A formal methodology to specify hierarchical agent-based systems *

César Andrés, Carlos Molinero, and Manuel Núñez
Dept. Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid, 28040 Madrid, Spain
e-mail: c.andres@fdi.ucm.es, molinero@fdi.ucm.es, mn@sip.ucm.es

Abstract

In this paper we introduce a formal framework to specify agent-based systems where each agent is specialized in a single task that will be fulfilled by making calls to other simpler agents. In other words, we are interested in systems that can perform a task by subdividing it in easier tasks and by using the knowledge about each agent already introduced in the system. The idea is to prefabricate a basic structure that can be reused by either changing the main goal or by adding several different specialized agents.

The main characteristic of our methodology is that each complex agent contains a schematic definition of other agents. Each agent is thus able to retain and produce certain information, such as the time needed to accomplish a certain task, taking into account a given set of agents and resources. This allows to quickly produce information regarding the necessities in resources and derive the demands to other subsystems.

In order to increase the applicability of our approach, we have fully implemented a tool that allows us to graphically specify complex systems. In addition, the tool allows us to simulate the behavior of the specified systems so that some interesting properties, such as starvation and maximal progress, can be studied.

1 Introduction

The representation and study of communities where intelligent (electronic) agents replace their (human) owners is a topic that has attracted a lot of interest. In particular, there is ongoing research in technologies able to model users by means of agents which autonomously perform electronic transactions (see [9] for a survey on the topic). In order to increase the power of these agents they must know the preferences of the corresponding user. In this line, the concept of *utility function* is very useful. Essentially, a *utility function* returns a real number for each possible basket of goods: The bigger this number is, the happier the owner is with this basket. Intuitively, agents should appropriately simulate the systems that they are representing by considering the utility function that would establish the expected behavior (see e.g. [25, 7, 6, 14, 19, 12]). In fact, there exist several proposals showing how agents can be trained to learn the preferences of users (see e.g. [2, 6, 26]). Besides, a formal definition of the preferences of the user provides the agent with some negotiation capacity when interacting with other agents [13, 26, 17]. Let us remark that, in most cases, utility functions take a very simple form. For instance, they may indicate that an agent A is willing to exchange the item a by the items b and c.

Even though there are general purpose formalisms to formally describe complex concurrent systems (such as process algebras [11, 21, 3] or Petri Nets [4, 5]) they are not suitable to describe agents since these languages and notations do not provide specific operators to deal with the inherent characteristics of agents. However, there has been already several studies to formally describe the use of intelligent electronic agents that are nested into one another (see, for example, [15, 16] for two approaches based on Petri Nets and automata, [22, 23] for approaches based on process algebras, and [24, 20] for approaches based on finite state machines). Most of these approaches have been created in favor of comprehensibility. Therefore they facilitate to derive and apprehend new properties.

Even though there are already several formal approaches to describe the systems that we are interested in, our experience shows that there is a need for another viewpoint to confront this problem. If we try to incorporate the base of facts to a system, there will always be a lack of capacity to implement every possible structure of the agent, every different solution to the same problem, and every combination of small pieces that constitute a complex problem. This is the reason why we think that there is a need for a new

*Research partially supported by the Spanish MEC project WEST-/FAST (TIN2006-15578-C02-01) and the Marie Curie project TAROT (MRTN-CT-2003-505121).

framework to formally define the class of systems previously described. We believe it is easier and more feasible to incorporate bits of knowledge by having the system recomposing this information into complex tasks. This approach simplifies at least two aspects. First, it helps to ensure the completion of the base of facts. Second, it allows to relocate the different agents, due to its modularity, so that they can be spread over a network to parallelize some of the tasks.

If we take a look in another direction, we would like to assimilate the systems that we are interested in to a *common places* structure in which one is able to locate the rest of the structure from higher order points. If we use the subway lines as a metaphor, we only need to know the location of the different stations, but the exact location of that small fruit shop that we are trying to reach is bounded to the location of the closest metro station. Once we arrive to that particular metro station, we will check the neighborhood map so that we can find the shop; we do not need to know in advance all the local maps associated with all the stations of the network. This is how our system will work: Once we have all the atomic agents, each time that a new complex agent, embracing the knowledge of several atomic agents, is created we will refer to this new agent when making subsequent calls to the system. In this line, we are able to *forget* how atomic actions are performed because we have a higher order element to which we can call upon. In any case, even with a complex structure, atomic agents are still the ones that execute *real* tasks. Using another metaphor we could say that our systems are similar to economic structures in which there exist intermediate agents that gives us the result of the transformation of resources as a final product. These agents, in a hidden way, contract the prime manufacturers that create these resource transformations.

Another point in favor of our approach is that it allows us to have an unbounded growth (equivalently, subdivisions as small as needed) either by adding systems in between existing ones or by assigning new atomic agents to the ones that we had before. It is important to note that the way our systems are subdivided, in so called *communication cellules*, facilitates their deployment in a distributed system in which one can obtain a perspective of variable magnitude of the global tasks. This holds as long as we keep the hierarchical structure of the ensemble.

The rest of the paper is organized as follows. In Section 2 we introduce some auxiliary notation that will be used during the rest of the paper. In Section 3 we present the formalism to describe our systems. In Section 4 we briefly present our tool and give a small example to show its main features. Finally, in Section 5 we present our conclusions and some lines for future work.

2 Preliminaries

In this section we introduce some notation that will be used throughout the rest of the paper.

Since users have different preferences, in order to properly design agents the first step consists in expressing these preferences. In order to extract preferences from users several mechanisms have been presented in the literature (see e.g. [6, 8, 10]). In this paper, preferences in a given moment will be given by a *utility function*. These functions associate a value (a utility measure) with each possible combination of resources a user could own. Alternatively, other mechanisms such as *preference relations* could be used (see e.g. [18] for conditions to transform one of the possibilities into the other).

Definition 1 Let $\mathbf{R}_+ = \{x \in \mathbf{R} \mid x \geq 0\}$. *Vectors* in $\mathbf{R}_+^n$ (for $n \geq 2$) are usually denoted by $\bar{x}, \bar{y}, \ldots$ The vector $\bar{0}$ denotes the tuple having all the components equal to zero. Given $\bar{x} \in \mathbf{R}^n$, x_i denotes its *i-th* component. Let $\bar{x}, \bar{y} \in \mathbf{R}_+^n$. We write $\bar{x} \leq \bar{y}$ if for all $1 \leq i \leq n$ we have $x_i \leq y_i$.

If there exist n different kinds of resources then a *utility function* is any function $f : \mathbf{R}^n \mapsto \mathbf{R}_+$. □

Intuitively, $f(\overline{x}) > f(\overline{y})$ means that $\overline{x}$ is preferred to $\overline{y}$. For instance, if the resource x_1 denotes the amount of apples and x_2 denotes the amount of oranges, then $f(x_1, x_2) = 3 \cdot x_1 + 2 \cdot x_2$ means that, for example, the agent is equally happy owning 6 apples or 9 oranges. Let us consider another agent whose utility function is $f(x_1, x_2) = x_1 + 2 \cdot x_2$. Then, both agents can make a deal if the first one gives 3 oranges in exchange of 4 apples: After the exchange both are happier. Alternatively, if x_2 represents the amount of money in any currency (for example in dollars) then the first agent would be a customer while the second one might be a vendor. A usual assumption is that no resource is a *bad*, that is, if the amount of a resource is increased, so does the value returned by the utility function. Using a derivative expression, this property can be formally expressed as $\frac{\Delta f(x_1,\ldots,x_n)}{\Delta x_i} \geq 0$ for all $x_1, \ldots, x_n \in \mathbf{R}$ and $1 \leq i \leq n$. This requirement does not constrain the expressive power of utility functions, as the existence of any undesirable resource can be always expressed by considering a resource representing the *absence* of it.

During the rest of the paper we consider that agents use messages to communicate among them. The next definition introduces the different kinds of messages that can be sent.

Definition 2 We consider seven different types of messages and we enumerate them as follows:

TYPE	CONTENT
1	Information
2	Negotiation
3	Proposal message
4	Acceptance message
5	Hiring message
6	Job started message
7	Job finished message

Let ID be a set of agent identifiers. A *message* is a tuple (T, s, d, c, t, D, Z), where T is the *type* of message, $s \in ID$ is the agent source of the message, $d \in ID \cup \{\star\}$ is the agent destination of the message ($\star$ represents a broadcast message) $c \in String$ represents the string of characters containing the message, t is the information regarding the time that it takes an agent to perform a task, D is the information of the necessary conditions to start the task and Z is the transformation of resources function that the agent emitting the message applies to its set of resources.

We denote by λ the empty message. We denote by $\mathcal{M}$ the set of all messages. □

3 Definition of the formalism

In this section we present our formal language to specify complete systems as well as all the agents taking part in them. The main idea consists in having a *world* that is composed of *communication cellules*. These components will be interconnected in a hierarchical way, that is, the main communication cellule will hold the main agent while each of the next communication cellules will hold simpler agents. This process is iterated until we reach the last level that will hold only atomic agents.

We will make a distinction between *generic agents*, that is, agents fulfilling a complex task (which will be done by calling other agents) and *atomic agents*, which are basic agents in charge of executing simple tasks. This distinction is merely represented as a different set of variables taken from the same general definition of an agent. Agents will send messages using communication cellules. These cellules will forward these messages to other cellules that will broadcast them to the agents under their control, until the final atomic agent is reached. Then, an atomic portion of the global goal will be produced through the transformation of resources that this atomic agent performs.

Each agent will have a different utility function, as defined in the previous section, depending on the *utility* that each of the resources represent for the specific agent. This function will take into account different combinations of resources to decide which task should be performed.

We will start by defining the simpler element in the system, the agents, and scale up in complexity to define the complete system.

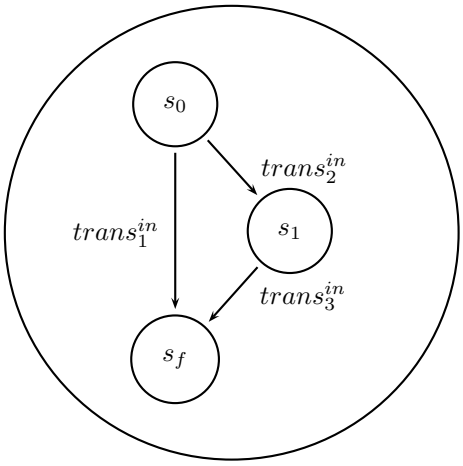


Figure 1. A generic agent.

Definition 3 We denote by $\mathcal{A}$ the agent domain; $a_1, \ldots, a_n$ will be used to denote elements of $\mathcal{A}$. We consider a set ID containing all the agent identifiers.

An *agent* is a tuple $(id, S, s_0, s_f, R, \mathcal{V}, T_{ex}, T_{in}, ib, ob, \Gamma)$ where

- $id \in ID$ is the *agent identifier*.
- S is the *set of states*.
- $s_0 \in S$ is the *initial* state.
- $s_f \in S$ is the *final* or *goal* state.
- R is the *set of resources* of the agent.
- A *valuation* is a function $\rho : R \mapsto \mathbf{R}_+^m$, being $|R| = m$, that returns the current value of each variable. We denote by $\mathrm{Val}(R)$ the set of all valuations R. $\mathcal{V} \in \mathrm{Val}(R)$ is the initial valuation of the agent.
- $T_{in} \subseteq S \times ID \times \mathbf{R}_+ \times S \times \wp(\mathrm{Val}(R)) \times \mathrm{Val}(R)$ is the set of internal transitions. The set of external transitions must fulfill the following restrictions:
  - $|\, T_{in} \,| \geq 1$.
  - If $|T_{in}| = 1$, that is $T_{in} = \{(s_1, id', t, s_2, D, Z)\}$ then $id' = id$.
  - If $|\, T_{in} \,| > 1$ then for all $(s_1, id', t, s_2, D, Z) \in T_{in}$ we have $id' \neq id$.
- $T_{ex} \subseteq \mathcal{M} \times \wp(\mathcal{M}) \times \wp(\mathcal{M})$ is the set of external transitions.
- $ib, ob \in \wp(\mathcal{M})$ are the input and output buffers, respectively. We will use them to receive incoming messages (ib) and to send outgoing messages (br). We can use the functions $\mathrm{Choose} : \wp(\mathcal{M}) \mapsto \mathcal{M}$,

Remove : $\wp(\mathcal{M}) \times \mathcal{M} \mapsto \wp(\mathcal{M})$ and Concat : $\mathcal{M} \times \wp(\mathcal{M}) \mapsto \wp(\mathcal{M})$, having the expected meaning, to manage buffers.

- $\Gamma : \mathrm{Val}(R) \mapsto \mathbf{R}_+$ is the utility function of the agent.

□

Intuitively, an internal transition $(s_1, id', t, s_2, D, Z) \in T_{in}$ is a tuple where s_1 is the initial state, s_2 is the final state, id' identifies the agent assuming the transition[1], $t \in \mathbf{R}_+$ is the time[2] consumed to perform the transition, $D \subseteq \mathrm{Val}(R)$ denotes a subset over the set of valuations that denotes which valuations allow the transition to occur, that is, the transition can be performed only if the current valuation belongs to this set, and $Z : \mathrm{Val}(R) \mapsto \mathrm{Val}(R)$ is the transformation of resources produced by this transition. We write $s_1 \xrightarrow{id',t,D,Z}_{in} s_2$ as a shorthand of $(s_1, id', t, s_2, D, Z) \in T_{in}$.

External transitions are tuples $(m, ob_{origin}, ib_{destination})$ where m is the message being transmitted containing all the necessary information, ob_{origin} is the output buffer of the agent/communication cellule that originates the message and $ib_{destination}$ is the incoming buffer of the communication cellule/agent that receives the message. Let us remember that a $\star$ message denotes a broadcast message, that is, a message that will be transmitted to all agents belonging to the destination cellule. We write $ob_{origin} \xrightarrow{m}_{ex} ib_{destination}$ as a shorthand of $(m, ob_{origin}, ib_{destination}) \in T_{ex}$.

In Figure 1 we show a graphical representation of a generic agent. Next, we introduce some auxiliary concepts that will be useful to describe the evolution of agents.

Definition 4 An agent is called *atomic* if it is in charge of executing a single task. Formally, we use a function $\mathrm{atomic} : \mathcal{A} \mapsto Bool$, such that for all $a = agent$ we have $\mathrm{atomic}(a) = (\mid T_{in} \mid = 1)$.

In order to have the current state of an agent we specify its configuration as an element belonging to $S \times \mathrm{Val}(R) \times \wp(\mathcal{M}) \times \wp(\mathcal{M})$. Configurations are modified through the performance of either internal or external steps:

Internal step of the system:

Given a configuration $M = (s, \mathcal{V}, ib, ob)$, an internal transition $s_1 \xrightarrow{id',t,D,Z}_{in} s_2$ will be triggered if $D(\mathcal{V})$ and will modify the configuration to $(s_2, Z(\mathcal{V}), ib, ob)$.

External step of the system:

Given a configuration $M = (s, \mathcal{V}, ib, ob)$ and another one $M' = (s', \mathcal{V}', ib', ob')$ belonging to a second agent, an external transition $(m, ob_{origin}, ib_{destination}) \in T_{ex}$ will modify the configurations to $M = (s, \mathcal{V}, ib, \mathrm{Remove}(ob, m))$ and $M' = (s', \mathcal{V}', Concat(m, ib'), ob')$. □

Agents are grouped into communications cellules. These structures are useful to organize, connect, and produce independent agents of different complexities. The need for this kind of organization is clear if we take into account the fact that when making a call to find an agent that fulfills the considered activity, if they would be grouped into the same structure, all agents, either atomic or complex, would answer the request, forcing the new agent to consider too many possibilities and therefore making the system unusable. The communication cellules allow to first ask more complex agents and then if none knows how to answer then the request will go on to next level, having agents of a greater simplicity.

Definition 5 A communication cellule is a tuple (l, A, ib, ob), where

- $l \in \mathbf{N}$ denotes the level of the cellule: Higher levels indicate more complex tasks.
- $A \subseteq \mathcal{A}$ is the set of agents associated with this cellule.
- $ib, ob \in \wp(\mathcal{M})$ are the incoming and outgoing buffers, respectively. We will use the functions introduced in Definition 3 to manage them.

The set of all communication cellules is denoted by $\mathcal{C}$. □

Let us remark that when an agent is added to a cellule, the incoming buffer of the agent is connected to the outgoing buffer of the cellule, while the outgoing buffer of the agent is associated with the input buffer of the cellule. This means that an agent does not send messages to another agent; it sends them to the associated cellule. Similarly, an agent receives messages only from the cellule to which it is attached. A graphical representation of this can be seen in Figure 2 where we can observe several agents connected to the cellule C.

In the next definition we introduce the concept of world, that is, a set of cellules interconnected in the appropriate way.

Definition 6 We say that the tuple $W = (\mathcal{C}_W, \mathcal{F}_W, \mathcal{S}_W)$ is a *world* if $\mathcal{C}_W \subseteq \mathcal{C}$ is a set of cellules, and $\mathcal{F}_W, \mathcal{S}_W : \mathcal{C}_W \mapsto \mathcal{C}_W \cup \{\lambda\}$ are two injective functions such that given a cellule C we have that $\mathcal{F}_W(C)$ (resp. $\mathcal{S}_W(C)$) returns the father (resp. son) of C. If a cellule does not have a father (resp. son) then the special symbol λ will be returned.

[1] If $id = id'$ then we are considering an atomic agent that is itself in charge of performing the transition

[2] This value will be defined by default only for atomic agents since complex agents will calculate the associated time from the information collected from the contracts with other agents. In addition to consider the sum of all the involved time values, we have to take into account the time that it takes to perform communications among agents.

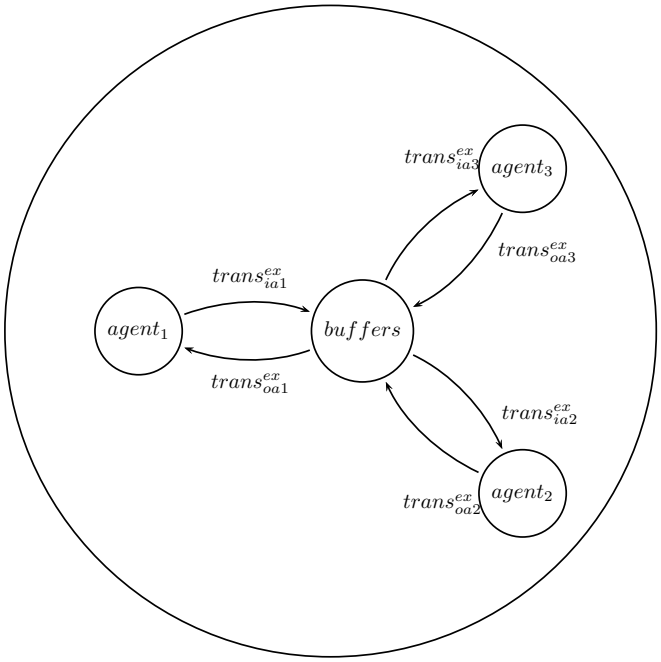


Figure 2. Communication cellule.

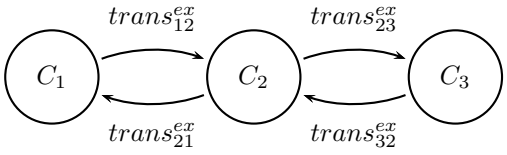


Figure 3. World.

In order to have meaningful worlds, we suppose that for all $C, C' \in \mathcal{C}_W$ we have $\mathcal{F}_W(C) = C'$ if and only if $\mathcal{S}_W(C') = C$. Moreover, we also assume that $\mathcal{F}_W$, through transitivity, induces a total lineal order. □

Let us remark that the previous conditions on worlds imply that we have a *linear* structure of cellules. In particular, $\mathcal{S}_W$ will also induce a linear order. Moreover, if our world contains more than one cellule, we also have that there are exactly two cellules $C, C' \in \mathcal{C}_W$ such that $\mathcal{F}_W(C) = \lambda$ and $\mathcal{F}_S(C') = \lambda$. In Figure 3 we show a graphical representation of a world including three cellules.

4 The $\mathcal{A}\backslash\sqcup$ tool

In this section we briefly describe the $\mathcal{A}\backslash\sqcup$ tool. In the previous section we have presented a modular framework that has the capability of expressing constraints, specifications, agents, cellules, and resources. The $\mathcal{A}\backslash\sqcup$ tool, facilitates the definition of systems so that a user of our methodology can abstract most of the mathematical technicalities needed to define a system/world. We illustrate the behavior of the most relevant phases of the framework by following an example from scratch.[3] First we will show how we can create a new world, assigning resources, agents, cellules, etc. Second, we will see the communication among agents, in order to obtain resources, subcontract other agents, etc. Finally we will show how we achieve the proposed goal.

In order to start the simulation, a preliminary phase is necessary to create the world, the cellules, and the agents. We create the world called `Complutense` and inside it we add three different Cellules (C_1, C_2, C_3) with some agents in them. In Figure 5 we can observe how $\mathcal{A}\backslash\sqcup$ shows the generated world as well as an agent of this world. The set of agents that we have introduced in the system is defined in Figure 4.

As we can observe in the table, agent_{15} is an atomic agent because it is in the lower level of the cellules and, according to the definition of level, this cellule can only have atomics agents; an atomic actions only goes from s_n to s_{n+1}. The values that are in the fifth column represent the money and the time. Even though there are other resources involved in the system we have only represented these two resources since they are the most relevant for our example. `Complutense` represents a world where agents can build houses. We have subdivided the task of *building* in six states. In Figure 6 we give a description of each of these states.

Once the world is created, we start with the second phase. We generate the connections that are in charge of

[3] Even though this is a toy example, so that we can concentrate on the main features of the tool, we have already tested our tool with some more complex examples. However, as we indicate in the last section of the paper, we still need to use our tool to specify a *real* system.

NAME	LEVEL	INITIAL	RESOURCES	GOAL
agent_1	1	s_1	(410, 290)	s_6
agent_2	1	s_1	(370, 340)	s_6
agent_3	2	s_1	(175, 151)	s_3
agent_4	2	s_1	(220, 241)	s_4
agent_5	2	s_1	(312, 287)	s_5
agent_6	2	s_2	(149, 150)	s_4
agent_7	2	s_2	(224, 159)	s_5
agent_8	2	s_2	(220, 176)	s_6
agent_9	2	s_3	(231, 148)	s_6
agent_{10}	2	s_4	(149, 101)	s_6
agent_{11}	3	s_1	(70, 100)	s_2
agent_{12}	3	s_2	(75, 50)	s_3
agent_{13}	3	s_3	(72, 68)	s_4
agent_{14}	3	s_4	(72, 35)	s_5
agent_{15}	3	s_5	(72, 43)	s_6

Figure 4. Agents in the world.

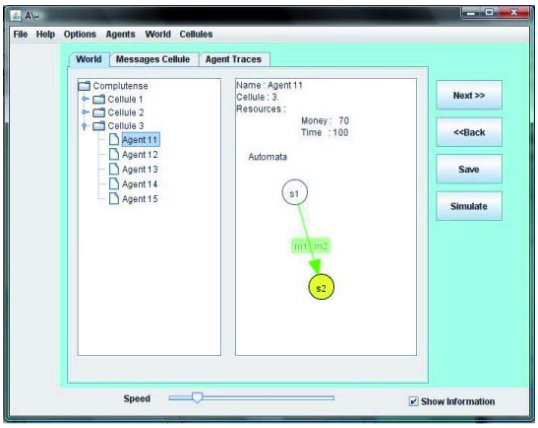
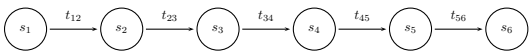


Figure 5. Phase 1 in $\mathcal{A}\backslash\sqcup$



s_1 :money.
s_2 :money+land.
s_3 :money+land+plan.
s_4 :money+land+plan+construction materials.
s_5 :money+land+plan+construction materials+worker.
s_6 :money+land+plan+construction materials+worker+house.

Figure 6. Global task.

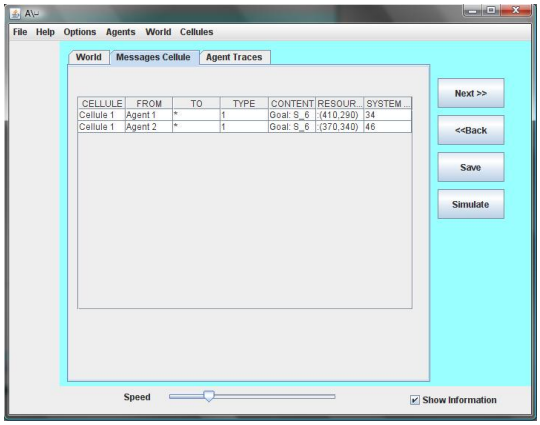


Figure 7. Phase 2 in $\mathcal{A}\backslash\sqcup$

telling the agent that it has reached a certain state or to allow the agent to move to a new state (and, subsequently, call the next agent to perform the new task). We try to let it solve a certain task. This phase is presented in Figure 7.

First agent_1 and agent_2 send a message. Both of them need to move from state 1 to state 6, but agent_1 owns more money than agent_2 while it has less time to reach its goal. So, we expect that agent_1 will find a faster, although more expensive, way to proceed than agent_2.

We will show the decision that agent_1 takes in order to obtain a faster path. The agent_2 follows a similar process to agent_1. The first message from agent_1 has type=1, that is, information, and it is a broadcast message to another communication cellule. This message is used to look for all agents in that cellule that have the same final state. This first message is sent from C_1 to C_2.

The agents agent_8, agent_9 and agent_{10} obtain their goal when they reach state s_6. All of them answer to the previous message by sending information concerning their amounts of *resources*. After receiving the answers, agent_1 starts building a tree in order to decide the best path to the goal.

Now agent_1 asks, by sending a broadcast message to C_2, which agents have as goal state s_2, s_3 and s_4. These messages flow through C_2. We have that agent_4, agent_6, and agent_3 will answer with messages sending s_1, s_1 and s_2. Respectively because the initial state of agent_1 is s_1, it only has to obtain one possible way for s_2.

In the final step of this phase agent_1 sends to C_2 a message asking for agents whose goal is s_2. No agents will answer to it. The $\mathcal{A}\backslash\sqcup$ tool has a timeout module which sends an internal transition for agent_1 denoting that nobody is going to answer this last message. When this timeout is sent, agent_1 starts to decide which path it prefers to follow according to its utility function.

In phase 3 we have the situation described in Figure 8.

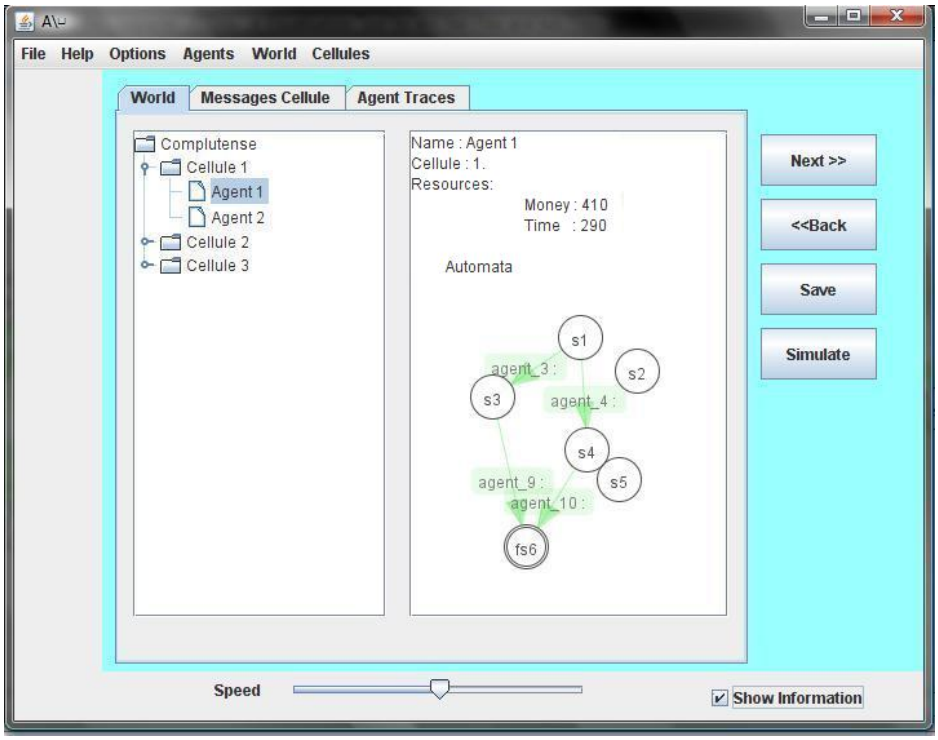


Figure 8. Phase 3 in $\mathcal{A}\backslash\sqcup$

First, agent_1 has all the possible paths to achieve its goal already displayed in front of him. By applying its utility function, as an example $\Gamma = 3 * money + 1/time$ which will give for path $agent_3$+$agent_9$ a value of 1218.00 and for the other path ($agent_4$+$agent_{10}$) a value of 1107.00, therefore this agent decides that the best way is to step through s_3. So, it calls agent_3 to start its internal processes. Then, agent_3 will then restart the whole process until it finally calls agent_{11} which is an atomic agent. Therefore, it will start to transform the resources by taking part of the money (in s_1) and transforming it into land (reaching s_2).

5 Conclusions and future work

In this paper we have presented a formalism to represent complex hierarchical systems where tasks can be distributed and/or *subcontracted* among agents. We are aware that our formalism is difficult to use since there are a lot of mathematical machinery underlying the definition of our systems. Thus, we have decided to build a tool that fully implements our methodology. In this way, a user of our methodology does not need to pay attention to the formal details and can concentrate on defining the appropriate hierarchical structure.

There are at least two lines for future work. On the one hand, there is a lot of room to continue the theoretical study. In particular, we can exploit the trace relation between agents so that we can define a *conformance relation* to determine whether a real system correctly implements one of our worlds. On the other hand, more practical, we have used our tool only with small/medium size examples. We are working on the complete definition of a *real* system by using our tool. Specifically, we are considering [1] as a non-trivial system to be described in our tool.

References

[1] C. Andrés, M. Merayo, and M. Núñez. Formal development of a complex information system. In *3rd Int. Conf. on Systems, ICONS'08*, pages 118–123. IEEE Computer Society Press, 2008.

[2] F. Bacchus and A. Grove. Graphical models for preference and utility. In *Uncertainty in Artificial Intelligence, UAI'95*, pages 3–10. Morgan Kaufmann, 1995.

[3] J. Bergstra, A. Ponse, and S. Smolka, editors. *Handbook of Process Algebra*. North Holland, 2001.

[4] W. Brauer, W. Reisig, and G. Rozenberg, editors. *Petri Nets I: Central Models and Their Properties*. LNCS 254. Springer, 1987.

[5] W. Brauer, W. Reisig, and G. Rozenberg, editors. *Petri Nets II: Applications and Relationships to Other Models of Concurrency*. LNCS 255. Springer, 1987.

[6] M. Dastani, N. Jacobs, C. Jonker, and J. Treur. Modelling user preferences and mediating agents in electronic commerce. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective, LNCS 1991*, pages 163–193. Springer, 2001.

[7] T. Eymann. Markets without makers - a framework for decentralized economic coordination in multiagent systems. In *2nd Int. Workshop on ELectronic COMmerce, WELCOM'01, LNCS 2232*, pages 63–74. Springer, 2001.

[8] B. Geisler, V. Ha, and P. Haddawy. Modeling user preferences via theory refinement. In *5th Int. Conf. on Intelligent User Interfaces, IUI'01*, pages 87–90. ACM Press, 2001.

[9] R. Guttman, A. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey. *The Knowledge Engineering Review*, 13(2):147–159, 1998.

[10] V. Ha and P. Haddawy. Similarity of personal preferences: Theoretical foundations and empirical analysis. *Artificial Intelligence*, 146(2):149–173, 2003.

[11] C. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[12] J. Keppens and Q. Shen. A calculus of partially ordered preferences for compositional modelling and configuration. In *AAAI Workshop on Preferences in AI and CP: Symbolic Approaches*, pages 39–46. AAAI Press, 2002.

[13] S. Kraus. Negotiation and cooperation in multi-agent systems. *Artificial Intelligence*, 94(1-2):79–98, 1997.

[14] J. Lang, L. v. Torre, and E. Weydert. Utilitarian desires. *Autonomous Agents and Multi-Agent Systems*, 5(3):329–363, 2002.

[15] I. Lomazova. Communities of interacting automata for modelling distributed systems with dynamic structure. *Fundamenta Informaticae*, 60(1-4):225–235, 2004.

[16] I. Lomazova. Nested Petri Nets for adaptive process modeling. In *Pillars of Computer Science, Essays Dedicated to Boris Trakhtenbrot on the Occasion of His 85th Birthday, LNCS 4800*, pages 460–474. Springer, 2008.

[17] A. Lomuscio, M. Wooldridge, and N. Jennings. A classification scheme for negotiation in electronic commerce. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective, LNCS 1991*, pages 19–33. Springer, 2001.

[18] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[19] M. McGeachie and J. Doyle. Utility functions for ceteris paribus preferences. In *AAAI Workshop on Preferences in AI and CP: Symbolic Approaches*, pages 33–38. AAAI Press, 2002.

[20] M. Merayo, M. Núñez, and I. Rodríguez. Formal specification of multi-agent systems by using EUSMs. In *2nd IPM Int. Symposium on Fundamentals of Software Engineering, FSEN'07, LNCS 4767*, pages 318–333. Springer, 2007.

[21] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[22] M. Núñez and I. Rodríguez. `PAMR`: A process algebra for the management of resources in concurrent systems. In *21st IFIP WG 6.1 Int. Conf. on Formal Techniques for Networked and Distributed Systems, FORTE'01*, pages 169–185. Kluwer Academic Publishers, 2001.

[23] M. Núñez, I. Rodríguez, and F. Rubio. Formal specification of multi-agent e-barter systems. *Science of Computer Programming*, 57(2):187–216, 2005.

[24] M. Núñez, I. Rodríguez, and F. Rubio. Specification and testing of autonomous agents in e-commerce systems. *Software Testing, Verification and Reliability*, 15(4):211–233, 2005.

[25] L. Rasmusson and S. Janson. Agents, self-interest and electronic markets. *The Knowledge Engineering Review*, 14(2):143–150, 1999.

[26] T. Sandholm. Agents in electronic commerce: Component technologies for automated negotiation and coalition formation. In *2nd Int. Workshop on Cooperative Information Agents, CIA'98, LNCS 1435*, pages 113–134. Springer, 1998.