$OPERAS_{CC}$: An Instance of a Formal Framework for MAS Modelling Based on Population P Systems

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

[1] South-East European Research Centre, Thessaloniki, Greece
istamatopoulou@seerc.org
[2] Dept. of Computer Science, CITY College, Thessaloniki, Greece
kefalas@city.academic.gr
[3] Dept. of Computer Science, University of Sheffield, UK
m.gheorghe@dcs.shef.ac.uk

Summary. Swarm-based systems are biology-inspired systems which can be directly mapped to multi-agent systems (MAS), possessing characteristics such as local control over the decisions taken by the agents and a highly dynamic structure which continuously changes. This class of MAS is of a particular interest because it exhibits emergent behaviour through self-organisation and finds itself applicable to a wide range of domains. In this paper, we present *OPERAS*, an open formal framework that facilitates modelling of MAS, we describe how a particular instance of this framework, namely $OPERAS_{CC}$, could employ existing biological computation systems, such as Population P Systems, and demonstrate how the resulting method can be used to formally model a swarm-based system of autonomous spacecrafts.

1 Introduction

Lately, there has been an increasing interest toward biological and biology-inspired systems. From the smallest living elements, the cells, and how they form tissues in organisms to entire ecosystems and how they evolve, there is growing investigation on ways of specifying such systems. The intention is to create software that mimics the behaviour of their biological counterparts. Examples of biological systems of interest also include insect colonies (of ants, termites, bees etc.), flocks of birds, tumours growth—the list is endless. The understanding of how nature deals with various situations has inspired a number of problem solving techniques [13] that are applicable to a wide range of situations that had been puzzling computer scientists for decades. Swarm Intelligence [15, 16], Ant Colony Optimisation techniques [10] for example, has been successfully applied to robotics [11], network routing [8, 28] and data mining [1] and has inspired agent-based modelling platforms [19].

The promising feature is that these systems can be directly mapped to multi-agent systems (MAS) by considering each entity as an agent, with its own behavioural rules, knowledge, decision making mechanisms and means of communication with the other entities and with the environment. The overall system's behaviour is merely the result of the agents' individual actions, the interactions among them and between them and the environment. This also points to the issue of 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.

An additional modelling key aspect of MAS has not received much attention so far; it is the dynamic nature of MAS and how their structure is constantly reconfigured. By *structure* we imply (i) the changing number of agents in a MAS, and (ii) either their physical placement in the environment 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, modelling the behaviour and modelling the control.

Population P Systems with active membranes [4], a class of variants of P Systems [20] are membrane structures composed of membranes configured in an arbitrary graph and naturally possess the trait of reconfiguring their own structure through rules that restructure the graph and allow membranes to divide and die. Inspired by this appealing characteristic, in this paper we present a formal framework, called $OPERAS$, that facilitates the development of dynamic MAS of the nature of many biology and biology-inspired systems. The next section introduces $OPERAS$ formal definition, while section 3 presents an instance of this framework, namely $OPERAS_{CC}$ which utilises Population P Systems in order to model MAS. A brief description of a representative case study dealing with a swarm-based system follows in Section 4 which also deals with the formal model for the case problem in question. Finally, Section 5 discusses issues arising from our attempt and concludes the paper.

2 OPERAS: Formal Modelling of MAS

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. It is also important that the level of abstraction imposed by a formal method is appropriate enough to lead toward the implementation of a system. New computation approaches as well as programming paradigms 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) [2, 5, 20]. In agent-oriented software engineering, there have been several attempts to use formal methods, each one focusing on different aspects of agent systems development [3, 6, 9, 12, 21]. Other formal methods, such as π-calculus, mobile ambients and P Systems with mobile membranes [7, 17, 18], successfully deal with the dynamic nature of systems and concurrency of processes but lack intuitiveness when it comes to the modelling of an individual agent (lack of primitives and more complex data structures). An interesting comparison of various formal methods for the verification of emergent behaviours in swarm-based systems is reported in [22].

2.1 OPERAS Definition

We start this section by providing the definition of a model for a dynamic MAS in its general form.

A *Multi-Agent System* can be defined by the tuple (O, P, E, R, A, S) containing:

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

More particularly:

- the rules in O are of the form *condition* $\Rightarrow$ *action* where *condition* refers to the computational state of agents and *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 distributed union of the sets of percepts of all participating agents;
- $R : A \times A$ with $(A_i, A_j) \in R$, $A_i, A_j \in A$ meaning that 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: $A_i = (Beh_k, Ctrl_k)_i$.

2.2 OPERAS as an open framework

The general underlying idea is that an agent model consists of two parts, its behaviour and its control. The behaviour of an agent can be modelled by a formal method with its computation being driven by percepts from the environment. The control can be modelled by a set of reconfiguration rules which given the computation states of agents can change the structure of the system. The MAS structure is determined through the relation that defines the communication between the agents. The set of participating agents are instances of agent types that may participate in the system. This deals with the fact that an agent may be present at one instance of the system but disappear at another or that a new agent comes into play during the evolution of the MAS. This assumes that all agent types that may participate in the system should be known in advance.

There are still some open issues which, however, make the *OPERAS* approach a framework rather than a formal method. These are: (i) Which are the formal methods that can be used in order to model the behaviour? (ii) Which are the formal methods that can to use in order to model the control? (iii) Could the methods in (i) and (ii) be different? (iv) Should the agents' behaviour models communicate directly with other agents' behaviour models? (v) Should the agents' control models communicate with other agents' control models? (vi) Could communication be established implicitly through percepts of the environment? (vii) Which method chosen from (i) or from (ii) drives the computation of the resulting system? There is no unique answer to these questions but the modelling solution will depend on the choice of formal methods which are considered suitable to model either behaviour or control.

It is therefore implied that there are several options which could instantiate *OPERAS* into concrete modelling methods. 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. We have long experimented with various formal methods, such as X-machines with its communicating counterpart and Population P Systems with active cells. In this paper we present an instance of the framework that employs ideas from the latter, using a PPS to model both the behaviour as well as the control part of the agent.

3 $OPERAS_{CC}$

3.1 Population P Systems with active cells

A *Population P System* (PPS) [4] is a collection of different types of cells evolving according to specific rules and capable of exchanging biological / chemical substances with their neighbouring cells (Fig. 1). More formally, a PPS with active cells [4] is defined as a construct $\mathcal{P} = (V, K, \gamma, \alpha, w_E, C_1, C_2, \ldots, C_n, R)$ where:

- V is a finite alphabet of symbols called objects;

- K is a finite alphabet of symbols, which define different types of cells;
- $\gamma = (\{1, 2, \ldots n\}, A)$, with $A \subseteq \{\{i, j\} \mid 1 \leq i \neq j \leq n\}$, is a finite undirected graph;
- α is a finite set of bond-making rules of the form $(t, x_1; x_2, p)$, with $x_1, x_2 \in V^*$, and $t, p \in K$ meaning that in the presence of objects x_1 and x_2 inside two cells of type t and p respectively, a bond is created between the two cells;
- $w_E \in V^*$ is a finite multi-set of objects initially assigned to the environment;
- $C_i = (w_i, t_i)$, for each $1 \leq i \leq n$, with $w_i \in V^*$ a finite multi-set of objects, and $t_i \in K$ the type of cell i;
- R is a finite set of rules dealing with communication, object transformation, cell differentiation, cell division and cell death.

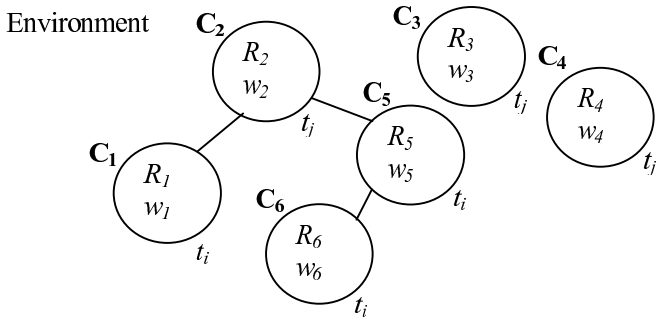

Fig. 1. An abstract example of a Population P System; C_i: cells, R_i: sets of rules related to cells; w_i: multi-sets of objects associated to the cells.

All rules present in the PPS are identified by a unique identifier, r. More particularly:

Communication rules are of the form $r : (a; b, in)_t$, $r : (a; b, enter)_t$, $r : (b, exit)_t$, for $a \in V \cup \{\lambda\}$, $b \in V$, $t \in K$, where λ is the empty string, and allow the moving of objects between neighbouring cells or a cell and the environment according to the cell type and the existing bonds among the cells. The first rule means that in the presence of an object a inside a cell of type t an object b can be obtained by a neighbouring cell non-deterministically chosen. The second rule is similar to the first with the exception that object b is not obtained by a neighbouring cell but by the environment. Lastly, the third rule denotes that if object b is present it can be expelled out to the environment.

Transformation rules are of the form $r : (a \rightarrow b)_t$, for $a \in V$, $b \in V^+$, $t \in K$, where V^+ is the set of non-empty strings over V, meaning that an object a is replaced by an object b within a cell of type t.

Cell differentiation rules are of the form $r : (a)_t \rightarrow (b)_p$, with $a, b \in V$, $t, p \in K$ meaning that consumption of an object a inside a cell of type t changes the cell, making it become of type p. All existing objects remain the same besides a which is replaced by b.

Cell division rules are of the form $r : (\, a\,)_t \rightarrow (\, b\,)_t\,(\, c\,)_t$, with $a, b, c \in V$, $t \in K$. A cell of type t containing an object a is divided into two cells of the same type. One of the new cell has a replaced by b while the other by c. All other objects of the originating cell appear in both new cells.

Cell death rules are of the form $r : (\, a\,)_t \rightarrow \dagger$, with $a \in V$, $t \in K$ meaning that an object a inside a cell of type t causes the removal of the cell from the system.

PPS provide a straightforward way for dealing with the change of a system's structure and this is the reason why we have chosen them to define an instance of the $OPERAS$ framework, namely $OPERAS_{CC}$.

3.2 Definition of $OPERAS_{CC}$

In $OPERAS_{CC}$, each agent (individual behaviour) is modelled as a PPS cell, and has a membrane wrapped around it, that is responsible for taking care of structure reconfiguration issues (control). In essence, this may be considered as a usual Population P System in which each cell is virtually divided in two regions, inner (for behaviour) and outer (for control), that deal with different sets of objects and have different kinds of rules that may be applied to them. An abstract example of an $OPERAS_{CC}$ model consisting of two agents is depicted in Fig. 2.

Additionally, when using a PPS for modelling purposes, we consider all objects to be attribute-value pairs of the form $att : v$ so that it is clear to which characteristic of the agent an object corresponds to.

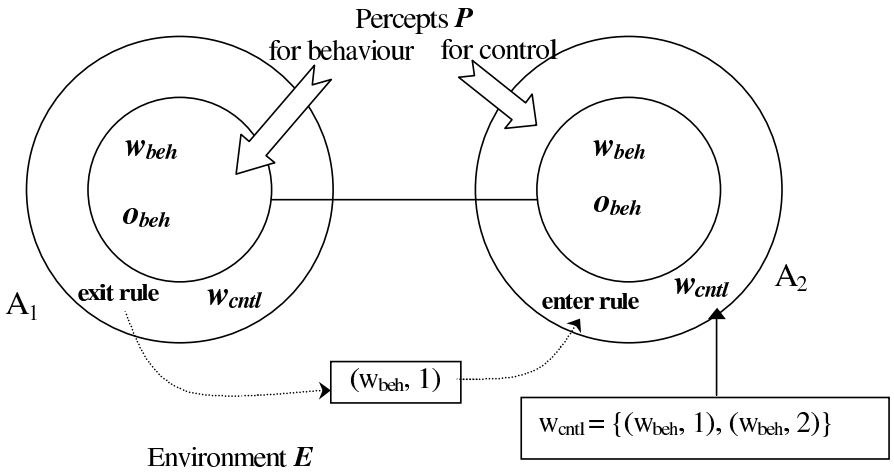

Fig. 2. An abstract example of a $OPERAS_{CC}$ consisting of two agents.

A MAS in $OPERAS_{CC}$ is defined as the tuple $(O, P, E, R, (A_1, \ldots A_n), S)$ (in correspondence to $P = (R, V, w_E, \gamma, (C_1, \ldots C_n), k)$ of a PPS) where:

- $A_i = (w_{beh}, w_{ctrl}, t)$, w_{beh} being the objects of the agent behaviour cell, w_{ant} the objects of the control cell (these objects possibly hold information about the w_{beh} objects (computation states) of neighbouring agent cells) and $t \in k$ the type of the cell;
- $O = O_A \cup O_C$.
  - The rules in O_A (to be applied only by the behaviour cells on the w_{beh} objects) are the transformation rules of a PPS that rewrite the objects, as well as the communications rules that move objects between cells that are linked with a bond (both kinds of rules do not affect the structure of the system).
  - The rules in O_C (to be applied only by the control cells on the w_{ctrl} objects) are the birth, death, differentiation and bond-making rules of a PPS (the kinds of rules that affect the structure of the system) as well as environment communication rules (receiving/sending objects from/to the environment) so that there is indirect communication between the control cells.
- $P = P_A \cup P_C$, the set of percepts of all participating agents where P_A is the set of inputs perceived by the behaviour cells and P_C is the set of inputs perceived by the control cells.
- E is the set of objects assigned to the environment holding information about the computation states of all the participating agents;
- R is the finite undirected graph that defines the communication links between the behaviour cells;
- S is the set of possible types of cells.

It should be noted that although the agent descriptions' set A appears fifth in $OPERAS$ definition tuple, from a practical perspective it is the first element being defined; the other tuple elements and their form are naturally dependent on the particular method(s) chosen to define the behavioural and control part of the agents.

Computation
In every computation cycle:

- In all the cells modelling the behaviour of the agent, all applicable object rules in O_A (transformation and communication) are applied;
- All control cells expel in the environment the w_{beh} objects (computation states of behaviour cells) along with the cell identity;
- All control cells import the computation states, w_{beh}, of neighbouring agents;
- All rules in O_C (bond-making, birth, death, differentiation) are triggered in the control cells, (if applicable) reconfiguring the structure of the system.

Since the model follows the computation rules of a PPS system, the overall system's computation is synchronous. Asynchronous computation may be achieved with the use of other methods for modelling the agents' behaviour and/or control. In [27] we present another instance of the framework, namely $OPERAS_{XC}$, which uses X-machines for the behavioural part of the agent and membranes wrapped around the machines for the control part, and apply it on the same swarm-based

system that we present hereafter. Because in that version of the framework computation is driven by the computation of the participating X-machines, overall computation is asynchronous.

4 $OPERAS_{CC}$ for a Swarm-based system

4.1 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 [22]. 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. Though each spacecraft can be considered as an autonomous agent, the successful exploration of an asteroid depends on the overall behaviour of the entire mission, as the latter emerges as a result of self-organisation. We chose this case study because relevant work on the particular project included research on and comparison of a number of formal methods [22, 23].

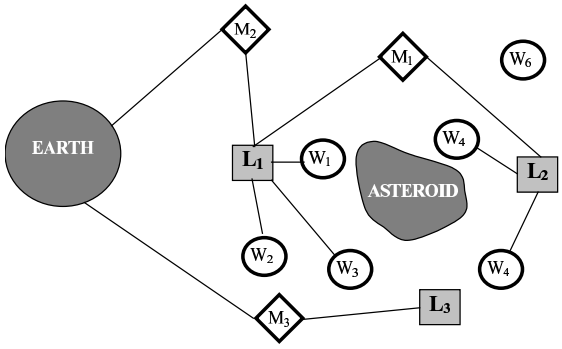

Fig. 3. An instance of the ANTS mission, L: Leader, W: Worker, M: Messenger.

The ANTS mission uses of three kinds of unmanned spacecrafts: L_i, leaders (or rulers or coordinators), W_i, workers and M_i, messengers (Fig. 3). 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 messengers. 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).

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 and 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.

4.2 The $OPERAS_{CC}$ approach to the ANTS mission

The swarm-based system in the ANTS mission can be directly mapped into the $OPERAS$ framework (Fig. 4). A number of agents of three different types (workers, W, leaders, L, and messengers, M) compose the MAS system. System configuration is highly dynamic due to its nature and unforeseen situations that may come up during the mission.

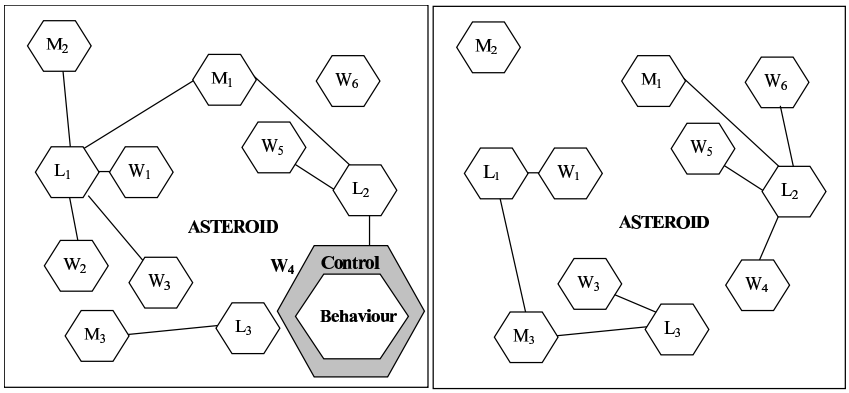

Fig. 4. (a) An instance of MAS structure corresponding to ANTS in Fig. 3 with an $OPERAS$ agent (W_4) consisting of separate Behaviour and Control components. (b) A change in the structure of MAS after possible events (e.g. destruction of worker W_2, leader L_1 employs worker W_6 etc.).

Leader: Formal Modelling of Behaviour in $OPERAS_{CC}$

For the modelling of the leader agent, one has to identify the internal states of the agent, its knowledge as well as the inputs it is capable of perceiving, so that they are represented as objects of the corresponding PPS.

The state of a leader can be either one of the three: *Processing* for an leader that is fully operational, $Malfunctioning$ for one that its facing problems and *Aborting* for one that is either facing irreparable problems or has been commanded by the control centre on earth to abort the mission.

Object	**Description**
$status$	The current operational state of the leader
$existingWorkers$	The set of IDs and statuses of the workers under its command
$existingMsgs$	The set of IDs and statuses of the messengers under its command
$results$	The set containing analysis results it has gathered
$model$	The current model of the agent's surroundings
$goals$	The agent's goals

Table 1. Objects representing the knowledge of the Leader agent.

The knowledge of the agent consists of the objects presented in Table 1 along with their description.

Similarly, the leader type of cell will be able to also perceive other objects representing input from the environment or from other agents. The most prominent ones are summarised in Table 2.

Object	**Description**
$abrt$	A request from the control centre that the agent should abort the mission
$worker$	A new worker that joins the team under the leaders command
$messenger$	A new messenger that joins the team under the leaders command
$requestForWorker$	A request for a worker, made by another leader (so that the worker is reallocated)
$requestForMsg$	A request for a messenger, made by another leader (so that the messenger is reallocated)
$message$	An object representing a message sent by another agent

Table 2. Objects representing the percepts of the Leader agent.

Indicatively, two of the operations that a leader may perform in the form of transformation rules follow.

The rule representing the joining of a worker w_i to the leader's team of $Workers$ is specified as:

$workerJoining$:
$(status : processing\ worker : w_i\ existingWorkers : Workers$
$\rightarrow status : processing\ existingWorkers : \{w_i\} \cup Workers)_L$

The newly allocated worker w_i may be received by another leader with the use of a communication rule of the form:

$receiveWorker : (message : canSendYouAWorker\ ;\ worker : w_i,\ in)_L$

which assumes that a $canSendYouAWorker$ message has been previously sent by the other leader informing that it is willing to reallocate one of its workers.

Similarly, the rule representing the reallocation of the messenger m_i to another leader is:

$reAllocatingMessenger$:
$(status : processing\ percept : requestForMsg\ existingMsgs : Messengers$
$\rightarrow status : processing\ existingMsgs : Messengers \backslash \{m_i\})_L,$
if $isMessengerNeeded(m_i) == false$

Worker: Formal Modelling of Behaviour in $OPERAS_{CC}$

Similarly for a worker agent, the internal states in which a it may be in are $Measuring$, when taking measurements from an asteroid, $Analysing$, when analysing the measurements in order to send results to its leader, $Idle$, $Malfunctioning$ and $Aborting$.

The knowledge of the agent consists of the objects presented in Table 3 along with their description.

Object	**Description**
$status$	The current operational state of the worker
$myLeader$	the identity of its commanding leader,
$teamWorkers$	The set of other coworkers belonging to the same team
$teamMsgs$	The set of messengers belonging to the same team
$Target$	The target asteroid
$Data$	The set of data collected from the asteroid
$Results$	The set of the data analysis results

Table 3. Objects representing the knowledge of the Worker agent.

The worker type of cell will also be able to also perceive other objects that represent either environmental stimuli or messages from other agents. Indicative ones are being summarised in Table 4.

Indicatively, some of the operations that a worker may perform in the form of transformation rules follow.

The rule representing the measurements' analysis mechanism of the worker is:

$analysingData$:
$(status : Analysing\ data : Data \rightarrow status : Idle\ results : Results)_W$

The rule that informs a worker that it is being reallocated to another leader is defined as:

$reAllocating$:
$(status : Idle\ myLeader : Leader\ reassignedTo : NewLeader$
$\rightarrow status : Idle\ myLeader : NewLeader)_W$

Object	**Description**
abrt	A request from the control centre that the agent should abort the mission
reassignedTo	The identifier of the new leader the worker is being reassigned to
data	The set of measurements taken from the asteroid

Table 4. Objects representing the percepts of the Worker agent.

4.3 Formal Modelling of Control in $OPERAS_{CC}$

According to $OPERAS_{CC}$, 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. According to $OPERAS_{CC}$ the above system would be defined as follows.

The set O contains all the aforementioned transformation rules that model the agents' behaviour as well as the reconfiguration rules (birth, death and bond-making) 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 all members of its team. More particularly O additionally contains the following rules.

The following birth rules create a new worker w_i or messenger m_i under the command of a leader L_i when the leader has received the corresponding messages (objects $earthSendsWorker$ and $earthSendsMsg$) from the control centre on earth.

$newWorkerFromEarth$:
$(status : Processing\ earthSendsWorker : w_i\ existingWorkers : Workers)_{L_i}$
$\rightarrow (status : Processing\ existingWorkers : Workers \cup \{w_i\})_{L_i}$
$(status : Idle\ myLeader : L_i)_{W_i}$

$newMessengerFromEarth$:
$(status : Processing\ earthSendsMsg : m_i\ existingMsgs : Messengers)_{L_i}$
$\rightarrow (status : Processing\ existingMsgs : Messenger \cup \{m_i\})_{L_i}$
$(status : Idle\ myLeader : L_i)_{M_i}$

Inputs, such as $earthSendsMsg : m_i$, from the environment are perceived with the use of communication rules of the form:

$receiveInput : (\varepsilon\ ;\ earthSendsMsg : m_i,\ enter)_L$

The death rule below removes agent instances that have aborted the mission from the model ($t \in S$ stands for any type of agent).

$abortion$:
$(status : aborting)_t \rightarrow \dagger$

Finally, the following bond-making rules ensure the creation of a communication bond between a leader agent (ε stands for the empty multi-set, i.e. no object is necessary) and any messenger or worker that belongs to this leader's team.
$workerBondMaking$:
$(L_i\ \varepsilon\ ;\ myLeader : L_i\ W)$

$messengerBondMaking$:
$(L_i\ \varepsilon\ ;\ myLeader : L_i\ M)$

The set P contains all objects recognised by the Population P System.

Regarding the environment E, it should initially contain objects representing the initial percepts for all agents.

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:

$R = \{(L_1, W_1), (L_1, W_2), (W_1, W_2), (M_1, L_1), (M_1, W_1), (M_1, W_2)\}$

Finally, the set S that contains the agent types is: $S = \{L, W, M\}$.

5 Conclusions and Further Work

We presented $OPERAS$ with which one can model multi-agent systems that exhibit dynamic structure, emergent and self-organisation behaviour. The contributions of $OPERAS$ can be summarised in the following:

- A formal framework for MAS modelling.
- The behaviour and the control of an agent are separate components which imply distinct modelling mental activities.
- Flexibility on the choice of formal methods to utilise and option to combine different formal methods.

It is because of this distinct separation between behaviour and control that $OPERAS$ provides this flexibility of choosing different methods for modelling

these two aspects; while some methods are better at capturing the internal states, knowledge and actions of an agent, others focusing on the dynamic aspect of a MAS are more suitable for capturing the control mechanisms.

In this paper, we employed Population P Systems with active cells to define $OPERAS_{CC}$, an instance of the general framework. We presented the $OPERAS_{CC}$ model of a swarm-based system of a number of autonomous spacecrafts. It could easily be spotted that an $OPERAS_{CC}$ model does resemble (as a final outcome) a model which could be developed if one used Population P Systems with active cells from scratch [24]. However, in the current context we have the following advantages:

- PPS can be viewed as a special case for $OPERAS$.
- The distinction of modelling behaviour and control as separate, offers the ability to deal with transformation/communication rules separately from cell birth/division/death and bond making, with implications both at theoretical as well as practical level.
- Practically, during the modelling phase, one can find advantages and drawbacks at any of the behaviour or control component and switch to another formal method for this component if this is desirable.

As far as the last point is concerned, we verified our initial findings [24] in which it was stated that modelling the behaviour of an agent with PPS rewrite and communication rules may be rather cumbersome. Especially in this rather complex case study, although the modelling of the control is absolutely straightforward, we had difficulties to establish the necessary peer to peer communication between agents by employing just the communication rules. That gave us the opportunity to consider alternatives. For example, we have experimented with Communicating X-machines which have a number of advantages in terms of modelling the behaviour of an agent. The resulting model, $OPERAS_{XC}$ [27], seems to ease the modelling process in complex MAS. It is worth noticing that none of the two formal methods (X-machines and Population P Systems) by itself could successfully (or at least intuitively) model a MAS [14, 26]. This is true for other formal methods too, which means the current framework gives the opportunity to combine those methods that are best suited to either of the two modelling tasks.

We would like to 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. Towards this direction, we are also currently working on various types of transformations that could prove its power for formal modelling as well as address legacy issues concerned with correctness.

Finally, efforts will also be directed towards enhancing existing animation tools on Population P Systems in order to come up with a new version of the tool that will be able animate $OPERAS_{CC}$ specified models. More particularly, the PPS-System [25] is a tool that generates Prolog executable code from Population P Systems models written in a particular notation. Future work will involve extend-

ing the notation and the system in order to integrate the necessary $OPERAS$ features, allowing us to gain a deeper understanding of the modelling issues involved with $OPERAS_{CC}$ and helping us investigate the practicability of our approach.

References

[1] A. Abraham, C. Grosan, and V. Ramos (Eds.). *Swarm Intelligence in Data Mining*, volume 34 of *Studies in Computational Intelligence*. Springer-Verlag, 2006.

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

[3] M. Benerecetti, F. Giunchiglia, and L. Serafini. A model-checking algorithm for multi-agent systems. In J. P. Muller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V*, Lecture Notes in Artificial Intelligence, pages 163–176. Springer-Verlag, 1999.

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

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

[6] F. Brazier, B. Dunin-Keplicz, N. Jennings, and J. Treur. Formal specification of multiagent systems: a real-world case. In *Proceedings of International Conference on Multi-Agent Systems (ICMAS'95)*, pages 25–32. MIT Press, 1995.

[7] L. Cardelli and A. D. Gordon. Mobile ambients. Number 1378 in Lecture Notes In Computer Science, page 140155. March 28 - April 04 1998.

[8] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *Proceedings of the 31st Hawaii International Conference on Systems*, January 1998.

[9] M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV*, volume 1365 of *Lecture Notes in AI*, pages 155–176. Springer-Verlag, 1998.

[10] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimisation by a colony of co-operating agents. *IEEE Transactions on Systems, Man and Cybernetics*, 26(1):1–13, 1996.

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

[12] M. Fisher and M. Wooldridge. On the formal specification and verification of multi-agent systems. *International Journal of Cooperating Information Systems*, 6(1):37–65, 1997.

[13] M. Gheorghe, editor. *Molecular Computational Models: Unconventional Approaches*. Idea Publishing Inc., 2005.

[14] P. Kefalas, I. Stamatopoulou, and M. Gheorghe. A formal modelling framework for developing multi-agent systems with dynamic structure and behaviour. In M. Pechoucek, P. Petta, and L. Z. Varga, editors, *Proceedings of the 4th International Central and Eastern European Conference on Multi-Agent Systems, Budapest, Hungary, September 15-17*, number 3690 in Lecture Notes in Artificial Intelligence, pages 122–131. Springer Verlag, 2005.

[15] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995.

[16] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.

[17] S. N. Krishna and Gh. Păun. P systems with mobile membranes. *Natural Computing: an international journal*, 4(3):255–274, 2005.

[18] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, i. *Information and Computation*, 100(1):1–40, 1992.

[19] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system: a toolkit for building multi-agent simulations. Working paper 96-06-042, Santa Fe Institute, Santa Fe, 1996.

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

[21] S. R. Rosenschein and L. P. Kaebling. A situated view of representation and control. *Artificial Intelligence*, 73:149–173, 1995.

[22] C. Rouf, A. Vanderbilt, W. Truszkowski, J. Rash, and M. Hinchey. Verification of NASA emergent systems. In *Proceedings of the 9th IEEE International Conference on Engineering Complex Computer Systems (ICECCS'04)*, pages 231–238, 2004.

[23] C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash. 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 (SEFM04)*, pages 24–33, 2004.

[24] I. Stamatopoulou, M. Gheorghe, and P. Kefalas. Modelling dynamic configuration of biology-inspired multi-agent systems with Communicating X-machines and Population P Systems. In G. Mauri, G. Păun, M. J. Pérez-Jiménez, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing: 5th International Workshop*, volume 3365 of *Lecture Notes in Computer Science*, pages 389–401. Springer-Verlag, Berlin, 2005.

[25] I. Stamatopoulou, P. Kefalas, G. Eleftherakis, and M. Gheorghe. A modelling language and tool for Population P Systems. In *Proceedings of the 10th Panhellenic Conference in Informatics*, Volos, Greece, November 11-13, 2005.

[26] I. Stamatopoulou, P. Kefalas, and M. Gheorghe. Modelling the dynamic structure of biological state-based systems. *BioSystems*, 87(2-3):142–149, February 2007.

[27] I. Stamatopoulou, P. Kefalas, and M. Gheorghe. OPERAS: a formal framework for multi-agent systems and its application to swarm-based systems. In *The 5th IEEE International Conference on Software Engineering and Formal Methods (SEFM'07)*, 2007. Submitted.

[28] T. White and B. Pagurek. Towards multi-swarm problem solving in networks. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, page 333, July 03-07 1998.