Computational Complexity of a Constraint Model-based Proof of the Envelope of Tendencies in a MAS-based Simulation Model

Oswaldo Terán[*†], Bruce Edmonds[*]

[*]Centre for Policy Modelling, Manchester Metropolitan University,
Aytoun Building, Aytoun Street, Manchester, M1 3GH, UK.
Tel. +44 161 247 6478 *Fax*. +44 161 247 6802
b.edmonds@mmu.ac.uk

[†]Department of Operation Research and Centre for Simulation
and Modelling, Universidad de Los Andes. Venezuela
Tel. +58 274 240 2879
oteran@ula.ve

Abstract. This paper determines the complexity of a constraint model-based proof of the envelope of a tendency in the dynamics of a Multi-Agent-based Simulation model. The proof is performed via a constraint model-based exploration of simulation trajectories using forward inference, by means of which a whole fragment of the simulation model theory is investigated. Such exploration allows for all simulation trajectories defined by a range of parameters of the model and a range of choices of the agents. The paper verifies that the search is *coNP-complete*.

1 Introduction

There is a *need* for studying and *proving (emergent) tendencies* in the *simulation of social systems* (including simulation of organizations). This need has been especially remarkable in those works related with elaborating or testing theories [1, 4, 6]. Such a need has not been satisfied by traditional approaches for exploring the dynamics of simulation models, such as Scenario Analysis and the Monte Carlo method. Neither of these approaches performs exhaustive explorations of simulation trajectories in subspaces of the simulation theory. The explored trajectories are chosen, in the first case, by a domain expert, and in the second case, randomly. Owing to these facts, those approaches cannot be used for proving tendencies in the dynamics of a simulation model - the allowed conclusions are valid either according to the expertise of a domain expert or statistically.

As an alternative to these traditional methods, in previous papers [8-10] a *hierarchy of computational architectures* for searching for and proving tendencies in a Multi Agent System-based (MABS) simulation model has been proposed. The first architecture, that at the higher level, consists of the MABS model where tendencies will be searched for by the modeller. After a tendency is found, at a *second architectural level*, a constraint logic model[1] proof of the envelope of the simulation trajectory is proposed. In those papers, a computational technique for doing this proof

[1] The term 'logical model' means model in the logical sense, which is different to the idea of model in modeling and simulation theory. In the terminology used in this paper, a logical model corresponds to a simulation trajectory.

efficiently is implemented and illustrate by using an example. And, at a third architectural level, a more general proof of the envelope of the found tendency would be implemented by exploring a wider fragment of the simulation theory by using a syntactic driven search. As explained better in [10], *this research contributes in bringing closer the simulation and the logic programming communities*.

The present paper examines the *computational complexity of the procedure implemented in the second architectural level*. First, in the second section, simulation theory formalisms are given. Then, in section three, the idea of envelope is reviewed. Afterwards, in the fourth section, the logic-based exploration of simulation trajectories implemented for proving the envelope of tendencies in simulation models is described. Then, in the fifth section, the computational complexity of such exploration is established. And finally, in section six, some conclusions are drawn.

2 Simulation Theory

2.1 Formal Representation of a System (according to Zeigler [13-15])

The idea is to provide a formal description of a target system or of a model of it. Originally, Zeigler's [15] basic formalism is intended to describe simulation models whose structure is fixed, which are common in modelling and simulation of industrial systems, for instance, of queue systems. These ideas can be easily extended to MABS models.

Aspects of Zeigler's formalism are represented in Figure 1. The situation is shown for two simulation (time) steps: from the former $time_1$ to the latter $time_2$. An input value x, an output value y, an internal state q, at each time step, and the transition from the first time step to the next one, are illustrated. The output is a function defined by a modeller or by an observer. The domain of the output function is the internal state of the system. It is supposed that there is a function λ generating the output y_i from the state of the system q_i, and a function δ defining the new state of the system q_{i+1} as a function of a previous state q_i and the input x_{i+1}. Time is considered as an independent variable.

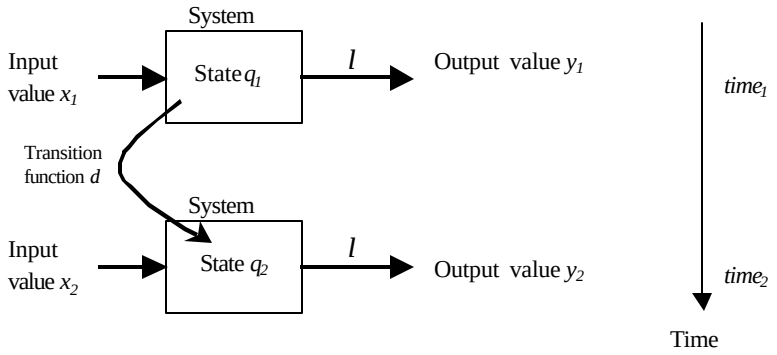

Figure 1. Basic notions in Zeigler's formalism: input values x_1, x_2; system states q_1, q_2; output values y_1, y_2; output function λ; and transition function δ

More formally, Zeigler's notation for a system S is: $S = < T, X, W, Q, Y, d, l >$. Where:

T: Time base (T = Reals or T = Integers).

X: Input *value* set (each input is a sequence of values)

W: Input *segment* set, subset of (X, T), $W = \{w / w: < 0, t > \to X, t \in T\}$.

Q: State set.

d: State transition function. $d: Q \times W \to Q$.

l: Output function, $l: Q \to Y$.

Y: Output *value* set. (there should exist a set of output segments $\{r / r: < 0, t > \to Y, t \in T\}$)

The set $< T, X, W, Q, Y, d, l >$ is called the *system structure*. The subset $< X, W, Q, Y >$ gives the *static structure* and the rest of the specification, d and l, the *dynamic structure*. The dynamic structure consists of the laws for changing the state of the model (usually having time as the independent variable). Notice that there are no laws of change either for the static or for the dynamic structure, that is, this specification does not consider systems with variable structure.

2.2 Zeigler's Levels of System Specification

The idea is to have a hierarchy of descriptions of systems by increasing levels of elaboration in the sense that the higher the level, the more detail the system specification offers. This is helpful, for instance, for comparing model descriptions and for determining the degree of specification a model has.

Level 0: Observation Frame, $S = < T, X, Y >$. The sets of inputs (X), outputs (Y), and the time base (T) are distinguished but it is not known how the two first interrelate.

Level 1: Input/0utput (I/O) Relation Observation (IORO), $S = < T, X, W, Y, R>$, $(R \subseteq W \times (Y,T)$, *where* $(w,r) \in R \to$ *(implies)* $(dom(w) = dom (r))$. W is the set of inputs and R is a relation between the input and output sets. Still, it is not possible to differentiate among the different outputs associated with one input and *vice versa*.

Level 2: I/O Function Observation (IOFO), $S = < T, X, W, Y, F >$, $(f \in F \to$ *(implies)* $f \subseteq W \times (Y,T)$ is a function, and if $f = (w,r)$ then $dom (w) = dom (r))$. At this level, there is a function from the input to the output set which permits to differentiate among the different outputs associated with an input. According to Zeigler's theory, this is granted by the knowledge of the initial state. Still, there is no knowledge about the states of the system along time (apart from the initial state).

Level 3: I/O System Specification, $S = < T, X, W, Q, Y, d, l >$. At this stage, the state set (Q), the transition (d) and output (l) functions are also known. Nevertheless, there is no distinction of the components of a system.

Level 4: Multicomponent System Specification. Each component is defined as a subsystem and how components work together is indicated. At this level, Zeigler presents two slightly different specifications: the nonmodular coupled multicomponent system and the modular coupled network of systems. The difference between this two specifications is put in the following terms by Zeigler *et al.* [13, pp. 125]: "Whereas in networks of system specification individual component systems are coupled by connecting their input and output interfaces in a modular way, components of [nonmodular] multicomponent systems influence each other directly through their state transition functions". Thus, the difference is only in the way the interaction among subsystems is defined: via an interface and in a modular way in a

network of system specification, or directly by their state transition functions in a nonmodular way in a nonmodular multicomponent system specification. Both cases are useful to describe Multi-Agent Systems (MAS). In particular, the description of a nonmodular coupled multicomponent system can be summarised as follows:

MS (a multicomponent system) = < *T, X, W, Y, D, {M_d}* >. Where: *T, X, W, Y,* are as defined above. *D* is the set of component references, and, the set *{M_d}* gives the specification for each *d* in *D*. For all *d* ∈ *D*: M_d = < Q_d , E_d , I_d , d_d , l_d >, is a component, where:

Q_d is the set of states of the component *d*,

E_d Í *D* is the set of components influencing *d*,

I_d Í *D* is the set of components influenced by *d*,

$d_d : \times_{i \in I_d} Q_i \times \Omega \rightarrow \times_{j \in E_d} Q_j$ is the state transition function of *d*,

$l_d : \times_{i \in I_d} Q_i \times \Omega \rightarrow Y$ is the output function of *d*.

2.3 MAS-based Simulation

A MAS consists basically of a hierarchy of agents. Each agent can be described as a system by using Zeigler's formalism: agents without sub-agents will be described at level 3 while containers will be described at level 4. In general, a multicomponent system specification is useful to describe any MABS model.

A sort of change allowed in MAS but no explicitly considered in Zeigler's formalism is structural change. Structural change happens, for instance, when agents are eliminated or when new agents are introduced in a simulation trajectory. For this to happen, laws of structural change should be given, *e.g.*, conditions for introducing a new agent, as well as specifications about the structure and initial state of the new agent, and about how it will interact with other agents. Structural change brings in changes at the system specification at level 4 in a simulation trajectory. Formalisms for simulations involving structural change are presented by Barros [2-3].

3 Enveloping Tendencies in a Simulation Model

The idea is to enclose or encircle in some sense several instances of the simulation output as an alternative to statistical summaries. An envelope will be, in some sense, a contour of several instances of the simulation output. It does not seem convenient to use the strong concept of envelope managed in mathematics. Rather, an envelope will be chosen considering the trade-off between practical usefulness (for a modeller) and precision.

By precision we mean how close the concept is to the ideal mathematical notion of a tangent curve/surface. For example, in mathematics given a family of functions (which might correspond to several runs of a simulation output *Y*, got, for example by varying some parameters), let us say $y_j(t)$, $j = 1,2, .,.k$, an envelope of this family will be a curve *E* being tangent at each point to a member of that family (let us say it is tangent to y_j *at* $(t_i, y_j(t_i))$. Obviously, in simulation only as a casualty the instances of a simulation output could conform a family of curves having such a kind of tangent - so we have to use a more relaxed concept of envelope.

Consider the case of *enveloping a single simulation output, Y.* Each trajectory will generate a sequence of real values over time, *Y*. Calling y_{ij} the output value at time instant *i* for trajectory *j*, an envelope might consist of two sequences of values over

time: E_{upper} and E_{lower}, which in some sense cover all trajectories. The value of E_{upper} at time instant i must be greater than or equal to y_{ij} for all j, and E_{lower} at time instant i must be lower than or equal to y_{ij} for all j. That is, the envelope would be given by two sequences of values over time, where for each time instant all values generated by the simulation trajectories are enclosed by the two values given by these two value sets. More precisely, if the outputs y_{ij} are given for the simulation trajectories $j = 1,\ldots,k$, and for the time instants $t_i = 1,2,\ldots l$, then the envelope of interest at t_i might be defined by the two values: $E_{upper,i} = max_j(y_{ij})$ and $E_{lower,i} = min_j(y_{ij})$.

Alternatively, first an approximating function, f, for the output value set Y that each trajectory generates might be elaborated; then, the instances of these functions (one function for each trajectory) might be enveloped.

To illustrate the practical use of analysing the simulation outputs by enveloping the output think about the simulation of a chaotic system, where the envelope might help in defining the area where a chaotic attractor is placed.

4 Proving Tendencies Via a Model-based Exploration of Simulation Trajectories in a MAS-based Simulation Model

4.1 Logical Model-Constrained Exploration of Simulation Trajectories

A simulation – *e.g.*, an event-driven, a finite differences, or a MABS - can be seen as a partial logical model defined by the sequence of states (of the set Q, in Zeigler's formalism) generated by the transition function d for a system in the third or fourth level of specification defined in Zeigler's formalism. Usually, in a trajectory only a partial set of all the facts of the logical model corresponding to the trajectory are explicitly generated. This partial set consists of those facts that are relevant, either because they are required for the modeller as outputs or because they are necessary to generate the simulation transition states. The remaining facts (although knowable) are left as unknown.

There are different methods to specify a theory in a language. One method commonly employed in logic consists in using a set of formulas of the language to represent the axioms of the theory. In a declarative program a simulation model is specified via a database, a rulebase (which defines the transition function) and the underlying logic of the program. *Potential trajectories are defined via non-deterministic factors* of the simulation. These factors are usually represented by the parameters of the model and the choices of the processes.

The interest in this paper is in exploring the simulation trajectories corresponding to a range of parameters of the model and choices of the agents. The transition function, d, either for each agent (or other simulation process) or for the whole simulation model is *nondeterministic*.

The idea in previous studies [8-10] has been to analyse the *emergence of tendencies* in a simulation by exploring a subspace of the space of trajectories. For this exploration, a logical model-based constraint search was implemented where constraints standed for selected parameters and choices. The exploration allows a modeller to explore that fragment of the simulation theory defined by the selected range of parameters and choices (see Figure 2). Consequently, the resulting conclusions and proofs will be valid over that fragment of the theory and, under appropriate justifications, they could be extrapolated to the whole simulation theory or to a corresponding real model.

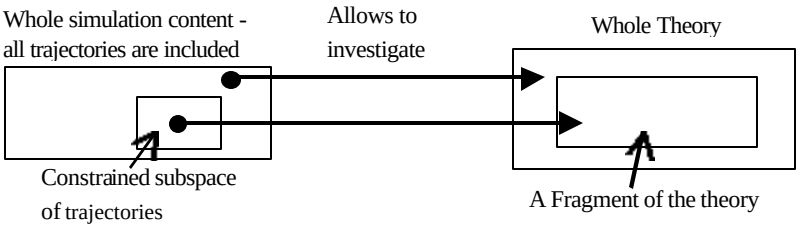

Figure 2. Theory given by simulation trajectories

4.2 Logical Model Exploration for Proving the Necessity of a Tendency

The idea is to generalise about tendencies going from the observation of individual trajectories to observation of a group of trajectories generated for certain parameters and choices. In particular, it is intended to know if a certain tendency is necessary or contingent in the explored trajectories. We understand a simulation trajectory as a logical model embedded in a simulation program (a 'possible world' in semantic terms) and involving trajectories of entities (e.g., agents) inside the simulation and, hence, different from trajectories of these entities. It is a cross-product of all settings of the structure of the simulation model and all processes (e.g., agents' choices) into one path through a high-dimensional space (see Figure 3).

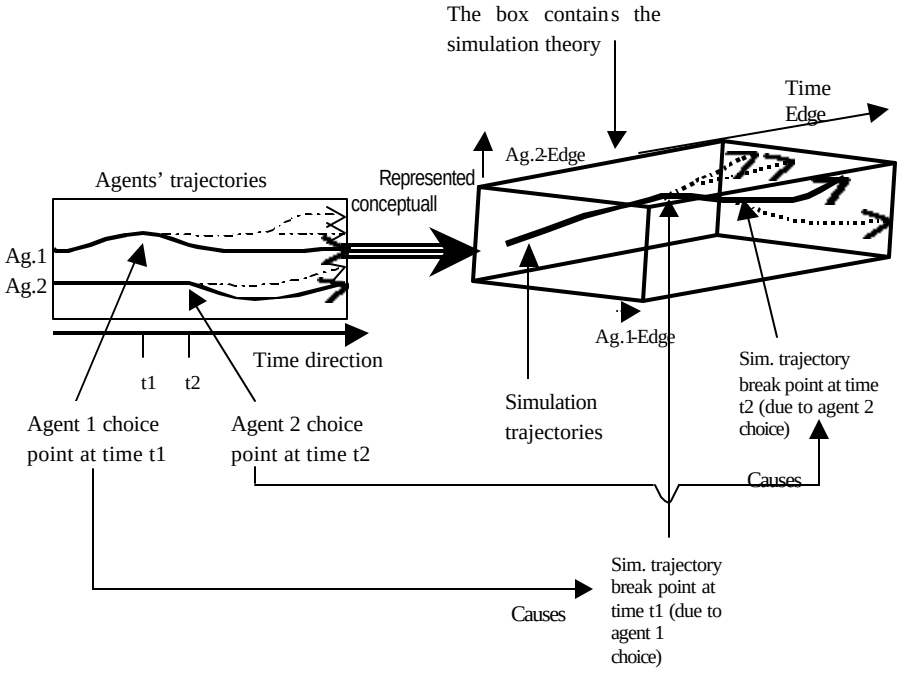

Figure 3. Representation of a simulation theory in terms of the simulation trajectories, and of these in terms of agents' choices (for a single parameter-setting and two agents)

As said above, the transition function of an agent, other process, or the whole simulation model is nondeterministic as the simulation model can assume alternative parameters and as the agents and/or process can select alternative choices.

The character of the search in our models has been predominantly logical model, constraint, forward-chaining, and clausal ordered. *A logical model is generated for each combination of parameters and choices and for a finite iteration number*, *n*. Given a combination of parameters and choices a deterministic transition function may be defined to generate the logical model by iterating from the initial state until the iteration number, *n*, is reached.

In the suggested exploration, first, each combination of parameters provides a different structure of the simulation model (see Figure 4). Following, 'paths' representing trajectories are generated for each structure. Then, while the simulation is going on, choices produce branch points where alternative settings for each choice turn out into a different simulation trajectory.

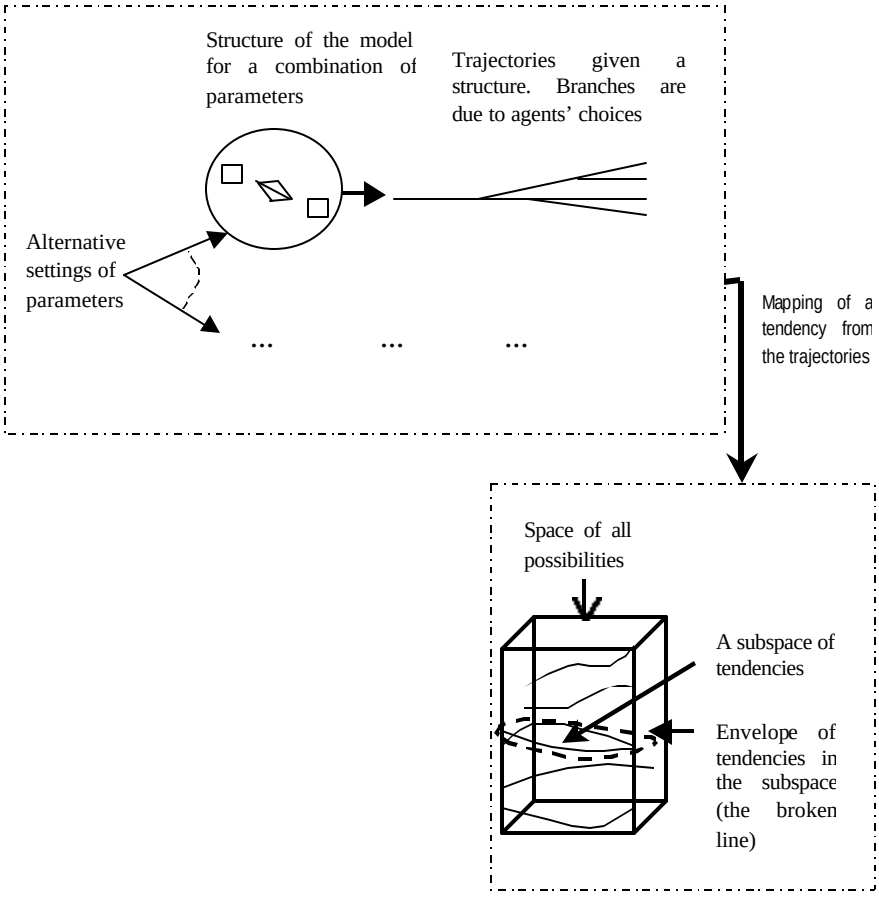

Figure 4. A model constraint-based exploration of the dynamics of a simulation model

This exhaustive constraint-based search over a range of possible trajectories makes it possible to establish the necessity of postulated emergent tendencies. Following a procedure similar to that used in theorem-proving [5,12], a subset of the possible simulation parameterizations, agent choices and iteration number is specified, the target emergent tendencies are prearranged in the form of negative constraints, and an automatic search over the possible trajectories is performed.

Tendencies are shown to be necessary for the finite number of iterations *n*, with respect to the range of parameterisations and non-deterministic choices, by first finding a possible trajectory without the negative constraint to show the rules are consistent and then showing that all possible trajectories violate the negation of the hypothetical tendency when this is added as a further constraint. This is equivalent to showing that all possible tendencies obey the positive form of the constraint, i.e., that the positive form is true for all tendencies.

5 Determining the Complexity of a Constraint Model-based Proof of the Envelope of a Tendency

The *aim* of this section is to demonstrate that the exploration of trajectories proposed in the previous section applied over an infinite (theoretically) number of iterations is *coNP-complete*. To make clearer the exposition, the simulation exploration subject of this paper will be called the *target problem*. As is usual for this sort of verification, two steps are followed:

First, it will be proved that the target problem is in coNP by expressing it as a binary tree of depth *n*.

Second, it will be proved that the problem is also *coNP-complete* by translating the *validity (of Boolean expressions) problem*, a typical *coNP-complete* problem, into the target problem.

For the first part of the proof the aim is to form a Boolean quantified expression:

$$\forall x_1 \forall x_2 \forall x_3 \forall x_4 \forall x_5 \ldots \forall x_{2n-1} \forall x_{2n} (F) \qquad (1)$$

where F is the formula to be evaluated over the variables $x_1 \ldots x_{2n}$ and n is the number of iterations.

The deterministic part in the state transition of the simulation will be called environment's actions, and it will be assumed that it corresponds to the impair variables in *(1)*. It captures changes not associated with *agents' choices* – and basically that part of the simulation where "agents are placed". Consequently, there is only one alternative action for the *impair* variables[2]. The *even* variables correspond to the agents' choices (which are going to be called *agents' actions*). More precisely, for iteration i, $i = 1, 2, \ldots, n$, there are two subsets of variables: $\{x_{2i-1}\}$ and $\{x_{2i}\}$, where $\{x_{2i-1}\}$ is used to represent the environment actions and $\{x_{2i}\}$ stands for the agent's actions. Thus, a whole simulation path or *simulation trajectory is represented* by a concatenation of branches, where each branch corresponds to a unique assignment of values to each variable in the whole set $\{x_i\}$.

Finally, F will be the question: whether the searched *tendency* has occurred in a simulation trajectory. The whole expression *(1)* is true if for all possible assignments of values to the variables the tendency occurs. As each particular assignment of values to the whole set of quantified variables corresponds to a simulation trajectory,

[2] Environment's actions are assumed deterministic. The results of this paper are easily extendible to the case where the environment's actions are non-deterministic.

the proof is successful if this expression is valid for all possible values the quantified variables can take! (e.g., for all possible agents' choices[3]).

To check if the proof is successful, a Boolean circuit, where an *AND* gate stands for the " quantifier, is written (see Figure 5). A leaf in this circuit is evaluated to *true* if the tendency is found in the corresponding simulation path and to *false* otherwise. The whole circuit will be *true* if and only if the tendency appears in all simulation paths. Hence, the proof is successful if and only if the circuit is true (e.g., the tendency is found in *all* paths).

These two expressions of the problem (that is, the Boolean circuit shown in figure 5 and the expression of equation *(1)*) are sufficient to prove that the target problem is *coNP*.

The next task is to prove that the problem is *coNP-complete*. It is easy to see the similarities between the target problem and the validity of a Boolean expression. A Boolean expression is an expression: (a) x, where x is a Boolean variable (variable that takes the values True and False), (b) $Ø f$, where $Ø$ is the *logical not*, and f is a Boolean expression c) $f_1 Ú f_2$, where f_1 and f_2 are Boolean expressions and $Ú$ is the logical symbol *or* (d) $f_1 Ù f_2$, where f_1 and f_2 are Boolean expressions and $Ù$ is the logical symbol *and*. Validity of a Boolean expression f, consists in determining

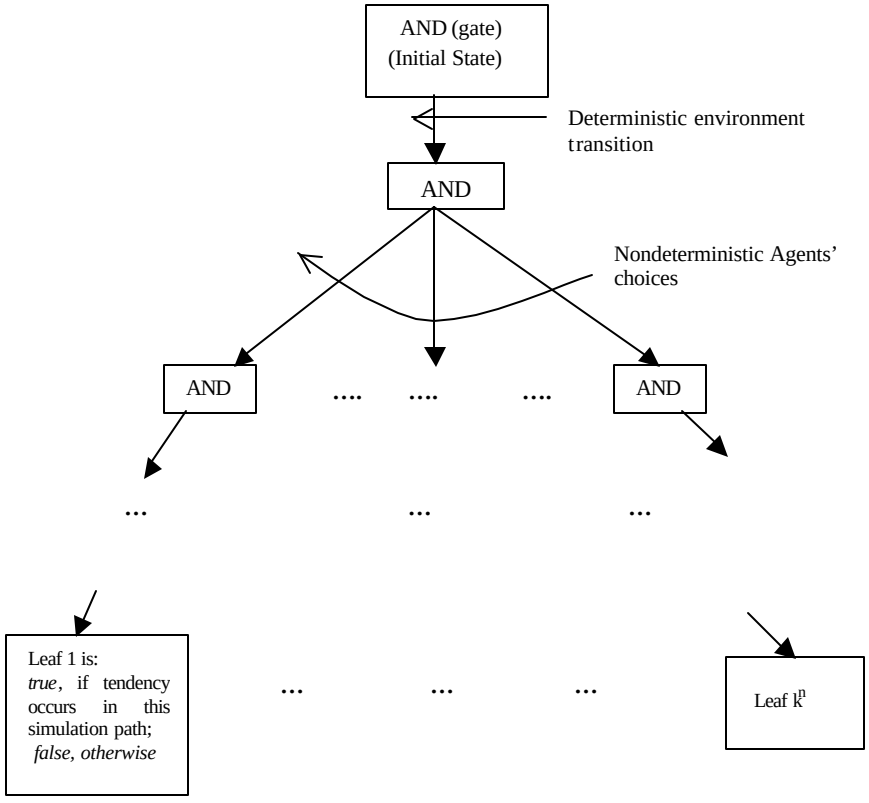

Figure 5. Boolean circuit for the target problem

[3] And, for all environment's actions, in case of a nondeterministic environment.

whether the Boolean expression f is valid under all truth assignments (interpretations). If f is not a valid formula, it can be disqualified by exhibiting a truth assignment that does not satisfy it.

We may evaluate a Boolean expression by using a Boolean circuit similar to that given in figure 5 (see figure 6). A first variable is chosen from the Boolean expression f and represented by the first node, and then two branches are generated from this node: one the case the variable is given the *false* value and the other for the case the variable is given the *true* value. Then a node is aggregated to each of these branches representing a second selected variable, and two branches from each of these new nodes will represent the *true* and *false* value assignments to this second variable. Imagine that this procedure is continued until all variables in the expression f are considered. A leaf of this tree (*i.e.*, a path) will be evaluated to true if the Boolean expression f is true for the particular (an unique) value assignments the variables have in that path of the Boolean tree. Consequently, the expression f is valid iff all leaves of the Boolean tree have been evaluated to true.

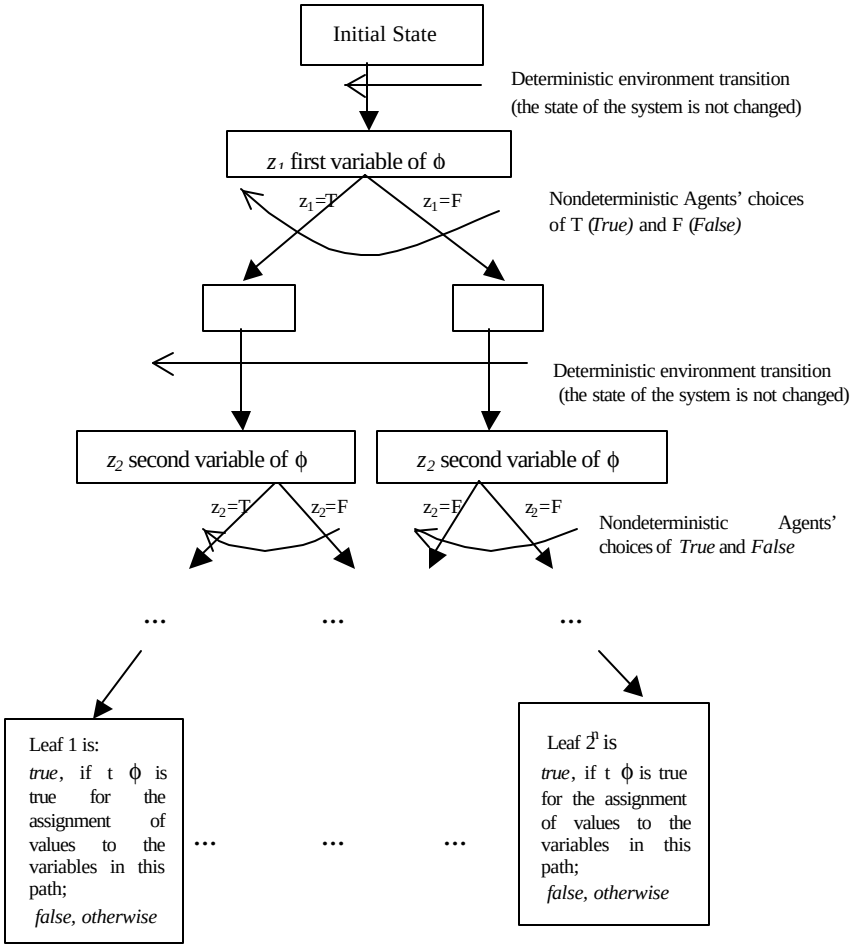

Figure 6. Boolean circuit for the validity problem

This tree corresponds to a target problem, where:

a. The number of iterations, n, corresponds to the number of variables in the Boolean expression f,
b. The environment decisions are not considered (do not change the state of the system),
c. The agents have only two nondeterministic choices: true or false (corresponding to the two possible assignment of values a Boolean variable can be given),
d. The question: is the Boolean expression f true for the assignment of values the variables hold in a certain path?, corresponds (in the translation of the validity problem into the target problem) to the question: does the tendency appears in the corresponding path where agents take decisions in accordance to the assignment of values to the variables?,
e. The tendency appears in a simulation path if the expression f is true for the assignment of values to the variables in accordance to the decisions of the agents in that path.
f. Finally, the expression f is valid iff the tendency appears in all simulation paths.

Thus, the output of the *validity* problem has been reduced to the target problem: The *validity of a* Boolean expression f can be checked simulating the equivalent MABS problem. Therefore, the target problem is *coNP-complete*.

6 Conclusion

This paper has verified that the complexity of a constraint model based exploration of simulation trajectories for proving (the envelope of) tendencies in the dynamics of a MABS model is *coNP-complete*.

Proving the envelope of tendencies in simulation outputs is an alternative to traditional methods used for examining simulation outputs, such as scenario analysis and Monte Carlo techniques. The former allows elaborating more general conclusions than the latter.

As explained better in [10], constraint exploration of simulation trajectories brings closer the simulation and the logic programming communities. This paper contributes in making clearer a property of a constraint exploration of simulation trajectories, namely its complexity, an area of high interest to these two communities.

Acknowledgements. The research reported here was funded by the CDCHT (the Council for Scientific, Humanistic and Technological Development) of the Universidad de Los Andes, Venezuela, under project I-524-AA, by CONICIT (the Venezuelan Governmental Organisation for promoting Science), and by the Faculty of Management and Business, Manchester Metropolitan University.

References

1. Axtell, R., R. Axelrod, J. M. Epstein, and M. D. Cohen, "Aligning Simulation Models: A Case Study and Results", *Computational Mathematical Organization Theory*, 1(2), pp. 123-141, 1996.

2. Barros, F.J., "Modeling Formalisms for Dynamic Structure Systems", *ACM Transactions on Modeling and Computer Simulation*, 7(4) (1997), pp. 501-515.
3. Barros, F.J., *Modeling and Simulation of Dynamic Structure Discrete Event Systems: A General Systems Theory Approach*, Ph.D. Dissertation, Department of Informatics Engineering, University of Coimbra, 1996.
4. Carley K., M. Prietula, and Z. Lin, "Design Versus Cognition: The Interaction of Agent Cognition and Organizational Design on Organizational Performance", *Journal of Artificial Societies and Social Simuation* 1(3), 1998 (accessible at: http://www.soc.surrey.ac.uk/JASSS/1/3/4.html).
5. Loveland, D. W., *Automated Theorem-proving: A Logical Basis*, North-Holland Pub., Amsterdam, 1978.
6. Moss, S., "Social Simulation Models and Reality: Three Approaches", *MAB's 98: Multi-agent Systems and Agent-Based Simulation*, Paris, 1998 (accessible at http://www.cpm.mmu.ac.uk/cpmrep35.html).
7. Papadimitriou, Christos, *Computational Complexity*, Addison-Wesley Publishing Company, California, USA, 1994.
8. Terán Oswaldo, Bruce Edmonds and Steve Wallis, "Mapping the Envelope of Social Simulation Trajectories", MABS2000 @ ICMAS-2000: The Second Workshop on Multi Agent Based Simulation, Boston, July 9, 2000. Published in: Moss, Scott and Paul Davidsson (Editors), *Multi Agent Based Simulation (MABS-2000), Lecture Notes in Artificial Intelligence, Vol. 1979*, Springer Verlag, Berlin
9. Terán Oswaldo, Bruce Edmonds and Steve Wallis, **"**Determining the Envelope of Emergent Agent Behaviour via Architectural Transformation", ATAL-2000: The Seventh International Workshop on Agent Theories, Architectures, and Languages, Boston, July 7-9, 2000. Published in: Castelfranchi, C. and Y. Lesperance (Editors), *Intelligent Agents VII. Agent Theories, Architectures, and Languages. Lecture Notes in Artificial Intelligence*, *Vol. 1986*, Springer-Verlag, Berlin
10. Terán Oswaldo, Bruce Edmonds and Steve Wallis, "Constraint Exploration and Envelope of Simulation Trajectories", First Workshop on Rule-Based Constraint Reasoning and Programming at the First International Conference on Computational Logic (CL2000), July 24-28, 2000, Imperial College, London, UK (this paper is accessible at: http://www.pst.informatik.uni-muenchen.de/personen/fruehwir/cl2000r.html; and at: http://arXiv.org/abs/cs/0007001)
11. Woolridge, Mike "The Computational Complexity of Agent Design Problems", in *Proceedings Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, Boston, MA, USA, July 10-12, 2000, pp. 341-348.
12. Wos, L., *Automated Reasoning: Introduction and Applications*, Prentice Hall, London, 1984.
13. Zeigler, Bernard, Herbert Praehofer, and Tag Gon Kim, *Theory of Modelling and Simulation*, Academic Press, San Diego, CA, USA, 2000.
14. Zeigler, B., *Multifaceted Modelling and Discrete Event Simulation*, Academic Press, London, UK, 1984.
15. Zeigler, B., *Theory of Modelling and Simulation*, Robert E. Krieger Publishing Company, Malabar, FL, USA, 1976.