"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Scientific Studies and Research
Series Mathematics and Informatics
Vol. 19 (2009), No. 2, 357 - 364

INTELLIGENT AGENTS FOR MONITORING SYSTEMS

BOGDAN PĂTRUȚ

Abstract. This paper's aim is to define the concepts of s-agent and multi-agent monitoring system we used for constructing the MageLan and ContTest systems. We used formal language theory to present the environment and the abstract architecture for constructing multi-agent systems.

1. S-AGENTS AND MULTI-AGENT MONITORING SYSTEMS

Intelligent agents have to be reliable in order to offer accuracy in the results, in dissimilar, open or unpredictable environments Software agents are situated in particular environments and capable of autonomous actions, in order to fulfill their objectives. On the other hand, distribution of hardware, software and data offer the possibility for the agents to be fragmented or replicated on diverse nodes on the computer network ([4], [5]).

Definition 1. We use the name of *environment* for a set of elements $\mathrm{E} = \{e_0, e_1, e_2, e_3, ..., e_n\}$ among which there is a relation of partial order marked with "<". We use the notation $e \le f$ for the fact that $e < f$ or $e = f$., respectively $f > e$ if $e < f$.

The environment can be, at a certain point, in a certain state e, which we will express by $st(E) = e$. At first, the environment leaves an initial state e_0, for which $e_0 < e_1, \forall i \in \{1,2,...,n\}$. The state e_n is called final state for which it is considered that $e_i < e_n, \forall i \in \{1,2,...,n\}$.

Definition 2. We use the name of *agent* for a triple of the type (3.1) $A = (S, s_0, R)$ where S is a finite set of states, s0 in S is called the agent's initial state, and R is a set of evolution rules.

Keywords and phrases: intelligent agents, distributed systems; monitoring systems; hyper-encyclopedia
(2000) Mathematics Subject Classification: 68Q45, 68T35

If agent A is in state s, then we express this by $st(A) = s$. Among the states of S there is a special state marked with λ. At first $st(A) = s_0$, and when $st(A) = \lambda$ we say that the agent is inactive. For the rest, the agent is active.

The rules in R are of the type (1) or (2):

$$r_1 = (A, s, e) \to (A, t, f) \quad (1)$$

$$r_2 = (B, s, e) \to (B, t, f) \quad (2)$$

Rule (1) states that if $st(A) = s$, $st(E) = e$, then $st(A)$ becomes t and $st(E)$ becomes f, if $e \le f$. The second rule (2) states that agent A ceases its implementation (*st(A)* becomes λ), transferring the control to agent *B*, for which *st(B)* becomes *t*, and the environment remains in the same state.

If *st(E)* = *e* and there are two agents *A* and *B* with $st(A) = a \neq \lambda$ and $st(B) = b \neq \lambda$ and $(A, s, e) \to (C, z, f)$ and $(B, t, e) \to (D, z', f')$, then we will consider $st(E) = \max(f, f')$ if f and f' are comparable, one of them respectively, if f and f' are not comparable, and $st(A) = 0$ and $st(C) = z$ if $f < f'$, respectively $st(B) = 0$ and $st(D) = z'$, if $f < f'$.

This can be generalised for more active agents.

Definition 3. We use the name of *s-agent* for an *n-uple* of the type (3)

$$S = (C, A_1, A_2, ..., A_n) \quad (3)$$

where C is an agent called coordinating agent, and $A_1, A_2, ..., A_n$ are agents corresponding to the definition above that are called effectors or atomic agents. The coordinating agent will interact directly with the user and the architecture and its functionality depends on the concrete implementation (the examples will be offered in the following chapters).

Because inside an s-agent, the agents transfer the control form one to another, an s-agent behaves like a communicative multi-agent system.

Definition 4. Let there be $S = (C, A_1, A_2, ..., A_n)$ an s-agent. If $s_1, s_2, ..., s_n$ are the states of the atomic agents that make up S (without the coordinator C), we then say that $(s_1, s_2, ..., s_n)$ is the *state* of S (at a given moment).

Definition 5. Let there be $S = (C, A_1, A_2, ..., A_n)$ an s-agent in the environment E, that cannot be modified by the user. If the initial state of *S* is of the type $(0, \lambda, \lambda, ..., \lambda)$ we say that S is a *normal s-agent* (s_{01} marks the initial state of the agent A_1).

Directly, we obtain the following lemma:

Lemma 1. *In a normal s-agent,* $\{s_1, s_2, ..., s_n\} = \{s, \lambda, \lambda, ..., \lambda\}$ *is*

enacted, with $s \neq \lambda$ *, at any point of time t.*

Proof. The s-agent being normal, it follows that its initial state (at the moment t_0) is $(0,\lambda,\lambda,...,\lambda)$, therefore $\{s_{01},s_{02},...,s_{0n}\}=\{s_{01},\lambda,\lambda,...,\lambda\}$, and $s_{01} \neq \lambda$. Let us suppose that the state in the moment t_i is $\{s,\lambda,\lambda,...,\lambda\}$, with s≠λ. This means that an active Ai agent exists and that it is unique, with and $st(A_i)=s \neq \lambda$ and $st(A_j)=\lambda, \forall j \neq i$.If there is a relation of evolution of the type $(A_i,s,e)\rightarrow(A_j,t,f)$, then by applying this relation of evolution, we will obtain in the moment $t_{i+1}: st(A_i)=\lambda$ and $st(A_j)=\lambda$, with $t \neq \lambda$. Therefore the state of the s-agent will become (no matter the situation) $(\lambda,\lambda,...,\lambda,t,\lambda,...,\lambda)$, with $t \neq \lambda$, on the position *j*, therefore. If there is no relation of evolution with (A_i,s,e) on the left side, then the s-agent will remain in the state $\{s,\lambda,\lambda,...,\lambda\}$, with s≠λ. The fact that the agent will remain in that certain state will be called blockage, a notion that we will eventually formally define.

Definition 6. We use the name of *multi-agent monitoring system (MMS) of the environment E*, based on s-agents (or on groups) for a triple of the type (4)

$$SMM=(Sa,L,E) \tag{4}$$

where *Sa* is a set of s-agents, having the same structure, *E* is the environment within which these exist and are implemented, and *L* are communication or linkage rules of the type $C_i \rightarrow C_j$, where C_i and C_j are coordinating agents of some s-agents from *Sa*.

The communication relations among the s-agents form an oriented graph depending on the architecture and the concrete implementation of the multi-agent system, as we will see in the following chapters. Our interest lies in some evolution rules of the environment within an s-agent and the structure and graphic representation of an s-agent.

A SMM containing only normal s-agents is called a normal SMM.

Definition 7. The *purpose* of an s-agent is to take the environment E to a state as close as possible to its final e_n state. If the relation can be obtained (5), we then say that the aim of the s-agent can be *carried out*.

$$st(E)=e \ \wedge \neg \exists f \in E, f \neq e_n: e<f \tag{5}$$

By extension, we can say that the aim of SMM is reached if all the targets of the component s-agents are achieved.

Definition 8. Two rules of evolution of the type r_1 = a→b şi r_2 = b→c, where a, b, and c are triples of the type of those in (1) or (2), they are called

adjacent.

Definition 9. Let there be r_1, r_2, ..., r_k a line of adjacent rules of evolution, two by two. We will use the notation (6) and we will call this relation as the *derivation relation* (see (6)).

$$r_1 \Rightarrow r_k \tag{6}$$

Within an s-agent, the following results are obvious:

Proposition 1. *If st(E)=e, there is an agent A with st(A)=s≠λ, (A, s, e) ⇒ (A', s', e') and there is no f≠e_n, so that e'< f, then the aim of the s-agent can be reached (with the environment in state e'). (From the very beginning, we have noted the final state of the environment with e_n). Most of the times, set E is not fully ordered. If, however, a fully ordered relation „<" is found, then the following sentence is enacted*.

Proposition 2. *If the relation "<" is fully ordered, then the purpose of the s-agent can be reached with the environment in the final state (e_n) if there is a derivation of the type (A, s_0^A, e_0) ⇒ (B, t, e_n)*.

So, the purpose of the s-agent can be reached if there is a derivation which leads the environment to the state en, starting from the environment's initial state e_0 and the initial state of any agent (A), without the user's intervention (in the case of a fully ordered relation "<").

If $|\{A; st(A)\neq\lambda\}| = 1$, for any $e=st(E)$ (therefore a single agent is active at a given moment, as in lemma 1), then the MMS operates sequentially. This is what happens most of the times.

We consider that the environment E modifies its current state in two cases:

- as a result of applying an evolution rule, by one of the system's agents;
- as a result of the direct intervention of a human user. We can also consider, in some exceptional cases, that the state a certain agent is in can be modified by the evolution rules as by the user as well. Therefore, we cannot hold control over what is going to happen, or how the state of the environment is going to evolve within a certain interval of time. If, ideally, the human user cannot randomly modify neither the environment E nor the current state of the agents, then the system is entirely deterministic.

2. BLOCKAGES AND INFINITE CYCLES

Our interest does not lie simply in building absolutely sequential systems or deterministic systems, but in specifying in the best possible way the evolution rules so that blockages cannot occur.

Definition 10. If relation (7) is certified, then we say that the s-agent is

under blockage.

$$\exists A \in S : st(A)=s, st(E)=e, e \neq e_n \wedge \neg\exists (A, s, e') \rightarrow (B, s, f), f, e' \neq e_n \quad (7)$$

Therefore, we say that an s-agent is under blockage when an atomic agent of the system gets into a state *s*, and the environment is in a non-final state *e*, and there is no evolution relation that can allow for the agent's passing out from the state s, although the environment is suddenly modified by the user, into another non-final state *e*'.

In (7) *f* and *e*' are certain non-final states of the environment (*e*'and *f* may also be even *e*), and s' a certain state of a certain agent *B* from the s-agent, being even possible for *B* to be *A*.

In other words, there is no evolution rule, with s on the left side, which can lead to another state of *A* or within another agent *B*, no matter the evolution of the environment *E*.

Definition 11. In an s-agent a derivation of the type $(A,s,e) \Rightarrow (A,s,e)$ is called *infinite cycle*.

This occurs when, if the user does not intervene through modifying the environment, the execution of the s-agent's agents cycles infinitely, without the environment reaching the final state. In this case, the user may intervene to take the MMS out of the cycle.

Blockages and infinite cycles can be identified easier if we represent the s-agents and the MMS. Graphically, a multi-agent monitoring system (MMS) can be represented in the shape of a graph oriented thus (figure 1):

- optionally, more s-agents are represented in the shape of some polygons or other geometrical figures, containing more s-agents, the internal structure being represented only for one of them;
- optionally, the links among the s-agents will have the shape of some curves; graphically only one s-agent will be represented, because all s-agents are considered to have the same internal structure;
- the generic s-agent will be represented through a geometrical figure where all atomic agents are represented;
- the atomic agents are represented through some rectangles labeled with their names; inside those squares there will be circles representing the different states of the respective agent;
- each state will represent a point in an oriented graph, where the edges are the evolution relations, labeled with a pair of clues for the states of the environment: the starting state and the state that is finally reached.

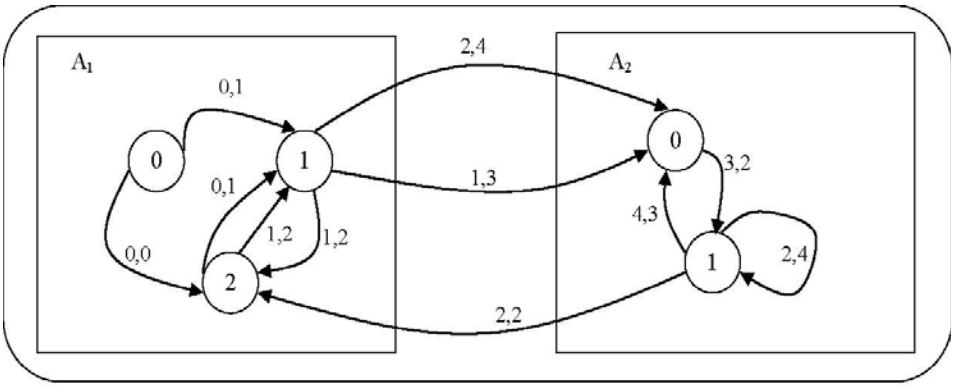

Figure 1. A simple s-agent containing a blockage in A_1 and an infinite cycle in A_2

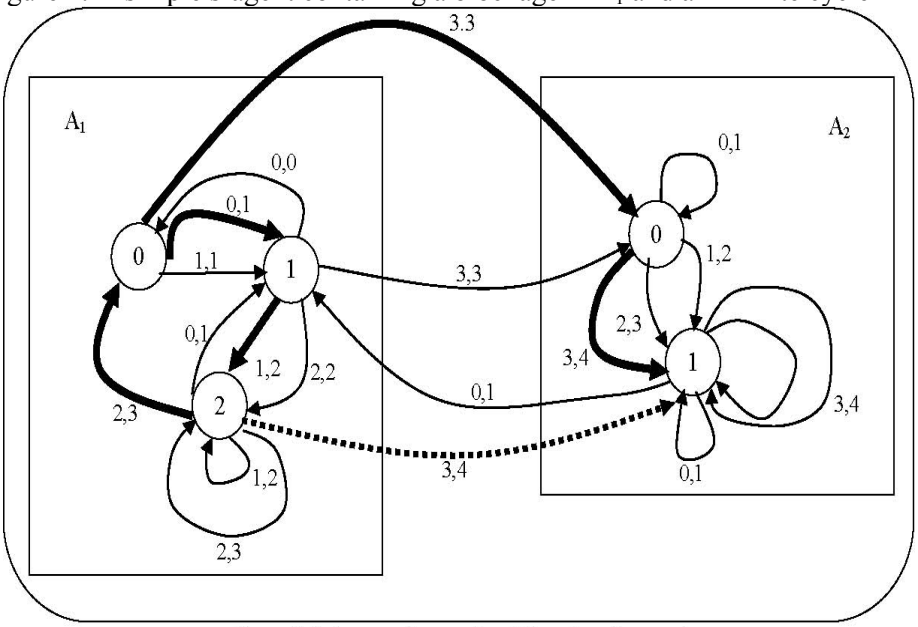

Figure 2. A normal s-agent and its reduced s-agent

This is an ideal example of MMS functioning, illustrated by the bold arcs in figure 2. Obviously, the other arcs except the bold ones are useless in this graph, because they will never be passed through. If, however, the user interferes, for example the moment the MMS is in state 2 from the atomic agent A_1, modifying the state of the environment from e_2 to e_3, then the evolution $(A_1, 2, e_3) \rightarrow (A_2, 1, e_4)$ will occur, expressed in figure 3 through the dotted arc. We can obviously "endow" an s-agent with many evolution rules. Inside a co-operating working environment, different users will introduce different rules of evolution. The question is, if under the circumstances of some normal agents, some of those rules will ever be applicable, will ever

come into action. It is as if we had a program with functions or pieces of codes which are not resorted to by anywhere, cannot be touched, or an expert system with production rules whose part of the premises will never be fulfilled.

Thus, we will show that normal agents can be reduced to other normal agents, on the basis of an algorithm, so that certain evolution rules could become useless and could be eliminated from the system, the later behavior being not affected. On the contrary, the systems become more simple. By an s-agent' behavior we mean the applicability of the rules of its component agents. Therefore, if a certain evolution rule could ever be applied, it will be a part of the s-agent's behavior, and if not, then it will not be a part of this behavior.

3. PRACTICAL IMPLEMENTATIONS AND FUTURE WORKS

Let there be S_1 and S_2 two normal s-agents, functioning simultaneously within the same environment E. If the behavior of S_1 is identical with the one of S_2 at all moments, meaning that the same evolution rule is applied both in S_1 and S_2, we say that the two agents are equivalents. Equivalence also refers to the situations of blockages or infinite cycles.

Our next interest is to consider normal s-agents and reducing them. In our next research, we will proof that a normal s-agent S can be reduced to a normal s-agent T that is equivalent with S, by eliminating some evolution relations, according to a specific algorithm.

One first example of MMS is that where the environment is given by an intelligent hyper-encyclopaedia ([1],[2]) and the MAgeLan multi-agent system.

A second example is given by an instructive system for accounting, ContTest ([3]).

References

[1] Pătruţ, B., Pandele, I., 2008, **How to Compute the References Emergencies in a Hyper-encyclopedya**, in "Recent Advances in Systems Engineering and Applied Mathematics. Selected papers from the WSEAS conferences in Istanbul, May 27-30, 2008", ISBN 978-960-6766-91-6, ISSN 1790-2769, Istanbul, Turkey, p.72-75

[2] Pătruţ, B., Socaciu, T., 2008, **Constructing a Hyper-enciclopedya: How to Transfer the Emergencies Between the Nodes**, in Proceedings of the Fourth International Bulgarian-Greek Conference (Computer Science' 2008), 18-19 September 2008, Kavala, Greece, p. 468-473

[3] Pătruț, B., Vârlan S. E., Socaciu, T., 2008, **ContTest – a Multiagent System for Accounting Education**, in Proceedings of the Third International Multi-Conference on Computing in the Global Information Technology, ICCGI 2008, July 27 - August 1, 2008, Athens, Greece
[4] Vlassis, N., 2007, **A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning**, Department of Production Engineering and Management, Technical University of Crete, Morgan and Claypool Publishers
[5] Weiss Gerhard (ed.), 1999, **Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence**, MIT Press, ISBN 978-0-262-23203-6

Department of Mathematics and Informatics,
Faculty of Sciences
"Vasile Alecsandri" University of Bacău,
Spiru Haret 8, 600114 Bacău, ROMANIA
Email: bogdan@edusoft.ro