EXTENDING X-MACHINES TO SUPPORT REPRESENTATION OF SPATIAL 2-D AGENTS

Isidora Petreska[1], Petros Kefalas[2], Marian Gheorghe[3] and I. Stamatopoulou[2]
[1]*South East European Research Centre (SEERC), 24 Proxenou Koromila Str., Thessaloniki 54622, Greece*
[2]*CITY College, International Faculty of the University of Sheffield, 3 Leontos Sofou Str., Thessaloniki 54626, Greece*
[3]*University of Sheffield, Dept. of Computer Science Regent Court, 211 Portobello Str., Sheffield S1 4DP, UK*

Keywords: Formal modelling, X-machines, Spatial agents.

Abstract: Starting with the notion of modelling biologically inspired agents, this paper focuses on their spatial characteristics. It will be demonstrated that one of the most prominent formalisms in modelling the behaviour of biological colonies, X-machines, cannot provide a neat and effective way to modelling spatial agents (i.e. agents distributed and move through a physical space). We introduce a X-machines variation that besides facilitating formal modelling, will provide grounds towards visual animation of these systems. This approach resulted into a novel progression, Spatial X-machines, without retracting the legacy characteristics of X-machines such as testing and verification strategies. Unlike other formalisms that go behind the concept of treating the agent's behaviour as one uniform component, Spatial X-machines tend to draw a separation between different types of agent's behaviour.

1 INTRODUCTION

Formal modelling is considered as one of the most essential stages in Multi-agent system (MAS) development and it can be carried out with many different methods and techniques (Kefalas et al., 2002). There are varieties of formal methods in agent-oriented engineering, and a number of approaches towards modelling biological phenomena have been developed (Kefalas et al., 2002). X-machines (XM) and Communicating X-machines (CXM) are targeted into representing the behaviour of biological colonies, which in turn can be characterised as spatial agents. Spatial agents can be defined as collections of agents distributed and move through a physical space. They have incomplete knowledge of the environment and can change their direction and position within the environment (move in an n-dimensional space). When it comes to data modelling of spatial agents, there are three key features in their development (Y. Bdard and Caron, 1992):

- Conceptual data model, which acts as a representation of the reality as it is, defined by the users;
- Logical data model, which defines how the conceptual model will be implemented; and
- The physical data model or the computer code of the application.

Targeting the conceptual data model, there are different approaches for modelling spatial phenomena of biological systems: process algebra can be applied to develop a calculus of processes that could describe the spatial geometric transformations (Cardelli and Gardner, 2010). Membrane computing can also be utilised by introducing geometric information (Romero-Campero et al., 2009) or population P systems (Petreska and Kefalas, 2011). Yet another agent-based approach is the intracellular NF-κB signalling pathway for modelling spatial information in predictive complex biological systems (Pogson et al., 2008). However, the combination of biological agents and spatial data modelling still remains an active research field. With this work we focus on modelling spatial agents with the XM approach.

On the other hand, visual animation as an informal verification technique, helps in discovering the flaws of the formally unverifiable properties within a biologically inspired systems (such as their position or diction with respect to the environment). We discuss that attempts to formally verify such properties would result into a combinatorial explosion. Moreover, visual animation provides means to facilitate the communication gap between the formal experts and the

Petreska I., Kefalas P., Gheorghe M. and Stamatopoulou I..
EXTENDING X-MACHINES TO SUPPORT REPRESENTATION OF SPATIAL 2-D AGENTS.
DOI: 10.5220/0003744600540061
In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence* (ICAART-2012), pages 54-61
ISBN: 978-989-8425-96-6

scientists (which in turn have no formal background) by providing an immediate feedback understandable to both of the teams. And finally, visual animation can often lead to detecting some emergent behaviour within a system. Therefore, this work opens the question about how to automatically visualise a given XM model.

Starting with a detailed discussion on XMs, later extended by a case study (Sec. 2), we demonstrate the drawbacks of XMs when it comes to modelling the spatial characteristic of an agent. This provides grounds to introducing a formal variation called Spatial X-machines (^{sp}XMs, Sec. 3). This structure will be carefully deliberated, followed by discussion about their support towards verification and validation (Sec. 4). And finally, the paper will be concluded with a discussion on whether ^{sp}XMs provide means to support visual animation strategies.

2 BACKGROUND ON X-MACHINES: FROM DEFINITION TO PRACTICE

An XM resembles a Final State Machine (FSM) with the power of being more expressive (Kefalas et al., 2002). It is achieved due to the addition of a memory, and functions operating on the inputs and memory values. Considering stream XMs (a representative class of XMs), the memory is a typed tuple of values which appears to enhance the modelling of complex data structures. The control, on the other hand, can be visualised by utilising a diagrammatic approach. Thus stream XMs have the ability of modelling both the data (held in the memory) and the control. The processing of the data is modelled by transitions between states, represented with functions. A function receives the memory values together with an input, performs changes on these memory values and produces an output. Based on the current state, the memory changes after an application of a function and the output of that function, the stream XM evaluates the next state.

Formally, a stream XM can be described as an 8-tuple XM = (Σ, Γ, Q, M, Φ, F, q_0, m_0), such that (Kefalas and Kapeti, 2000; Kefalas et al., 2003b):

- Σ and Γ are input and output sets of symbols;
- Q is a finite set of states;
- M is an n-tuple called memory;
- Φ is a finite set of partial functions that map an input and a memory state to an output and a new memory state, ϕ: $\Sigma \times M \rightarrow \Gamma \times M$;
- F is a function that determines the next state, given a state and a function from the type Φ, F: $Q \times \Phi \rightarrow Q$; and
- q_0 and m_0 are the initial state and memory respectively.

With the focus on the practical development of communicating systems, the output of an X-machine function can become input to a function of another X-machine. This way a structure known as Communicating X-machine (CXM) is being formed, providing a way to deal with agents communication (Kefalas et al., 2003a; Kefalas, 2002).

2.1 Case Study: A Foraging Agent

Let us consider the following example of an agent that randomly moves in 2-D space, picks up an object it encounters and carries it back to the base (Fig. 1). Clearly, although this is a very simple example, there could be quite a few solutions (from a very abstract to more detailed one). Likewise, Fig. 2 pictures alternative different XM models that can be considered as solutions to the foraging agent problem.

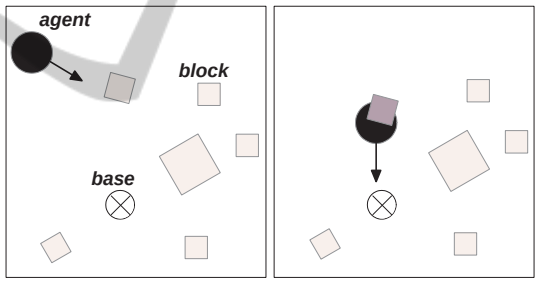

Figure 1: The foraging agent example.

Table 1 demonstrates three ways of modelling the foraging agent problem (the numbering a), b) and c) corresponds to the numbering in Fig. 2). The first solution a) is a very abstract representation that does not even takes into consideration the position (the coordinates) of the agent. The fact that XMs are generic and do not impose modelling of a position, in such an example might result into an incomplete model. A more detailed representation can be derived from the second solution b) from Table 1 (it can be noted that the representation is a design choice, for instance the memory variables that correspond to positions are integers). Yet again, this representation is way too complex and probably more difficult for understanding. Finally, the last representation c), is the best solution in terms of completeness and complexity. More comprehensive specification can be found as (Petreska, 2011).

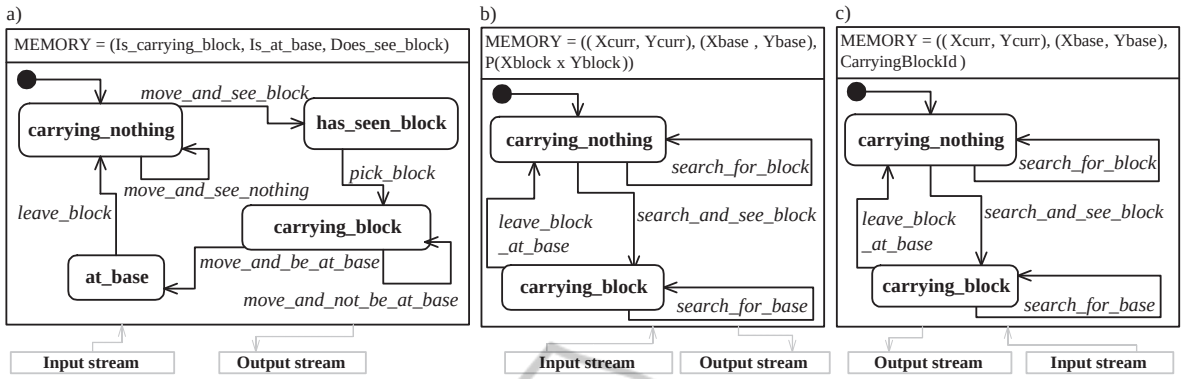

Figure 2: Examples of modelling the foraging agent example: a) very abstract representation b) more detailed, but complex representation c) the best represented solution.

Table 1: Different ways to modelling the foraging agent example.

a)	Q = {carrying_nothing, has_seen_block, carrying_block, at_base} M = {Is_carrying_block, Is_at_base, Does_see_block}, where Is_carrying_block, Is_at_base, Does_see_block ∈ {true, false} m_o = {false, false, false} q_o = {carrying_nothing} Σ = {"move to a place w/o block", "move to a place with block", "pick block", "search for base", "move to base", "leave block"} Γ = {"agent keeps moving empty", "agent detected block", "agent picked block", "agent searches for base", "agent found base", "agent left block"}
b)	Q = {carrying_nothing, carrying_block} M = ((X_{curr}, Y_{curr}), (X_{base}, Y_{base}), $\mathbb{P}(X_{block} \times Y_{block})$, Hand) where X_{curr}, Y_{curr}, X_{block}, Y_{block}, X_{base}, $Y_{base} \in \mathbb{Z}$, Hand ∈ {full, empty} m_o = ((2, 3) , (0, 0), {(2, -3), (4, -6), (2, 1), (3, 5), (-1, 5)}, empty) q_o = {carrying_nothing} Σ = (X_{new} , Y_{new}), where X_{new} , $Y_{new} \in \mathbb{Z}$ Γ = {"agent keeps moving empty", "agent detected and picked block", "agent searches for base", "agent found base and left block"}
c)	Q = {carrying_nothing, carrying_block} M = ((X_{curr}, Y_{curr}), (Xbase, Ybase), CarryingBlockId) where X_{curr}, Y_{curr}, Xbase, Ybase ∈ $\mathbb{Z}$, CarryingBlockId ∈ {$block_1$, $block_2$, ... $block_n$} ∪ nil, n ∈ $\mathbb{N}$ m_o = ((2, 3), (0, 0), nil) q_o = {carrying_nothing} Σ = ((X_{new}, Y_{new}), BlockId), where X_{new}, $Y_{new} \in \mathbb{Z}$, BlockId ∈ {$block_1$, $block_2$, ... $block_n$} ∪ {nil}, n ∈ $\mathbb{N}$ Γ = {"agent keeps moving empty", "agent detected and picked block", "agent searches for base", "agent found base and left block"}

2.2 Shortcomings of XM for Spatial Agents

The case study in the previous section demonstrated that there might be different ways to modelling spatial agents with the XM approach, even for the simplest scenario. This lead towards identification of the following shortcomings when modelling spatial agents with XM is concerned:

- There might be many different solutions (even for the simplest model) for representing the commonly found properties, such as the initial position or the direction of a spatial agent. This makes it more difficult to read a given model (we have to understand how the modeller decided to represent these properties) and even to create one (every time the modeller has to think how to represent them).

- There are difficulties in simulating a given model because there is not a standard way that deals with

manipulation and processing of the spatial properties like the initial position or the direction of a spatial agent.

- The memory holds all data structures required, including the position and the direction.

Initiated by these shortcomings, a question that can be imposed is: *How can we extend XMs to support spatial agent modelling natively and why?* The motivation behind this question can be further broadened into the following aspects:

- The subset of MAS that deal with movement in space is quite numerous, starting with biologically inspired MAS, up to MAS used in many industrial applications like robotics, etc.
- Different modellers might represent a spatial agent's basic characteristics, like position and direction, in different ways.
- The current XM representation for a spatial agent model does not directly map to an animation/simulation.
- The current XM representation for a spatial agent model is rather cumbersome/difficult to code, and in many situations it is also difficult to be understood.
- When it comes to verifying a spatial model with XM, this will result into space explosion due to the spatial information.

3 INTRODUCTION TO SPATIAL X-MACHINES

^{sp}XMs represent an extension of stream XMs by defining three new components and modifying some existing ones in order to facilitate unification with the newly defined components. The input and the output set, the memory, the set of states and the next state remain intact, because these structures do not deal with the spatial attributes that we intend to support. On the other hand, the following components have been introduced:

- A tuple containing the current position of the agent and an integer that represents its current direction. The current position determines the agent's location in its environment, and the direction represents its heading (such as south, north, etc.).
- A set containing elementary operations. These operations allow manipulation with the current position tuple and the current direction.

A ^{sp}XM is a 13-tuple ^{sp}XM = (Σ, Γ, Q, q_0, π, π_0, θ, θ_0, M, m_0, E, Φ, F) that can be formally defined as (See Fig 3):

- Σ is an input set of symbols;
- Γ is an output sets of symbols;
- Q is a finite set of states;
- q_0 is the initial state;
- M is an n-tuple called memory;
- m_0 is the initial memory;
- π is a tuple of the current position, i.e. (x, y) when a 2D representation is considered;
- π_0 is the initial position;
- θ is an integer in the range 0 to 360, that represents a direction (integer values are used as a design choice);
- θ_0 is the initial direction;
- E is a set which contains elementary positioning operations: e_i such as $e_i : \Pi \times \Theta \longrightarrow \Pi \times \Theta$, such as direction, moving forward and moving to a specific position;
- Φ is a finite set of partial functions ϕ that map a memory state, position, direction and set of inputs to a new memory state, position, direction and set of outputs:
  ϕ: $M \times \pi \times \theta \times \Sigma \longrightarrow M \times \pi \times \theta \times \Gamma$; and
- F is a function that determines the next state, given a state and a function from the type Φ, such as F: $Q \times \Phi \rightarrow Q$.

If we take a closer look at the memory M, it may be noted that M is composed of M' $\hat{}< \pi, \theta >$, where M' is a memory structure from the standard XM. Moreover, talking about the set which contains elementary operations, we have currently defined three operations:

- change_direction m - changes the spatial agent's direction to m, where m is of type θ and m $\in \Theta$ (ex. *change_direction 60*)
- move_n_forward n - moves forward for n units, where n is an integer (ex. *move_n_forward 3*)
- move_to_position x y - moves to specific position (x, y), where x is the x-coordinate, y is the y-coordinate of the agent and (x, y) $\in \Pi$ (ex. *move_to _position 126 43*)

As it can be determined from the definition, a ^{sp}XM in essence provides a separation of the behaviour within the system that deals with the movement (and the other spatial attributes) from the rest

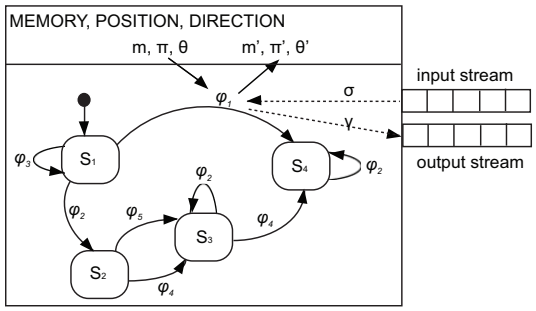

Figure 3: An abstract example of a ^{sp}XM.

of the behaviour. This way we establish a standardised way to modelling motion, which is easily understandable and provides a direct mapping to an animation/simulation. And finally, this work is achieved by maintaining an obvious equivalence with the standard XM.

A definition language for modelling the foraging agent problem (from Fig. 2) as a ^{sp}XM, is presented in (Petreska, 2011). Moreover, a discussion about the specification and the grammar for describing a ^{sp}XM model, can be found in (Petreska, 2011) as well.

4 VERIFICATION, VALIDATION AND ANIMATION OF ^{SP}XMS

Formal verification of spatial agents is an extremely complex task. On one hand stands the fact that the verification process leads to combinatorial explosion, because modelling these agents means modelling of their spatial properties (such as position or direction). Therefore, the verification would require exploration of a state space developed by the combination of all agent positions evolved through time (Petreska et al., 2011). On the other hand, there is the fact that the emergent properties of the system should be known in advance in order to be verified. The concept of emergence can be explained as a pattern appearing in the configuration of the agents, at some instance during the lifetime of the system. In biology or biology-inspired agents the emergence can be observed in-vivo (for example, line formation, flocks, schools, herds etc.). However, when it comes to artificial agents, it is not always straightforward. Driven from these two problems, it might be desirable to combine several formal with informal techniques that would be able to join forces towards the verification of spatial MAS (Petreska et al., 2011).

The models of an X-machine can be described with the X-machine Definition Language (XMDL (Kefalas and Kapeti, 2000)) and textual simulation. XMDL is a listing of definitions that match the tuples of X-machine's definition. Starting with the diagram in Fig. 4, XMDL is facilitated with a parser built using Definite Clause Grammars (DCG) notation (Kefalas et al., 2002), which apart from the syntax errors, output as warnings any kind of logical errors or omissions. The semantic analysis and the rules for transformation are being checked by the compiling component, with the aid of defined rules under which the specification is translated into the equivalent Prolog code. This Prolog code is after utilised by an animation tool, which simulates the computation of an X-machine. Furthermore, the model checking component defines a new logic, XmCTL (Eleftherakis et al., 2002), and with the implementation of a model checking algorithms can determine whether a property is true or false. And finally, XMs are supported with automatic generation of test cases, which is proved that finds all faults in the implementation (Ipate and Holcombe, 1998).

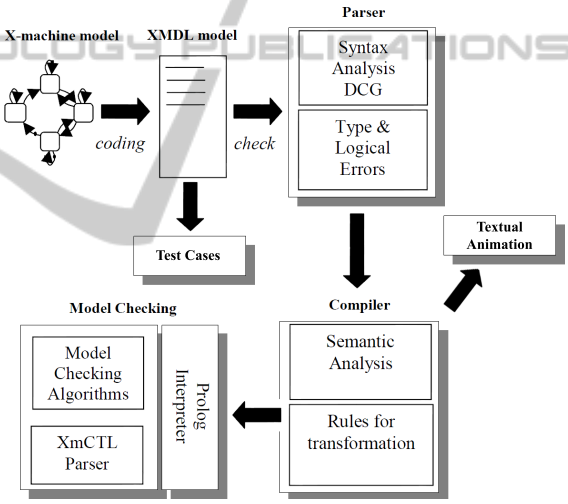

Figure 4: Verification and validation of an X-machine.

Taking ^{sp}XMs into consideration, the following discussion will concentrate on investigating whether they inherit the mentioned verification and validation techniques of XMs. An informal proof that a ^{sp}XM is equivalent to any XM could be derived by investigating:

- The memory M of a normal XM is equivalent to the structure of memory M, position Π and direction Θ within a ^{sp}XM. In other words, the position and direction can either become members of the memory tuple in a normal XM model, or they can be excluded from the model without loss of its integrity.
- Any function in a ^{sp}XM model can be translated into a function of the normal XM. More particu-

larly, the predefined operations in a *where* statement of a function in a ^{sp}XM model can be omitted or replaced with the standard XMDL syntax to preserve the logic flow.

Along these lines, by removing the newly defined components that in essence define a ^{sp}XM, what we get is still a completely valid skeleton of a normal XM model. Therefore, ^{sp}XMs tend to provide a standardised way to representing some properties of the system, which could also be represented with an XM model. Thus, the ^{sp}XM definition can lead in easy formalisation, verification (model checking), testing and implementation (Fig. 5). The only condition imposed would be not to test or model check the position (coordinates) and direction properties, which in turn will result into a state explosion.

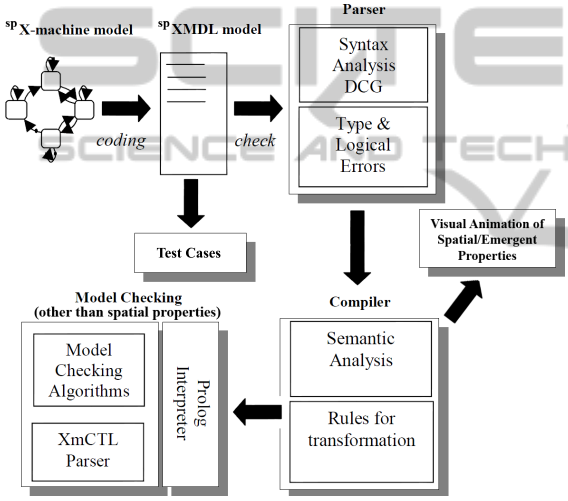

Figure 5: Verification and validation of a ^{sp}X-machine.

Referring to the initial discussion of this chapter for combining formal with informal techniques towards the verification of spatial MAS, we suggest that visual animation can be exploited for detecting the emergent properties of a system. In biological MAS, animation becomes even more interesting because of the spatial attributes of an agent, e.g. agents move in an n-dimensional space. This raises the question: *Having a model of a system, how can we visualise it?* An animator as a form of simulation, is any kind of program which given the code in the intermediate language, implements an algorithm to facilitate the computation of the model and its output though a textual description (Wilensky, 1999; Stamatopoulou et al., 2007). However, most of the animation techniques share one major drawback, i.e. the outputs they produce are in a textual form and thus not even close to the real visual perceptions on the system. Therefore, we focus on a visual simulation platform, namely NetLogo (Wilensky, 1999; Wilensky, 1997).

NetLogo is a simulation platform for visual animation of multi-agent systems regardless the number of agents, being supported by a functional language that can represent an agent's behaviour, as well as an environment for creation of a graphical user interface. NetLogo facilitates the verification of a biological model in a way that simulation scenarios may be executed, thus the expected behaviour of the system could be compared to the visual outcome. This platform was our initial choice due to its simplicity and the legacy of work we have done so far in experimenting with Netlogo features and emergent biological phenomena. Similar but more advanced development toolkits such as Repast (Collier and North, 2011) should not be excluded but could considered too, as alternatives to visualisation.

Given an XM model, it is not always easy nor uniform to map its representation into the equivalent NetLogo code. This is due to the already discussed disadvantages in Sec. 3 that deal with the behaviour of the system that represents the motion (and the other spatial attributes). However, ^{sp}XMs overcome this issue, and thus enhance visual animation (the agent's position and direction can be interpreted into motion within an animation platform). This feature opened the horizon towards ideas for automation of the simulation scenarios for a ^{sp}XM model, on which the authors are currently working.

5 CONCLUSIONS AND FUTURE WORK

Given X-machines, one might argue that the biological agent models might be very abstract, i.e. there is a freedom in the representation of a model. On the other hand, certain knowledge is required for simulating a biological agent, for instance the initial position or direction of the agent. This introduces difficulties in simulating a given model, because an X-machine does not specify how these knowledge will be modelled. Thus we presented an idea of extending X-machines into more specific formalism for modelling spatial agents that move in space, called ^{sp}XMs.

Further work could include extending ^{sp}XMs in a way to support other spatial agent properties and more functions which will bring in more realistic modelling of the spatial concept. Additionally, examples could be created towards simulation, testing and model checking of an XM and a ^{sp}XM model for a critical comparison, pointing out their differences and advantages.

Regarding simulation, a tool for automatic trans-

lation of a ^{sp}XM model into the NetLogo platform for a visual animation is currently being developed by the authors. Moreover, we currently work on a framework towards the verification of emergent behaviour of spatial MAS by utilising the ^{sp}XMs approach (Petreska et al., 2011). This framework basically tries to support identifying emergent behaviour by utilizing the tool for automatic translation (Petreska et al., 2011). Initially, we propose that the formal modelling should be accomplished with a formalism that is able to clearly distinguish modelling of various types of behaviours (spatial or other behaviours), such as ^{sp}XM. This would make it possible to apply different transformations facilitating further processing.

At this point, there are two paths. On one hand, the spatial behaviour can lead towards visual animation which will help detection of emergence (by utilizing NetLogo through the automatic translation tool). On the other hand, the spatial behaviour should be abstracted (together with the rest of the behaviours) in order to lead towards simulation and logging of time series data (Petreska et al., 2011). This might be accomplished with a tool such as FLAME (M. Pogson and Holcombe, 2006; R. Smallwood and Walker, 2004), used to animate XM models. The next step involves utilizing a tool for identifying patterns, such as DAIKON (Michael et al., 1999). Therefore, all of the patterns of behaviours together with the visual animation would produce a set of desired properties. Finally, they can be verified in the original spatial agent model by model checking.

Finally, the ^{sp}XM can be suitably transformed into an equivalent model in SPIN, PRISM or SMV (Holzmann, 1997; M.Kwiatkowska et al., 2001; McMillan, 1993). In this case, given a temporal formulae, all of the desired properties could be verified upon the original model.

REFERENCES

Cardelli, L. and Gardner, P. (2010). Processes in space. In *CiE'10*, pages 78–87, Heidelberg. Springer-Verlag Berlin.

Collier, N. T. and North, M. J. (2011). Repast SC++: A platform for large-scale agent-based modeling. Large-Scale Computing Techniques for Complex System Simulations, Wiley. (In Press).

Eleftherakis, G., Kefalas, P., and Sotiriadou, A. (2002). XmCTL: Extending temporal logic to facilitate formal verification of X-machines. pages 79–95, Analele Universitatii Bucharest. Matematica-Informatica.

Holzmann, G. J. (1997). The model checker spin. *IEEE IFans. on Software Engineering*, pages 279–295.

Ipate, F. and Holcombe, M. (1998). Specification and testing using generalised machines: a presentation and a case study. pages 61–81. Software Testing, Verification and Reliability.

Kefalas, P. (2002). Formal modelling of reactive agents as an aggregation of simple behaviours. In Vlahavas, I. P. and Spyropoulos, C. D., editors, *Proceedings of the 2nd Hellenic Conference on AI, SETN02, Lecture Notes in Artificial Intelligence 2308*, pages 461–472. Springer-Verlag.

Kefalas, P., Eleftherakis, G., and Kehris, E. (2003a). Communicating X-machines: A practical approach for formal and modular specification of large systems. *Information and Software Technology*, 45:269–280.

Kefalas, P., Eleftherakis, G., and Sotiriadou, A. (2002). Developing tools for formal methods. In *Proceedings of the 9th Panehellenic Conference in Informatics*.

Kefalas, P., Holcombe, M., Eleftherakis, G., and Gheorge, M. (2003b). A formal method for the development of agent based systems. In Plekhanova, V., editor, *Intelligent Agent Software Engineering*, pages 68–98. Idea Group Publishing Co.

Kefalas, P. and Kapeti, E. (2000). A design language and tool for X-machines specification. In Fotiadis, D. I. and Nikolopoulos, S. D., editors, *Advances in Informatics*, pages 134–145, Singapore. World Scientific Publishing Company.

M. Pogson, R. Smallwood, E. Q. and Holcombe, M. (2006). Formal agent-based modelling of intracellular chemical interactions. *Biosystems*, 85:37–45.

McMillan, K. L. (1993). *Symbolic Model Checking*. Kluwer Academic Publishers, Englewood Cliffs, NJ.

Michael, D. E., William, G. G., Yoshio, K., and Notkin, D. (1999). Dynamically discovering pointer-based program invariants. Technical Report UW-CSE-99-11-02, University of Washington Department of Computer Science and Engineering, Seattle, WA. Revised March 17, 2000.

M.Kwiatkowska, G.Norman, and D.Parker (2001). Prism: Probabilistic symbolic model checker. *In Proc. PAPM/PROBMIV'01 Tools Session*, pages 7–12.

Petreska, I. (2011). Further material. http://people.seerc.org/petreska/further_material.html.

Petreska, I. and Kefalas, P. (2011). Population p systems with moving active cells. In Gheorghe, M., Păun, G., and Verlan, S., editors, *Twelfth International Conference on Membrane Computing (CMC12)*, pages 421–432, Fontainebleau, France. Laboratoire d'Algorithmique Complexité et Logique of the University of Paris Est – Créteil Val de Marne.

Petreska, I., Kefalas, P., and Gheorghe, M. (2011). A framework towards the verification of emergent properties in spatial multi-agent systems. In Ivanovi, M., Ganzha, M., Paprzycki, M., and Badica, C., editors, *Proceedings of the Workshop on Applications of Software Agents*, pages 37–44. Department of Mathematics and Informatics Faculty of Sciences, University of Novi Sad, Serbia.

Pogson, M., Holcombe, M., Smallwood, R., and Qwarnstrom, E. (2008). Introducing spatial information into

predictive NF-kB modelling – An agent-based approach. *PLoS ONE*, 3(6):e2367.

R. Smallwood, M. H. and Walker, D. (2004). Development and validation of computational models of cellular interaction. *Journal of Molecular Histology*, 35:659–665.

Romero-Campero, F. J., Twycross, J., Camara, M., Bennett, M., Gheorghe, M., and Krasnogor, N. (2009). Modular assembly of cell systems biology models using P Systems. In *International Journal of Foundations of Computer Science*, pages 427–442.

Stamatopoulou, I., Kefalas, P., and Gheorghe, M. (2007). Operas: A framework for the formal modelling of multi-agent systems and its application to swarm-based systems. In *ESAW*, pages 158–174, Berlin, Heidelberg. Springer-Verlag.

Wilensky, U. (1997). *NetLogo Segregation model*. Center for Connected Learning and Computer-Based Modeling, Northwestern Univ., Evanston, IL. http://ccl.northwestern.edu/netlogo/models/Segregation.

Wilensky, U. (1999). *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern Univ., Evanston, IL. http://ccl.northwestern.edu/netlogo/.

Y. Bdard, J. P. and Caron, C. (1992). Spatial data modeling: The Modul-R formalism and CASE technology. *ISPRS Symposium*. Washington, United-States.