Proceedings of the Fifth IASTED International Conference
MODELLING, SIMULATION, AND OPTIMIZATION
August 29-31, 2005, Oranjestad, Aruba

A MULTI-AGENT THEORY FOR SIMULATION

Jacinto Dávila[1], Mayerlin Uzcátegui[1,2] and Kay Tucci[1,2]
[1] CeSiMo. Facultad de Ingeniería
[2] SUMA. Facultad de Ciencias
Universidad de Los Andes
Mérida, 5101. Venezuela
email: {jacinto,maye,kay}@ula.ve

ABSTRACT

This paper discusses a multi-agent simulation theory which is serving as a formal specification to guide the development of a multi-agent simulation platform. We have extended an existing simulation language: GLIDER, with abstractions to model systems where autonomous entities (agents) perceive and act upon their environments. Thus far, we have completed the development of the platform that implements the theory and we are now applying it to the study of multi-agent systems. In particular, an Implementation on Biocomplexity is briefly discussed in the paper.

KEY WORDS:

Multi-agent simulation

1 Introduction

This paper discusses a multi-agent simulation theory which is serving as a formal specification to guide the development of a multi-agent simulation platform.

We have extended an existing simulation language: GLIDER [1], with abstractions to model systems where autonomous entities (agents) perceive and act upon their environments. Those abstractions are based on the simulation theory and provide the semantics for a new family of multi-agent, simulation languages.

A theory is a *"supposition or system of ideas explaining something, esp. one based on general principles independent of the particular things to be explained"* (Oxford Dictionary). Mathematicians have another definition: *"A collection of propositions to illustrate the principles of a subject"* (.ibid). In the more accepted *simulation theory* [2], one finds a general explanation of what a system is, its components and its transitions rules, stated all as a collection of formalized, mathematical propositions. The goal of [2] and the others with that formalization (.ibid), besides supporting the explanations that are expected from a theory, was to provide the developers of *systems simulators* with a specification that says what a simulator must do and how it must behave to simulate a system.

In this paper, we present a multi-agent, system-simulation theory with exactly those purposes and with similar style. This theory has served as the basic specification for the computing simulation platform GALATEA [3, 4, 5] and we also expect it to enrich the foundations for our studies on the problem of structural change, where agents are regarded as important sources of change in the structure of systems.

The presentation of the theory is organized as follows: The section 2 briefly describes the basic simulation framework we are extending: GLIDER. The section 3 offers a review of Feber and Müller's multi-agent theory [6] (hereafter F&M). In the same section, we re-introduce the hierarchy of models of agents proposed by Genesereth and Nilson[7] (hereafter G&N). We combine and extend both hierarchies by describing, following [8], an agent that is both reactive and rational (not accounted for by G&N[7]). In the section 4, we present an abstract machine as the specification of that agent type and briefly explain how this specification has been integrated into the multi-agent simulation theory. We also develop, in section 5, the mathematical description of a multi-agent, rational system which serves as the specification for the simulation platform. Finally, in the section 6, we sketch an example to illustrate the family of multi-agent, programming languages used in GALATEA[3] and whose semantics is provided by the theory.

2 GLIDER: The starting point

In GLIDER a system is conceived as a structured collection of objects that exchange messages. Such message exchange and processing is closely related to the scheduling and occurrence of events as in DEVS [2]. Modelling a system (for simulation purposes) amounts to write a code describing a network of nodes. Those nodes state the behaviours of the objects in the system and how, when and with which they exchange messages. GLIDER offers to the modeller-programmer a set of node types (Gate, Line, Input, Decision, Exit and Resources give its name to the language, but there are others) which the modeller-programmer instantiates to represents the objects he or she wants to simulate.

In GALATEA, we have enriched GLIDER semantics

(and syntax, although a full compiler is still pending) to accommodate the description of agents. Agents correspond to those entities in the modelled system that can perceive their environment, have goals and beliefs and act, according to those beliefs, to achieve those goals, presumably changing the environment in the process.

This enriching of GLIDER required more than an additional set of language elements. We had to extend its current simulation framework to include the behaviour of the new, specialized objects: the agents. To achieve this, we have drawn directly from AI mainstream research on multi-agent theories. In particular, the unified agent architecture described in [8], and the model of situated multi-agent systems presented in [6] are employed in the extended framework. We aim to have a family of languages, supported by a unique computing platform, to model and simulate multi-agent systems. Languages of diverse nature (ranging from procedural, object-oriented, network-oriented to logic-based languages[9, 10]) are, we believe, an important contribution to a multi-disciplinary approach for modelling and simulation, especially when they are mapped against the same explanatory device: a common theory.

Agent type specification Main features
REACTIVE AND RATIONAL $\langle S_a, P_a, Knowledge, Actions, Perception, Memory, Decision\rangle$ Iteratively senses, records, reasons and acts, changing the environment
KNOWLEDGE LEVEL AND DELIBERATIVE $\langle S_a, P_a, Knowledge, Actions, Perception, Memory, Decision\rangle$ Senses, records, reasons and acts, changing the environment
HYSTERECTIC $\langle S_a, P_a, InternalS, Actions, Perception, Decision\rangle$ Senses, records and changes the environment
TROPISTIC $\langle S_a, P_a, Actions, Perception, Effector\rangle$ Senses and changes the environment
OPERATOR OR COMPONENT $\langle S_a, P_a, Actions, Effector\rangle$ Changes the environment

Figure 1. The extended hierarchy of agent types

3 A theory of influences and reactions

In [6] F&M present a theory of multi-agent systems. They describe dynamics systems with a sort of *enhanced* state in which the universe being modelled is described via two types of "state components": *influences* and *environmental variables*. The later correspond to what is commonly known as state variables. Whereas influences are "what come from inside the agents and are attempts to modify a course of events that would have taken place otherwise" [6](p73). The influence concept in the theory of F&M allows to describe the concurrence of events and the transition of states.

F&M declare that their model of action relies on three main concepts:

1. A distinction between influences and reactions, to deal with simultaneous actions.

2. A decomposition of a whole system dynamics, δ, into two parts: the dynamics of the environment (σ, the *environmental state*) and the dynamics of the agents situated in this environment (γ, the set of all their *influences*). Σ is the set of all the possible *environmental states* and Γ is the set of all the possible sets of influences, with $\gamma \in \Gamma$ and $\sigma \in \Sigma$.

3. A description of the different dynamics by abstract state machines, which we use in the specification of the operational semantics of the languages illustrated in section five. Typically, an agent is characterized as tuple of attributes and a set of functions that transform that tuple. Similarly, a whole system is also characterized as a tuple (that includes its agents' tuples) and a set of transformation functions.

In the work presented here, we are taking on F&M's notions of influences and reactions and their proposal to describe dynamical systems via that enhanced state. However, we drop the use of *operators* and modify and extend their theory so that *laws* can be used as influence generators. With this movement, we also establish the base for an operational semantics for our simulation languages.

To illustrate the expressive power of *influences*, F&M adapt a classical work on agent technology to their theory. This work is G&N's hierarchy of agent's architectures [7]. In that work, offered a description of a hierarchy of agent architectures ranging from a non-rational, purely reactive TROPISTIC agent to a rational, DELIBERATIVE agent, via HYSTERETIC agents which are the first type of agent in the hierarchy with an internal, "mental" state. Each type of agent is, again, modelled as a tuple which includes a number of transforming functions. The whole hierarchy from G&N, enhanced with F&M's operators and our REACTIVE AND RATIONAL agent is displayed in figure 1, the elements of this description are:

S_a: The set of states the agent may be in.

P_a: Partial descriptions of the environment.

$Actions$: The set of actions the agent might perform.

$Knowledge$ and $InternalS$: The set of possible internal states the agent may be in.

$Perception$: The agent sensory function.

Effector and *Decision*: These functions encode the agent's action-selection mechanism which decides the action the agent will execute.

$Memory$: This function encodes the agent assimilation mechanism by means of which it updates its knowledge base, with information from the environment (including the feedback obtained when the actions are tried).

4 Our reactive and rational agent

To implement the type of agent at the top of that hierarchy, we describe an agent as a 6-tuple:

$$< P_a, K_a, G_a, Perception_a, Update_a, Planning_a > \quad (1)$$

where P_a and $Perception_a$ are the percept's domain and the perception function which we will not explain for the sake of space. The set K_a and G_a roughly correspond to S_a above. We want to state that a rational agent has a knowledge base, K_a, and a set of goals (or intentions), G_a, that, together, characterize its internal state. $Update_a : \Im \times P_a \times K_a \rightarrow K_a$ takes the place of $Memory_a$ in the memorization mechanism but it now has to ensure that the addition of new information preserves the internal structure of the knowledge base (and its consistency) because K_a is a collection of logical formulae with a well-defined syntax and semantics. Similarly, $Planning_a : \Im \times \Re \times K_a \times G_a \rightarrow G_a \times \Gamma$, substitutes the function $Decision_a$ and, instead of just producing influences from the internal state, the new reasoning function derives new goals and influences, taking into account the previous goals and the knowledge base. Notice that both $Update_a$ and $Planning_a$ introduce an argument (with domain $\Im$, the set of all the possible time points) to indicate the time at which each process (updating and planning) takes place. The introduction of explicit time is another major change in our proposal with respect to F&M (and G&N).

With $Planning_a$, we want to model the process by means of which an agent derives, from a set of high level goals, a set of lower level goals, some of which are actions that can be tried for execution. This view of an agent reducing goals to sub-goals has been studied in [8] in the context of agents in logic programming. This agent model also specifies a way to deal with the problem of bounded rationality. It basically says that an agent must interleave reasoning and acting, so there must exist time (or space) bounds for the reasoning and, then, it may be that the agent acts with no-completely-refined reasons. We mark that limit with a resource (time or space to compute) counter, as will be shown in the following section.

4.1 The behaviour of an agent

Following F&M, we characterize an agent a as a mathematical function $Behaviour_a : \Im \times \Re \times K_a \times G_a \times \Gamma \rightarrow K_a \times G_a \times \Gamma$ that maps the resource limits for reasoning, the agent internal state and the set of influences to a new internal state and a set of influences produced by this agent. Unlike, F&M, our agent internal state contains a knowledge base and a set of goals, as we described above.

This function $Behaviour_a$ is defined as follows ($'$ implies next time):

$$< k'_a, g'_a, \gamma'_a >= Behaviour_a(t, r_a, k_a, g_a, \gamma) \quad (2)$$

where

- t: Current time.
- r_a: Amount of time allocated for reasoning.
- k_a: Agent's knowledge base.
- g_a: Set of agent's goals.
- γ: Past set of influences.
- γ_a: Set of influences that this agent is producing.

The arguments of the function $Behaviour_a$ come as outputs from other functions:

$$k'_a = Update_a(t, Perception_a(\gamma), k_a) \quad (3)$$

$$< \gamma'_a, g'_a > = Planning_a(t, r_a, k'_a, g_a) \quad (4)$$

The $Update_a$ function will simply add the set of percepts observed by agent a into its knowledge base. In particular, in $Perception_a$, $obs(P, t)$ could stand for the fact that the agent observed the property P at time t. The $Planning_a$ function specifies an inference engine which transforms goals g_a into goals g'_a and influences γ'_a, using the rules and factual information in k'_a, starting at time t and taking no more than r_a units of time to do it.

5 A multi-agent rational system (MARS): a specification for a simulation language

Up until now, we have been describing one agent. To specify the behaviour of a multi-agent system, we need to define the functions that account for the evolution of the whole system dynamics. Let us, therefore, define $Evolution : \Im \times S \times \Sigma \times \Gamma \rightarrow \tau$ and $Cycle : S \times \Im \times \Sigma \times \Gamma \rightarrow S \times \Im \times \Sigma \times \Gamma$, the same kind of functions introduced by F&M, but each one with a new argument representing time; where S represents the set of all the possible mental states of all the agents.

$$Evolution(t, < s_1, s_2, .., s_n >, \sigma, \gamma) = Evolution(Cycle(< s_1, s_2, .., s_n >, t, \sigma, \gamma)) \quad (5)$$

$$s_a =< k_a, g_a > \quad (6)$$

Cycle, the function that steps from one global situation into the next, is defined as:

$$< t', < s'_1, s'_2, .., s'_n >, \sigma', \gamma' >= Cycle(< s_1, s_2, .., s_n >, t, \sigma, \gamma) \quad (7)$$

$$< \sigma', \gamma' >= React(\Lambda, \beta, t, \sigma, \gamma \cup_a \gamma_a) \quad (8)$$

in which the newly introduced symbols are explained as follows:

t: Current time.
s_a: Agent a's internal state.
σ: System "static" state: The environmental variables.
γ: Set of previous influences on the environment.
γ_a: Set of Agent a's new influences.
Λ: The laws of the system.
β: Background knowledge that supports the description of the system.

This description of the system must also include the equations:

$$\Lambda = Select(Network, \xi) \quad (9)$$
$$\xi = NextEvent(\gamma) \quad (10)$$
$$t' = TimeOf(\xi) \quad (11)$$
$$\beta = Interpret(InitDecl) \quad (12)$$

where,

$Select$ represents the process that extracts the laws of the system from the code provided by the modeller in the NETWORK section of a GALATEA model (illustrated in the last section within the example).

$NextEvent$ obtains the next event that will occur in the system from the list of influences. $TimeOf$ produces the time of that event. And $Interpret$, like $Select$, represents an interpreter that extracts background knowledge and initial settings of variables from the code that the modeller-programmer creates (also shown with the example).

5.1 The whole description of MARS

On that brief description of an reactive and rational agent and a modified $React$ function, we can build the mathematical description of a system populated by many of such agents. We only need to connect $React$ with the $Behaviour$ function for each agent, as follows:

$$< \sigma', \gamma' >= React(\Lambda, \Lambda, \mathbf{scan}, \beta, t, \sigma, \gamma \cup_a \gamma_a) \quad (13)$$
$$< s'_a, \gamma_a >= Behaviour_a(t, r_a, k_a, g_a, \gamma) \quad (14)$$

where, in turn, s'_a is an abbreviation of $< k'_a, g'_a >$, the knowledge base and goals of agent a. This links the influences from the agents' behaviour to the reaction of the environment and completes the definition of the multi-agent system.

6 An example: Agent Modelling of a Forest Reserve

Probably the best way to introduce the newly extended modelling language is through simple examples. What follows is the basic layout of one GALATEA simulation model of a real natural system. It is, actually, the model of a multi-agent system coupled with a natural dynamics.

The model here described is an outcome of the project Biocomplexity: Integrating Models of Natural and Human Dynamics in Forest Landscapes Across Scales and Cultures (`http://www.geog.unt.edu/biocomplexity`). It aims to model and simulate land use and changes in vegetation cover as a consequence of human actions.

As it has been explained in [11, 12], we have being devising a collection of toy models to cater for 1) the human dynamics, using the set of conceptual tools and data structures provided by GALATEA and 2) the environmental dynamics, by integrating a cellular automaton from the SpaSim [13] library into the actual simulator of a forest reserve. The data structures of GALATEA provide for the representation of the agents' goals, beliefs and observations, and, also, for a very elementary reasoning engine to deduce actions for each agent, according to its circumstances.

The simplified model considers several instances of "settler" agents and a lumber "concessionary" agent. For the sake of space, we will only consider here the behaviours of the first. For a complete account the reader is referred to [11].

```
NETWORK
 LANDSCAPE (A) :: // SpaSim's invocation code
AGENTS
  Settler (AGENT) ::
  GOALS
    if supervised then go_elsewhere;
    if not(occupied_land), not(supervised),
      abandoned_land
    then settle_down_with_strategy_1;
    if not(occupied_land), not(supervised),
      land_is_forest_without_timber
    then settle_down_with_strategy_2;
    if not(occupied_land), not(supervised),
      land_is_forest_with_timber
    then settle_down_with_strategy_3;
    if land_does_not_produce,
      not(occupied_land_next)
    then expand;
  BELIEFS
    to settle_down_with_strategy_1 do move_in;
    to settle_down_with_strategy_2 do move_in,
       cut;
    to settle_down_with_strategy_3 do move_in,
       cut, sale_wood;
INTERFACE
// Code to explain the effects of the agents'
// actions on the environment.
INIT
// Initiation services.
  time_step := 10;
  ACT(LANDSCAPE, 0);
DECL
// Instructions to declare the data structures
// including those based on the SpaSim library

END.
```

Figure 2. Partial view of the Caparo Model in GALATEA

The settler agent rules of behaviour can be put as

shown in figure 2. The settlers are people of limited economical resources that arrive at the reserve aiming to improve their economical status and to get the property of the land that they get to occupy. Initially they dedicated themselves to subsistence agriculture: they just try to maximize the benefits from their occupation of the area, without considering soil exhaustion due to poor management practices, and without much regard for ecological damage. After five years, the land loses its fertility, and the settler must move to another available place (i.e. an area not under government supervision) or expand his farm by deforesting some adjacent land.

Figure 2 partially depicts a GALATEA model of this system[12]. This is the normal layout of a simulation model in GLIDER now enriched with a logic-based description for each agent[9, 10].

As it is, it has been very useful to discuss the behaviour of the human actors in the reserve as part of validating exercises. All this, even though we have not finished implementing a higher level translator for the system and we still cannot provide a structured model like the one in that figure (but we do have a running example in pure Java).

Simulation results are portrayed as graphs (Figure 3) that show the percentage of total forest area by each of the policy scenarios (Agroforestry, Forestry, Hands-off) and maps that show the spatial distribution of land-use types obtained in each of the scenarios at each time step.

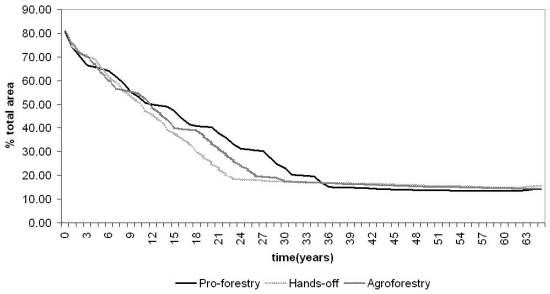

Figure 3. Percentage of total forest area by each of the policy scenarios

Figure 4 shows the final state of the Caparo Forest Reserve for each policy scenario. Our theory allows for modularity by means of a function $Behaviour_a$ for each agent but also a conceptually higher modularity by distinguishing the agents from the natural system of the forest reserve.

7 Conclusions

In this paper, we have described a mathematical theory that state what multi-agent systems are and how they evolve through time. This theory is being used as formal specification to guide the implementation of a multi-agent simulation platform that we have called GALATEA. This is a

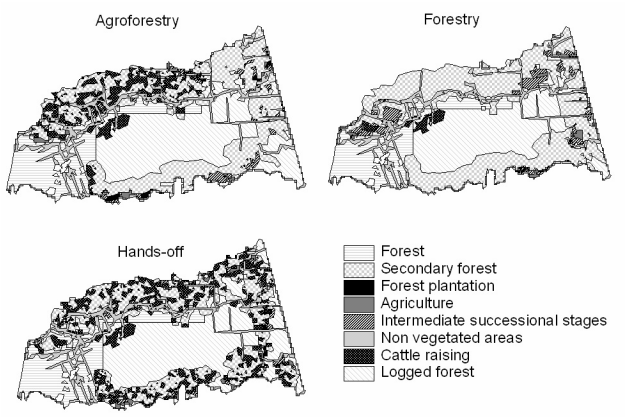

Figure 4. Resulting maps at the end of the simulation for each one of the policy scenarios

multi-language platform: we use an extension to a mature simulation language (GLIDER) to describe "the world" (the environment) in which the agents are embedded (the `NETWORK` section in the example above). And, we also use a set of logic programming languages to specify each agent's goals and beliefs (the `AGENTS` section).

We have completed the development of a platform that implements the theory and we are now applying it to the study of multi-agent systems.

Acknowledgements

We are very grateful to the CESIMO team and our students for many useful discussion. This work has been partially funded by CDCHT-ULA projects I-666-99-02-E and I-667-99-02-B and Fonacit project S1-2000000819. This work was also partially supported by an NSF Biocomplexity in the Environment grant CNH BCS-0216722. We wish to thank the other grant participants for their valuable contributions

References

[1] GLIDER Development Group. *GLIDER Reference Manual, Versión 5.0*. Cesimo & IEAC, Universidad de Los Andes, Mérida, Venezuela, 2000. CESIMO IT-02-00. Available from: `http://afrodita.faces.ula.ve/Glider/`.

[2] Bernard P. Zeigler. *Theory of modelling and simulation*. Interscience. Jhon Wiley& Sons, New York, 1976.

[3] Jacinto A. Dávila and Mayerlin Uzcátegui. Galatea: A multi-agent simulation platform. *AMSE Special Issue 2000. Association for the advancement of Modelling & Simulation techniques in Enterprises*, pages 52–67, 2002. Lion, France.

[4] Jacinto A. Dávila and Kay A. Tucci. Towards a logic-based, multi-agent simulation theory. *AMSE Special Issue 2000. Association for the advancement of Modelling & Simulation techniques in Enterprises*, pages 37–51, 2002. Lion, France.

[5] Jacinto Dávila and Mayerlin Uzcátegui. Gloria: An agent's executable specification. *Collegium Logicum. Kurt Gödel Society*, VIII:35–44, 2004. Vien, Austria.

[6] Jacques Ferber and Jean-Pierre Müller. Influences and reaction: a model of situated multiagent systems. In *ICMAS-96*, pages 72–79, 1996.

[7] Michael R. Genesereth and Nils Nilsson. *Logical foundations of Artificial Intelligence*. Morgan Kauffman Pub., California. USA, 1988.

[8] Robert A. Kowalski and Fariba Sadri. Towards a unified agent architecture that combine rationality with reactivity. In Dino Pedreschi and Carlos Zaniolo, editors, *LID'96 Workshop on Logic in Databases*, San Miniato, Italy, July 1996. Available from: `http://www-lp.doc.ic.ac.uk/UserPages/staff/rak.html`.

[9] Jacinto A. Dávila. Openlog: A logic programming language based on abduction. In *PPDP'99*, Lecture Notes in Computer Science. 1702, Paris, France, 1999. Springer. Available from: `http://citeseer.nj.nec.com/64163.html`.

[10] Jacinto Dávila. Actilog: An agent activation language. In *PADL2003*, LNCS, New Orleans, USA, 2003.

[11] M. Ablan, J. Dávila, N. Moreno, R. Quintero, and M. Uzcátegui. Agent modeling of the caparo forest reserve. In *EUROSIS 2003*, pages 367–372, Naples, Italy, October 2003.

[12] R. Quintero, R. Barros, J. Dávila, N. Moreno, Tonella G., and M. Ablan. A model of the biocomplexity of deforestation in tropical forest: Caparo case study. In C. Pahl, S. Schmidt, and T. Jakeman, editors, *IEMSs 2004*, Osnabrueck, Germany, June 2004. http://www.iemss.org/iemss2004/proceedings.

[13] N. Moreno, M. Ablan, and G. Tonella. Spasim: A software to simulate cellular automata models. In *IEMSs 2002, First Biennial Meeting of the International Environmental Modeling and Software Society*, Lugano, Switzerland, 2002. Available from: `http://mistoy.ing.ula.ve/INVESTIGACION/PROYECTOS/SpaSim/SpaSim/`.