OPERAS for Space: Formal Modelling of Autonomous Spacecrafts

Ioanna Stamatopoulou[1], Petros Kefalas[2], Marian Gheorghe[3]

[1]SEERC, 17 Mitropoleos Str, Thessaloniki, Greece, istamatopoulou@seerc.org
[2]City College, Dept. of Computer Science, 13 Tsimiski Str, 54624 Thessaloniki, Greece, kefalas@city.academic.gr
[3]The University of Sheffield, Dept. of Computer Science, Regent Court, 211 Portobello Str, Sheffield S1 4DP, UK, marian@dcs.shef.ac.uk

Abstract

We present OPERAS, a formal method that facilitates the development of biologically inspired multi-agent systems exhibiting emergent behaviour through self-organisation. We describe how a particular version of this method, namely $OPERAS_{XC}$, could employ and integrate the most prominent characteristics of finite state machines and biological computation systems, such as X-machines and P Systems respectively. We apply this method to formally model a system reported by NASA, which consists of autonomous spacecrafts in a mission for exploring asteroids.

1 Introduction

Throughout the past years, there has been an increasing interest towards biological and biologically inspired systems, particularly with the intent to create software systems that model the behaviour of their biological counterparts (ants, termites, bees, flocks of birds, tumours etc). The motivation behind the development of such software systems varies. In our discipline, the understanding of how nature deals with various problematic situations has inspired problem solving techniques that are applicable to a wide range of situations. Swarm Intelligence and Ant Colony Optimisation techniques have been successfully applied to robotics [Dorigo et al., 2004] and DNA computing [Paun et al., 1998]. Other unconventional, biology inspired computational models [Gheorghe, 2005] can solve NP-complete problems.

These systems can be directly mapped to multi-agent systems (MAS) by considering each entity as an agent. The overall system's behaviour is merely the result of the agents' individual actions, their interactions among them and the environment. This also points to self-organisation and how collective behavioural patterns emerge as a consequence of individuals' local interactions in the lack of knowledge of the entire environment or global control.

The more complex a MAS is, the more difficult to ensure correctness at the modelling level. Correctness implies that all desired properties are verified at the end of the

modelling phase and that an appropriate testing technique is applied to prove that the implementation has been built in accordance to the verified model. Not all agent-engineering paradigms provide such means and it is accepted that the most reliable means for ensuring correctness lie within the field of formal methods that can, by nature, provide the necessary verification and testing techniques.

Another key aspect that has to be dealt with at the modelling level is the dynamic nature of MAS and how their structure is constantly reconfigured. By 'structure' we imply (i) the number of the agents, and (ii) either their physical placement in space or, more generally, the structure that is dictated by the communication channels among them. Most modelling methodologies assume a fixed, static structure that is not realistic since in a dynamic MAS, communication between two agents may need to be established or ceased at any point and also new agents may appear in the system while existing ones may be removed. One additional issue that the inherent dynamic nature of these systems raises has to do with distinguishing between the modelling of the individual agents (behaviour) and the rules that govern the communication and evolution of the collective MAS (control). By 'control' we do not imply central control, as this would cancel any notion of self-organisation. Rather, we refer to the part of the agent that takes care of non-behavioural issues. A modelling method that allows such a distinction would greatly assist the modeller by breaking down the work into two separate and independent activities.

In this paper we propose a new formal approach, called OPERAS, that facilitates the development of MAS of the nature of many biology and biology-inspired systems. The next section provides a brief description of a representative case study from the class of similar MAS modelling problems. Section 3 introduces OPERAS formal definition, followed by a formal model for the case problem in question, in Section 4. Finally, Section 5 discusses issues arising from our attempt and concludes the paper.

2 Autonomous Spacecrafts for Asteroid Exploration

A representative example of a system, which clearly possesses all the aforementioned characteristics of a dynamic MAS is the NASA Autonomous Nano-Technology Swarm (ANTS) system [Rouf et al., 2004]. The NASA ANTS project aims at the development of a mission for the exploration of space asteroids with the use of different kinds of unmanned spacecrafts. This case is interesting because there has been a thorough investigation on how formal methods can ensure the safety critical nature of the mission. Since correctness of the system has been identified as a primary requirement, work on the particular project included research on and comparison of a number of formal methods [Rouf et al., 2004].

The ANTS mission uses of three kinds of unmanned spacecrafts: leaders (or rulers or coordinators), workers and messengers (Fig. 1). The leaders are the spacecrafts that are aware of the goals of the mission and have a non-complete model of the

environment. Their role is to coordinate the actions of the spacecrafts that are under their command but by no means should they be considered to be a central controlling mechanism, as all spacecrafts' behaviour is autonomous. Depending on its goals, a leader creates a team consisting of a number of workers and at least one messenger. Workers and messengers are assigned to a leader upon request by (i) another leader, if they are not necessary for the fulfillment of its goals, or (ii) earth (if existing spacecrafts are not sufficient in number to cover current needs, new spacecrafts are allocated to the mission).

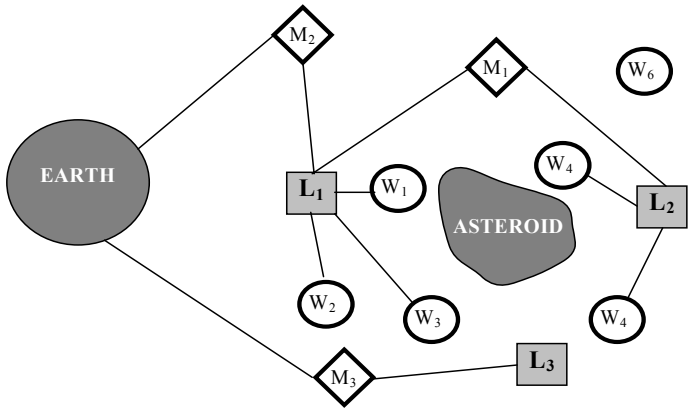

Figure 1. *An instance of the ANTS mission, L: Leader, W: Worker, M: Messenger.*

A worker is a spacecraft with a specialised instrument able, upon request from its leader, to take measurements from an asteroid while flying by it. It also possesses a mechanism for analysing the gathered data and sending the analysis results back to its leader in order for them to be evaluated. This in turn might update the view of the leader, i.e. its model of the environment, as well as its future goals.

The messengers, finally, are the spacecrafts that coordinate communication among workers, leaders and the control centre on earth. While each messenger is under the command of one leader, it may also assist in the communication of other leaders if its positioning allows it or conditions demand it.

What applies to all types of spacecrafts is that in the case that there is a malfunctioning problem, their superiors are being notified. If the damage is irreparable they need to abort the mission while on the opposite case they may "heal" and return back to normal operation.

3 OPERAS: Formal Modelling of Multi-Agent Systems

In an attempt to formally model each individual agent as well as the dynamic behaviour of the overall system, we need a formal method that is capable of rigorously describing all the essential aspects, i.e. knowledge, behaviour, communication and dynamics. Some formal methods have the means to efficiently define the data structures of a system and the operations employed to modify the values in these structures (Z, VDM). Others are better in describing the control over a system's states (FSM, Petri Nets) and yet others put emphasis on the concurrency and communication of processes (CCS, CSP). Finally, new computation approaches inspired by biological processes in living cells, introduce concurrency as well as neatly tackle the dynamic structure of multi-component systems (P Systems, Brane Calculus, Gamma, Cham, MGS) [Banatre et al, 1990; Berry et al, 1992; Paun, 2000]. An interesting comparison of various formal methods for the verification of emergent behaviours in swarm-based systems is reported [Rouf et al., 2004].

3.1 OPERAS

Definition 1. A *Dynamic Multi-Agent System* is defined by the tuple *(O, P, E, R, A, S)* where:

- the set of rules, O, that define how the system structure evolves by applying appropriate reconfiguration operators;
- the set of percepts, P, for the agents;
- the environment's model / initial configuration, E;
- the relation, R, that defines the existing communication channels;
- the set of participating agents, A, and
- the set of definitions of agent types, S, that may be present in the system.

More particularly:

- the rules in O are of the form $condition \Rightarrow action$ where *action* involves the application of one or more of the operators that create/remove a communication channel between agents or introduce/remove an agent into/from the system;
- P is the union of the set of percepts of all participating agents;
- R: $A \times A$ with $(A_i, A_j) \in R$, $A_i, A_j \in A$, i.e. agent A_i may send messages to agent A_j;
- $A = \{A_1, \ldots, A_n\}$ where A_i is a particular agent defined in terms of its individual behaviour and its local mechanism for controlling reconfiguration;
- $S_k = (Behaviour_k, Control_k) \in S$, $k \in Types$ where *Types* is the set of identifiers of the types of agents, $Behaviour_k$ is the part of the agent that deals with its individual behaviour and $Control_k$ is the local mechanism for controlling reconfiguration; each participating agent A_i of type k in A is a particular instance of a type of agent: $A_i = (Beh_k, Ctrl_k)_i$ (Fig. 2).

Regarding the modelling of each type of agent S_k, there are more than one options to choose from in order to specify its behavioural part and the same applies for its control mechanism. However, we have long experimented with two formal methods, each one appropriate for the behaviour and control respectively, namely Communicating X-machines and Population P Systems. A detailed description of how the two approaches are used in MAS modelling can be found in [Kefalas et al., 2005; Stamatopoulou et al., 2005].

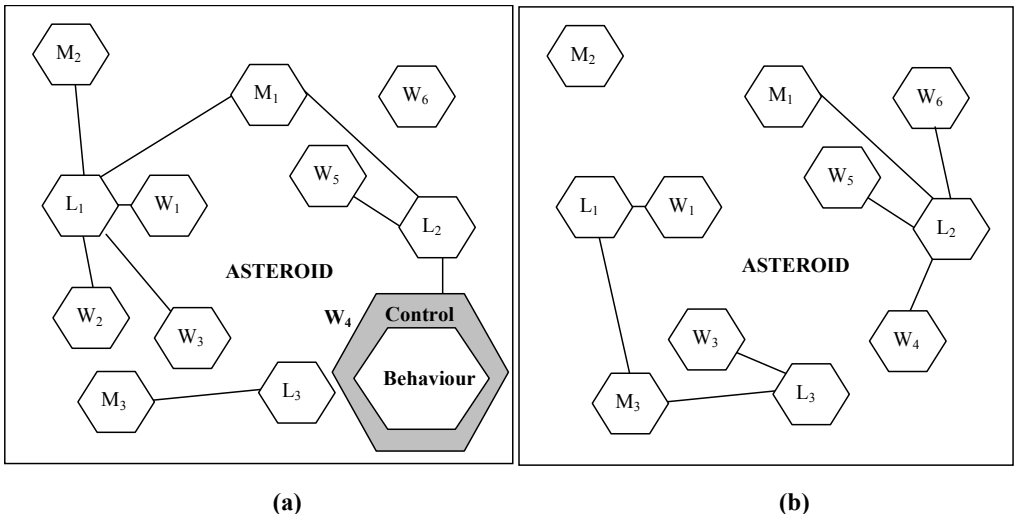

Figure 2. *(a) An instance of MAS structure corresponding to ANTS in Fig.1 with an OPERA agent (W_4) consisting of separate Behaviour and Control components. (b) A change in the structure of MAS after possible events (e.g. destruction of worker W_2, leader L_1 employs worker W_6 etc.).*

X-machines (XM), a state-based formal method [Eilenberg, 1974], are considered suitable for the formal specification of a system's components. Stream X-machines, in particular, were found to be well suited for the modelling of reactive systems. Since then valuable findings using the X-machines as a formal notation for specification, communication, verification and testing purposes have been reported [Eleftherakis, 2003; Holcombe and Ipate, 1998; Kefalas et al., 2003]. An X-machine model consists of a number of states and also has a memory, which accommodates mathematically defined data structures. The transitions between states are labeled by functions. In addition to having stand-alone X-Machine models, communication is feasible by redirecting the output of one machine's function to become input to a function of another machine. The system structure of *Communicating X-machines* is defined as the graph whose nodes are the components (CXM) and edges are the communication channels among them.

One the other hand, *Population P Systems* (PPS) [Bernandini et al., 2004] is a collection of different types of cells evolving according to specific rules and capable of exchanging biological/chemical substances with their neighbouring cells. PPS provide a straightforward way for dealing with the change of a system's structure, however, the rules specifying the behaviour of the individual cells in a PPS are more commonly of the simple form of rewrite rules which are not sufficient for describing the behaviour of the respective agent a cell may represent.

We may now move on to a more formal OPERAS definition, namely $OPERAS_{XC}$ that uses both a CXM and PPS-cell-inspired construct for specifying each of the agents. An abstract example of an $OPERAS_{XC}$ model with two agents is depicted in Fig. 3.

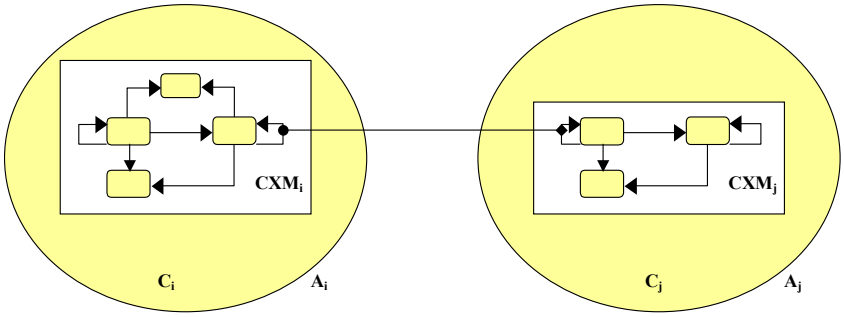

Figure 3. *An abstract example of an $OPERAS_{XC}$ consisting of two agents.*

For the following, we consider that the computation state of a CXM describing the behaviour of an agent is a 3-tuple $Q \times M \times \Phi$ that represents the state the XM is in (q_i), its current memory (m_i) and the last function that has been applied (φ_i).

Definition 2. A Dynamic Multi-Agent System $OPERAS_{XC}$ is defined as the tuple (O, P, E, R, A, S) where:

- the rules in O are of the form *condition*$\Rightarrow$*action* where *condition* is a conjunction of (q, m, φ) and *action* involves the application of one or more of the operators attachment **ATT**, detachment **DET**, reconfiguring the communication channels among existing CXMs and generation **GEN**, destruction **DES**, generating or destroying an agent in the system. Additional communication rules also exist, as in PPS, so that there is indirect communication (through the environment) between the control cells;
- $P = P_A \dot{\cup} P_C$ is the set of percepts of all participating agents, where $P_A = \Sigma_1 \cup ... \dot{\cup} \Sigma_t$ is the set of inputs perceived by the XMs and $P_C = (Q_1 \dot{\times} M_1 \times \Sigma_1) \cup ... \dot{\cup} (Q_t \dot{\times} M_t \times \Sigma_t)$ is the set of objects (alphabet) of the PPS (t being the number of types of agents);
- $E = \{(q, m, \varphi)_i \mid 1 \le i \le n, q \in Q_i, m \in M_i, \varphi \in \Phi_i\}$ holding information about the initial computation states of all the participating agents;
- R: $CXM \times CXM$ (*CXM*: the set of CXMs that model agent behaviour);

- $A = \{A_1, \ldots, A_n\}$ where $A_i = (CXM_k, C_k)_i$ is a particular agent of type k defined in terms of its individual behaviour (CXM_k) and its local mechanism for controlling reconfiguration (C_k). The control cell is of the form $C_k = (w_i, o_i)$ where w_i is the multi-set of objects it contains and $o_i \in O$ is the set of rules that correspond to the particular type of agent, k;
- $S = \{(XT_k, C_k) \mid k \in Type\}$, where XT_k is an XM *type* (no initial state and memory).

3.2 Computation in OPERAS$_{XC}$

In this model, each control cell implicitly knows the computation state (q, m, φ) of the underlying XM that models behaviour. Additionally, environmental input is directed straight to the agent's behavioural part. More particularly in each computation cycle:

- An input σ triggers a function of the behaviour CXM and the updated information about the agent's computation state (q', m', φ') is updated in the control cell;
- A copy of the object (q', m', φ') is placed in the environment for other agents in the local environment to have access to it;
- Objects from the environment representing the computation states of neighbouring agents are imported;
- Finally, all the reconfiguration rules in O of the type of the particular cell are being checked and if necessary applied, in any defined order.

Since the model follows the computation rules of a CXM system (triggered by the behaviour component's input, asynchronously for the different participating agents), computation of the behaviour-driven version of *OPERAS*$_{XC}$ is *asynchronous*. In another version of *OPERAS*$_{XC}$, the computation is control-driven (*synchronous*).

4 OPERAS$_{XC}$ Model for Autonomous Spacecrafts in ANTS

The leader agent L can be modelled as an X-machine, whose state transition diagram F_L is depicted in Fig. 4. $Q_A = \{Processing, Malfunctioning, Aborting\}$ is the set of states a leader may be in. Its memory contains information about its current status (i.e. position and operational status), the IDs and statuses of the messengers and workers under its command, the analysis results up to this point, its current model of the surroundings as well as its goals:

M_L: $Status \dot{\times} \mathbf{P}(M \times status) \dot{\times} \mathbf{P}(W \times status) \dot{\times} AnalysisResults \dot{\times} Model \dot{\times} Goals$

where *Status*: $(Z \times Z \times Z) \times \{Q_L\}$ (Z being the set of positive integers), $\mathbf{P}$ stands for powerset, M is the set of messengers, W is the set of workers and so forth.

The input set for the leader X-machine is: $\Sigma_L = \{abrt, problem, remedy\} \cup (W \times Status) \cup (W \times Measurements) \cup (\{request, requestFromEarth, requestedFor\} \times Instrument)$, where *abrt*, *problem*, *remedy*, *request*, *requestedFor* are constants and *Instrument*) is the set of containing the different types of installed instruments of the workers. The output set Γ_L is a set of informative messages.

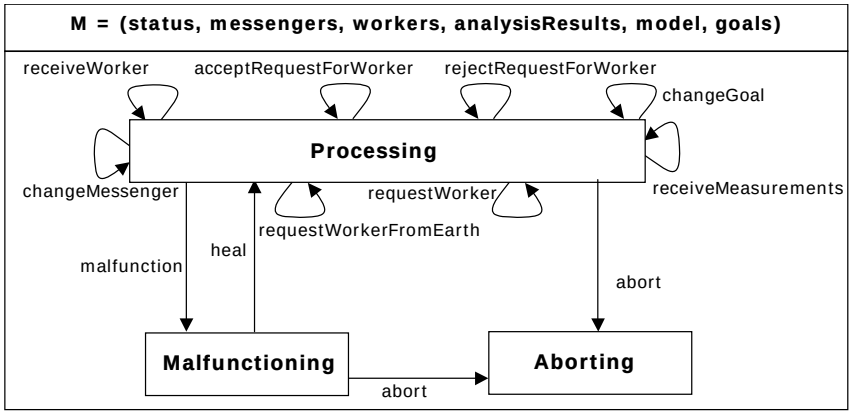

Figure 4. *State transition diagram of the Leader X-machine.*

Indicatively, some of the functions in the Φ_L set are:
acceptRequestForWorker((*requestedFor*,*instr*),(_,_,*workers*,_,_,_)) =
('*reassigned worker*', (_,_,*workers*',_,_,_))
if $(w_i, (\_,\_,instr)) \in workers$ and $isWorkerNeeded(w_i) == false$
where $workers' = workers \setminus (w_i, (\_,\_,instr))$

$receiveWorker(w_i,(\_,\_,workers,\_,\_,\_)) = ('received\ worker', (\_,\_,workers \cup (w_i),$

The models of the worker and messenger agent are similarly created but are not included in this paper due to space restrictions. According to $OPERAS_{XC}$, for the definition of the given system as a dynamic MAS, we need to assume an initial configuration. To keep the size restricted for demonstrative purposes, let us consider an initial configuration that includes one leader L_1, one messenger M_1 and two workers W_1, W_2.
The set O contains the five reconfiguration rules regarding (i) the generation of a new worker when the control centre on earth decides it should join the mission, (ii) the destruction (i.e. removal from the system) of any kind of agent in the case it must abort the mission, (iii) the establishment of a communication channel between a leader and a newly assigned to it worker and vice-versa, and (iv) the removal of a communication channel between a leader and a worker when the latter is being reassigned to a new leader. More particularly O contains the following rules:

$(\_,\_,requestWorkerFromEarth)_{A_i} \wedge earthHasAvailableWorkers() == true$
$\Rightarrow \mathbf{GEN}(W_i, q_{0_i}, m_{0_i}, ANTS)_A$

$(aborting,\_,\_)_{*this} \Rightarrow \mathbf{DES}(_{*this}, ANTS)_*$ where * stands for any type of agent.

$(\_,(\_,\_,L_i,\_,\_,\_,\_),leaveCurrentLeader)_{W_i} \Rightarrow \mathbf{DET}(W_i, L_i, \mathbf{DET}(L_i, W_i, ANTS))_W$

$(\_,(\_,\_,newLeader,\_,\_,\_,\_),joinNewLeader)_{W_i}$
$\Rightarrow \mathbf{ATT}(W_i, newLeader, ANTS)_W$

$(\_,(\_,\_,newWorker::workers,\_,\_,\_),receiveWorker)_{L_i}$
$\Rightarrow \mathbf{ATT}(L_i,newWorker,ANTS)_L$
$(\_,(\_,\_,newWorker::workers,\_,\_,\_),receiveWorkerFromEarth)_{L_i}$
$\Rightarrow \mathbf{ATT}(L_i,newWorker,ANTS)_L$

The set of percepts of all agents is: $P = \Sigma_L \times \Sigma_W \times \Sigma_M \times (Q_L \times M_L \times \Phi_L) \times (Q_W \times M_W \times \Phi_W) \times (Q_M \times M_M \times \Phi_M)$. Because all reconfiguration rules per type of agent rely only on conditions dependent on the computation state of the agent itself, the model needs not to communicate computation states among the different agents and there are, therefore, no additional communication rules.
Since in the assumed initial configuration we consider to have one group of spacecrafts under the command of one leader, all agents should be in communication with all others and so: $R=\{(L_1,W_1), (L_1,W_2), (L_1,M_1), (M_1,L_1), (M_1,W_1), (M_1,W_2), (W_1,L_1), (W_1,M_1), (W_2,L_1), (W_2,M_1)\}$. The set that contains all the agent instances becomes: $A = \{L_1, W_1, W_2, M_1)\}$ where $L_1 = (XM_{L1}, C_{L1})$, $W_i = (XM_{Wi}, C_{Wi})$, $1 \leq i \leq 2$, and $M_1 = (XM_{M1}, C_{M1})$. Finally, the set S that contains the "genetic codes" for all agent types is: $S = \{(XM_L, C_L), (XM_W, C_W), (XM_M, C_M)\}$ where XM_L, XM_W, XM_M are the X-machines defined previously.

5 Conclusions and Further Work

We presented OPERAS, a formal method framework, with which one can model the behaviour and the control of an agent as separate components as well as formally describe the changes that occur in the structure of a dynamic multi-agent system. We employed Communicating X-machines and ideas from Population P Systems to define $OPERAS_{XC}$, a particular instance of the framework. These gave us the opportunity to combine the advantages that X-machines have in terms of modelling the behaviour of an agent with the advantages that Population P Systems have in terms of defining the control over the structure of the system. We have experimented with modelling of various biological and biologically inspired systems, and in this paper, we presented the OPERAS model of autonomous spacecrafts.

We continue the investigation of how OPERAS could employ other formal methods that might be suitable for this purpose. In the near future, we will focus on theoretical aspects of the framework, in order to demonstrate its usefulness towards developing correct systems (i.e. complete testing and verification) as well as to consider various types of transformations that could prove its power for formal modelling.

References

Banatre, J. and Le Metayer, D. (1990). The gamma model and its discipline of programming. *Science of Computer Programming*, 15:55–77.

Bernandini, F. and Gheorghe, M. (2004). Population P Systems. *Journal of Universal Computer Science*, 10(5):509–539.

Berry, G. and Boudol, G. (1992). The chemical abstract machine. *Journal of Theoretical Computer Science*, 96(1):217–248.

Dorigo, M., Trianni, V., Sahin, E., Gross, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J. L., and Mondada, F. (2004). Evolving self-organizing behavior. *Autonomous Robots*, 17(2-3):223–245.

Eilenberg, S. (1974). *Automata, Languages and Machines*. Academic Press.

Eleftherakis, G. (2003). *Formal Verification of X-machine Models: Towards Formal Development of Computer-based Systems*. PhD thesis, Department of Computer Science, University of Sheffield.

Gheorghe, M., editor (2005). *Molecular Computational Models: Unconventional Approaches*. Idea Publishing Inc.

Holcombe, M. and Ipate, F. (1998). *Correct Systems: Building a Business Process Solution*. Springer-Verlag, London.

Kefalas, P., Eleftherakis, G., and Kehris, E. (2003). Communicating X-machines: A practical approach for formal and modular specification of large systems. *Journal of Information and Software Technology*, 45(5):269–280.

Kefalas, P., Stamatopoulou, I., and Gheorghe, M. (2005). A formal modelling framework for developing multi-agent systems with dynamic structure and behaviour. In Pechoucek, M., Petta, P., and Varga, L. Z., editors, Lecture Notes in Artificial Intelligence, No. 3690, pp 122–131. Springer Verlag.

Paun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143. Also circulated as a TUCS report since 1998.

Paun, G., Rozenberg, G., and Salomaa, A. (1998). *DNA Computing: New Computing Paradigms*. Springer-Verlag.

Rouff, C., Vanderbilt, A., Hinchey, M., Truszkowski, W., and Rash, J. (2004). Properties of a formal method for prediction of emergent behaviors in swarm-based systems. In *Proceedings of the Second International Conference on Software Engineering and Formal Methods (SEFM'04)*, pages 24–33.

Stamatopoulou, I., Kefalas, P., and Gheorghe, M. (2005). Modelling the dynamic structure of biological state-based systems. BioSystems, No 87, pp.142-149