has no temporal constrains. Other models are based on formal languages like Z [17], OZS [12] and have as an objective to facilitate the validation and verification during the design and implementation of the associated software. The model we propose is a general model, not dedicated in any way to a specific agent system architecture. Its objective is to be a tool for the verification of properties at the agent level as well as at the system level, and therefore to enable to better understand the relations between the micro-level (the agents) and the macro-level (the system) where can occur some emergent phenomena.

One of the advantages to use a transition system and automata to model a multi-agent system is the possibility for automated proof and the use of model-checking tools. In numerous tools, systems are extended forms of automata and transitions are described as we do by pre-conditions on a starting state and the transformation of this state in a destination state. These systems also enable the use of most temporal logics, linear or not. It can be noted that, because the model takes place at the level of each agent, it becomes possible to prove properties like convergence, for example, for multi-agent systems with an emergent functionality. Automated proof tools are numerous, each with is specificities: SPIN [13] (on-the-fly proofs), MEC 5 [11] (expressiveness), NuSMV 2 [9] (open kernel which can be integrated in other tools), PRISM [16] (stochastic), UPPAAL [4] (temporal), Kronos [25] (real time systems), TINA [3] (Petri net)... Because of the generic nature of the model, all the proof tools seem to be relevant and the choice will depend on properties to be proved and specificities of the systems of which we are building a model.

The next step concerns automated proofs of properties related to the dynamics of adaptive multi-agent systems with an emergent functionality based on cooperative self-organisation, and in particular of the question of convergence of such systems.

References

[1] S. Rao Anand and P. Georgeff Michael. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[2] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Masson, 1992.

[3] F. Vernadat B.Berthomieu, P.O. Ribet. The tool tina - construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14):2741–2756, Juillet 2004.

[4] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on UPPAAL. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, number 3185 in LNCS, pages 200–236. Springer–Verlag, September 2004.

[5] N. Busi, R. Gorrieri, and GL. Zavattaro. A process algebraic view of linda coordination primitives. In *Theoretical Computer Science*, volume 3, 1998.

[6] D. Capera, Georgé JP., M-P. Gleizes, and P. Glize. The amas theory for complex problem solving based on self-organizing cooperative agents. In *1st International workshop on Theory and Practice of Open Computational Systems (TAPOCS) at IEEE 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2003)*, pages 383–388. IEEE Computer Society, 9-11 June 2003.

[7] C. Carabelea, O. Boissier, and A. Florea. Autonomie dans les systèmes multi-agents : tentative de classification. In *Actes des 11èmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, Hammamet, Tunisie, Novembre 2003. Editions Hermès.

[8] C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3 – Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*. MElsevier Science, 1995.

[9] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. International Conference on Computer-Aided Verification (CAV 2002)*, volume 2404 of *LNCS*, Copenhagen, Denmark, July 2002. Springer.

[10] J.-P. Georgé, M.-P. Gleizes, and P. Glize. Conception de systèmes adaptatifs à fonctionnalité émergente: la théorie des amas. *Revue d'Intelligence Artificielle*, 17, n°4/2000, 2003.

[11] Alain Griffault and Aymeric Vincent. The mec 5 model-checker. In *CAV: International Conference on Computer Aided Verification*, volume 3114 of *Lecture Notes in Computer Science*, pages 488–491. Springer, July 2004.

[12] V. Hilaire, O. Simonin, A. Koukam, and J. Ferber. A formal approach to design and reuse of agent and multiagent models. In *AAMAS AGENT-ORIENTED SOFTWARE ENGINEERING (AOSE) 2004 Workshop*, 2004.

[13] Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[14] Ferber J. and Muller J.P. Influences and reaction: A model of situated multiagent systems. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS 96)*. The AAAI press, 1996.

[15] Georgé J.P., B. Edmonds, and P. Glize. *Making Self-organizing adaptive multi-agent systems work*, chapter 16, pages 321–340. Kluwer Publishing, 2004.

[16] M. Kwiatkowska, G. Norman, and D. Parker. Prism: Probabilistic symbolic model checker. In *Proc. PAPM/PROBMIV'01 Tools Session*. University of Dortmund, September 2001.

[17] M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the first international Conference*. MAS AAAI Press / MIT Press, 1992.

[18] M. Luck and M. d'Inverno. A formal view of social dependence networks. In C. Zhang and D. Lukose, editors, *In Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*. Springer-Verlag, 1996.

[19] M. McCallum, T. J. Norman, and W.W. Vasconcelos. A formal model of organisations for engineering multi-agent systems. In *ECAI 04 Workshop on Coordination in Emergent Agent Societies*, 2004.

[20] R. Milner. *Communication and concurrency*. Prentice Hall, 1989.

[21] S. Rao and M. P. Georgeff. Bdi agents: From theory to practice. Technical report, Australian Artificial Intelligence Institute, 1995.

[22] G. Weiss. *Multiagent systems. A modern approach to distributed artificial intelligence.* The MIT Press, 1999.

[23] D. Weyns, H. V. D. Parunak, F. Michel, T. Holvoet, and J. Ferber. Environments for multiagent systems, state-of-the-art and research challenges. In *Post-proceedings of the First International Workshop on Environments for Multiagent Systems.* Springer-Verlag, 2005.

[24] M. Wooldridge. *An introduction to multi-agent systems.* John Wiley & Sons, 2002.

[25] S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1:123–133, October 1997.