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

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, 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.

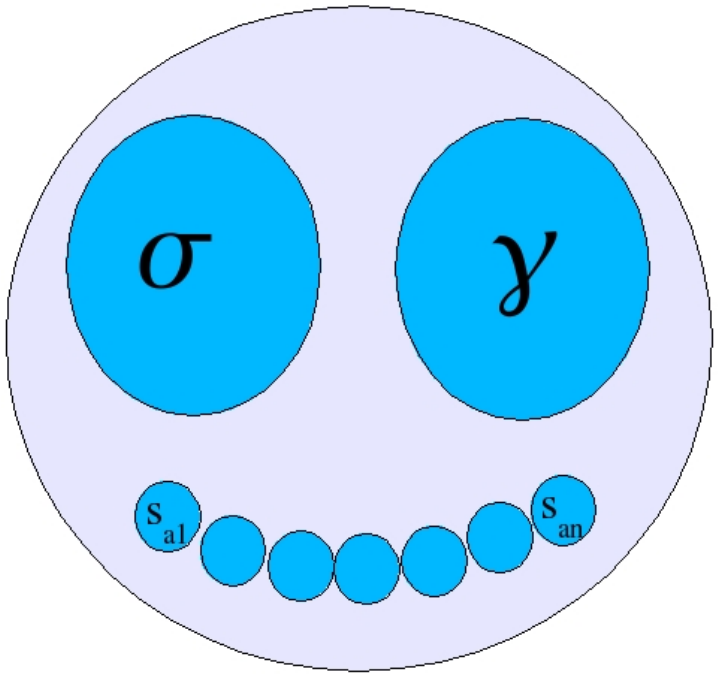

Figure 1: Map of concepts in the theory

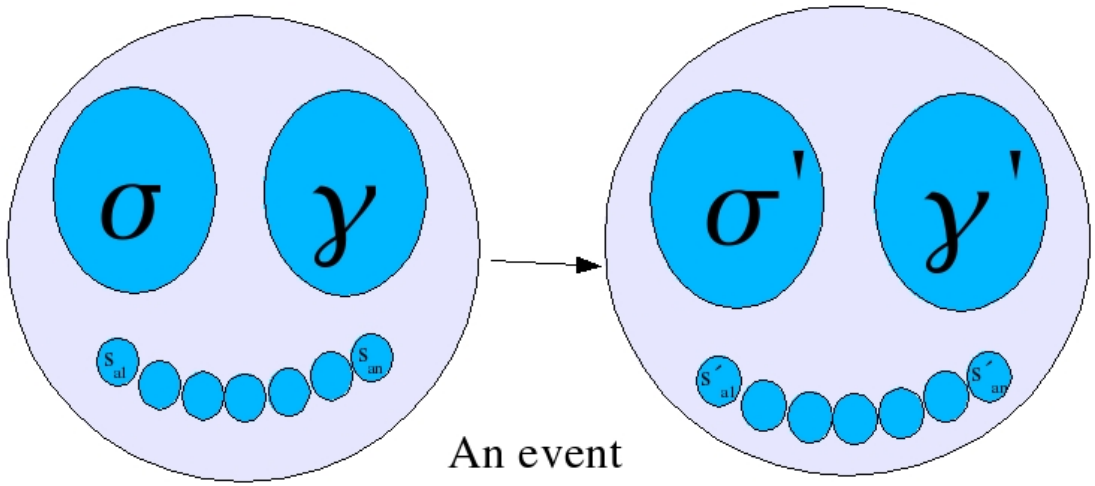

Figure 2: A traditional dynamics with a difference

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 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 presented 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.

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.

We describe an agent as a 6-tuple:

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

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

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:

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

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

$$Evolution(t, < s_1, s_2, .., s_n >, \sigma, \gamma) =$$
$$Evolution(Cycle(< s_1, s_2, .., s_n >, t, \sigma, \gamma)) \quad (5)$$
$$s_a =< k_a, g_a > \quad (6)$$

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:

$$\Lambda = Select(Network, \xi) \quad (9)$$

$$\xi = NextEvent(\gamma) \quad (10)$$

$$t' = TimeOf(\xi) \quad (11)$$

$$\beta = Interpret(InitDecl) \quad (12)$$

$$< \sigma', \gamma' >= React(\Lambda, \Lambda, \mathbf{scan}, \beta, t, \sigma, \gamma \cup_a \gamma_a) \quad (13)$$

$$< s'_a, \gamma_a >= Behaviour_a(t, r_a, k_a, g_a, \gamma) \quad (14)$$

The example 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.

```
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;
...
```

Figure 3: Partial view of the Caparo Model in GALATEA

```
  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 4: Partial view of the Caparo Model in GALATEA

Simulation results are portrayed as graphics (Figure 5) 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.

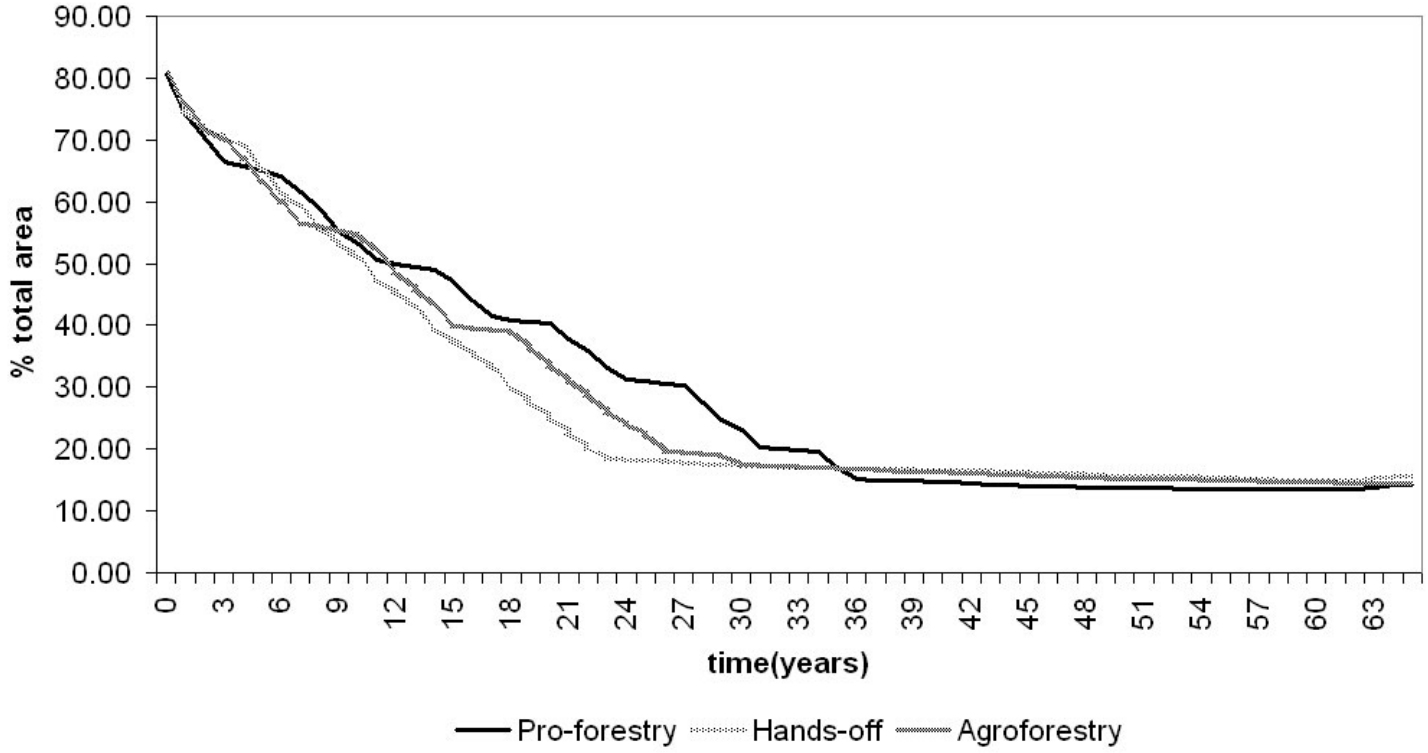

Figure 5: Percentage of total forest area by each of the policy scenarios

Figure 6 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.

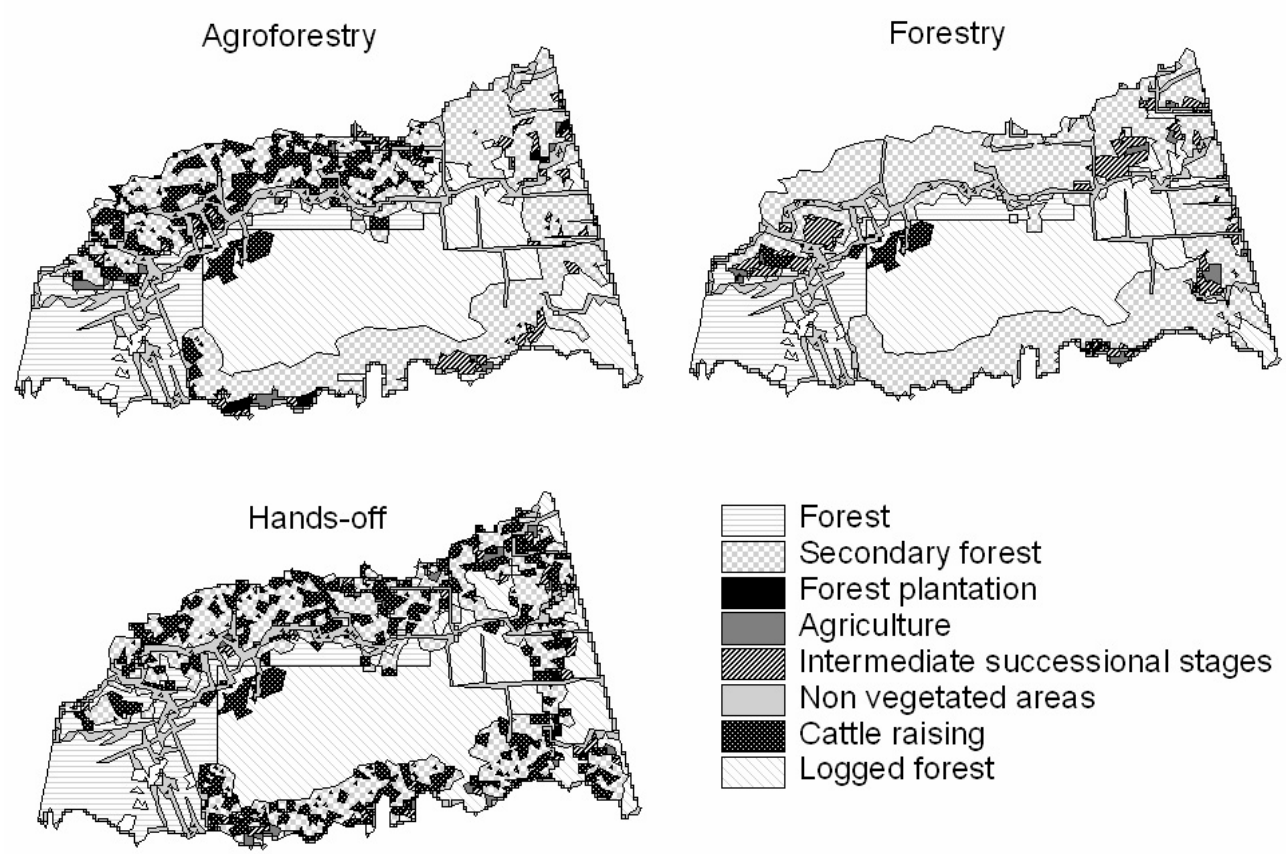

Figure 6: Resulting maps at the end of the simulation for each one of the policy scenarios

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 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/`.