Using Membranes to Model a Multi-Agent System towards Underground Metro Station Crowd Behaviour Simulation

Ilias Sakellariou[1] and **Ioanna Stamatopoulou**[2] and **Petros Kefalas**[3]

Abstract. Simulation of crowd behaviours has long been an area of active research due to its high impact on urban area design. Rather recently, the problem has been tackled using agent based simulation (ABMS), a modelling approach that offers a more natural and flexible method for describing pedestrian behaviour. The current work, concerns ABMS modelling of passengers boarding an underground station using Population P Systems (PPS), a membrane computing paradigm that allows formal definition of an agent based system. It is argued that the expressive power of PPS allows modelling the simulation under study, and an implementation of the model described is provided in a well known agent simulation platform to further prove its validity.

1 INTRODUCTION

Crowd behaviour simulation has long been recognised as a valuable tool for evaluation of urban area design. Such simulations are considered more complex than those involving vehicle traffic mainly due to the fact that pedestrian navigational patterns are more freely defined and individuals are involved in more complex interactions both with the environment and other pedestrians. This fact led naturally to the introduction of agent based modelling (ABM) [4] along with other techniques discussed in a later section (section 2). The former supports a number of essential characteristics for the task, such as ability to host heterogeneous individuals in the simulation and emergence of complex behavioural patterns.

P Systems [19] is a membrane computing paradigm that is inspired by how living cells can perform computation. A P System is composed of a hierarchical structure of membranes that can transform objects through evolution rules that reside within them, and also exchange objects between them through communication rules. Membranes can dissolve or divide thus changing the configuration of the system during computation. A class of P Systems, Population P Systems (PPS) consist of cells within a bigger membrane (the environment). The cells are configured in a graph rather than a hierarchical tree structure and are connected through bonds under specific dynamic conditions. In PPS with active cells [2], cells can die or divide thus giving important features to PPS as a formal modelling language.

[1] University of Macedonia, 156 Egnatia Str, Thessaloniki 54006, Greece, email:iliass@uom.gr

[2] The University of Sheffield International Faculty, CITY College 3 Leontos Sofou Str., Thessaloniki 54626, Greece, email:istamatopoulou@city.academic.gr

[3] The University of Sheffield International Faculty, CITY College 3 Leontos Sofou Str., Thessaloniki 54626, Greece, email: kefalas@city.academic.gr

Prominent characteristics of PPS include:

- Non-deterministic communication between cells;
- Dynamic addition and removal of cells;
- Dynamic restructuring of the communications network; and finally
- Maximal and arbitrary parallelism, i.e. the ability to support selection of a *maximal/arbitrary* number of evolution rules with non-deterministic selection of the rule that applies at each computation step.

These features are particularly useful in MAS and this is the reason why we suggest that PPS with active cells is one of the most suitable formalisms for modelling the macro-level of biologically inspired MAS of a highly dynamic nature [13].

In this paper, we present how PPS with active cells, facilitates formal modelling of MAS targeted to crowd simulation. The aim is to demonstrate that PPS is a natural choice for this domain for developers who wish to formalise their system before proceeding to the implementation or simulation. Formalisatation is necessary when properties of the system at hand need to be proved. PPS provides an elegant way to describe simple rules for agent movement in space, multi-agent dynamic structure and organisation as well as exchange of messages. Such simple mathematical notation can be refined towards simulation and visualisation through an appropriate tool, in this case NetLogo.

The rest of the paper is structured as follows: Section 2 provides an overview of common approaches to crowd simulation; Section 3 presents the formal definition of PPS with active cells while Section 4 discusses the reasons why they are appropriate for MAS modelling; Section 5 presents the case study description together with indicative parts of the the PPS model, and Section 6 the derived NetLogo simulation; Section 7, finally, concludes the paper an discusses directions of further work.

2 CROWD SIMULATION

Pedestrian simulation has long been the topic of active research, due to its importance to urban design, as for instance evaluating a building design in case of an evacuation. There is a vast number of research results reported in the literature, that follow different approaches with respect to the method used and most importantly the granularity of the simulation, also referred to as scale.

A lot of research deals with the problem following a “macroscopic” approach, i.e. it uses for example *flow dynamics* as in [11] and [10] in order to assess designs and simulate behavioural patterns of pedestrians moving in an area of study. These models describe crowd behaviour as a set of partial differential equations and are most useful in densely populated environments.

Social force models, originally introduced in [9], consider pedestrians exposed to "force field" generated by its own intentions, other pedestrians and points of attraction. Models in this category have been thoroughly studied and in some cases combined with other approaches, as for instance agent based [14], to successfully model crowd behaviour. Social force models adopt the "microscopic" approach to crowd simulation, according to which the overall behaviour is emerging as the result of interactions of individual entities.

Cellular automata [3],[21] also fall under the same category, and describe the world as a grid of cells while behaviour is determined by a set of simple local rules that update the state of each cell, based on the state of its neighbouring cells. Gas lattice models consider pedestrians to be "particles" moving on a grid with a set of probability rules determining their next position on that grid, as for instance in [7] and [8].

Finally, *agent based modelling* (ABM) has also been employed to tackle the problem. ABM has a number of advantages [4] in the sense that it allows emergent phenomena to manifest, it is flexible and allows a natural description of the model. For instance in [6] ABM are used to model pedestrian behaviour and cellular automata to model the environment. A similar approach is used in [22] to assess design choices in cases of emergency evacuation in a metro station. Bandini et al., employ the situated cellular agent model [1] to model pedestrians in an underground train station that exhibit both cooperative and competitive behaviour, i.e passengers boarding and descending the train wagon.

The list of references provided above is by no means complete; there is a significant amount of research work and tools addressing the problem and the reader should refer to [17] and [23] for a more in depth review and assessment of models used in crowd simulation.

3 POPULATION P SYSTEMS WITH ACTIVE CELLS

3.1 Formal Definition of PPS with active cells

A Population P System with Active Cells [2] is defined as:
$PPS = (V,\ K,\ C_1,\ C_2,\ \ldots,\ C_n,\ w_E,\ \gamma,\ \alpha,\ R,\ O)$ such that:

- V is the set of all objects within the system, including those of the environment;
- K is the set of the different types of cells in the system (each cell has an associated class/type);
- $C_i, 1 \leq i \leq n$ are the n cells in the system;
- w_E is a multiset over V containing the objects initially assigned to the environment;
- $\gamma = (N, A)$ is the undirected graph representing the initial structure of the system, where: $N = \{1, 2, \ldots n\}$, and $A \subseteq N \times N$;
- α is a finite set of bond making rules;
- $R = R_e \cup R_c$ is the set of all evolution and communication rules;
- O is a partial order over the set of all rules R.

3.2 Objects in Cells and Environment

In the above definition, an object has its normal mathematical semantics, i.e. a symbol. For the purposes of this work however an object is represented as an $attribute\ :\ value$ pair and therefore $V = \{(a\ :\ v) \mid a \in \mathit{Attributes},\ v \in D\}$ where $\mathit{Attributes}$ is a set of attributes/labels and D stands for the *domain set* of a. In PPS that deal with spatial properties, there is also a special kind of object $(\Pi_i\ :\ \pi_i) \in V$ for each cell, denoting the ith cell's position, with values $\pi_i \in \mathbb{N}_0 \times \mathbb{N}_0$ and a label Π_i that stands for *position*. In [18], a special class of PPS is defined, namely Spatial PPS or ${}^{sp}PPS$.

3.3 Cells and Types

A cell is defined as a tuple $C = (w,\ t)$ where:

- w is the multiset of objects over V that are contained by the cell, and
- $t \in K$ is the type of the cell.

Practically, this means that cells are classified in different types containing different objects and different evolution and communication rules.

3.4 Bonds between Cells

The bond making rules in α are of the form $(t, (x\ :\ a); (y\ :\ b), p)$, where $(x : a), (y : b) \in V$ and $t, p \in K$. This rules states that in the presence of $(x : a)$ in a cell of type t and $(y : b)$ in a cell of type p, the two cells are joined, i.e. they can exchange objects. Bond making rules are responsible for constructing the undirected graph γ, which represents the communication structure of the P System.

3.5 Evolution Rules

R_e is the finite set of cell evolution rules, that is, rewrite rules which given the presence of certain object transform a cell by introducing new objects or communicating objects to other cells or transforming the type of a cell etc. Evolution rules determine the computation of the system and are of various types (in the following t and p are the types of the cell that these rules refer to):

- Transformation rules: $((x\ :\ a) \rightarrow (y\ :\ b))_t$, that is, an object $(x : a)$ becomes $(y : b)$;
- Differentiation rules: $((x\ :\ a))_t \rightarrow ((y\ :\ b))_p$, that is, in the presence of an object $(x : a)$ the cell of type t is transformed into a cell of type p containing the object $(y : b)$;
- Division rules: $((x\ :\ a))_t \rightarrow ((y\ :\ b))_t((z\ :\ c))_t$, that is, in the presence of an object $(x\ :\ a)$ the cell is divided into two cells of the same type. The two cells contain the objects $(y : b)$ and $(z : c)$ respectively;
- Death rules: $((x\ :\ a))_t \rightarrow \dagger$, that is, in the presence of an object $(x : a)$ the cell disappears from the system.

Evolution rules are shown diagrammatically in Fig. 1. The outer membrane in each diagram represents the environment whereas the inner membranes are the cells.

3.6 Communication Rules

Communication rules are responsible for importing/exporting objects from/to the environment as well as importing/exporting objects from/to other cells. Communication rules R_c are of the form (in the following t is the type of the cell that these rules refer to):

- $((x\ :\ a); (y\ :\ b), enter)_t$, that is, in the presence of an object $(x : a)$ the object $(y : b)$ is imported from the environment;
- $((x : a); (y : b), enter_{copy})_t$ that is, in the presence of an object $(x : a)$ a copy of the object $(y : b)$ is imported from the environment;

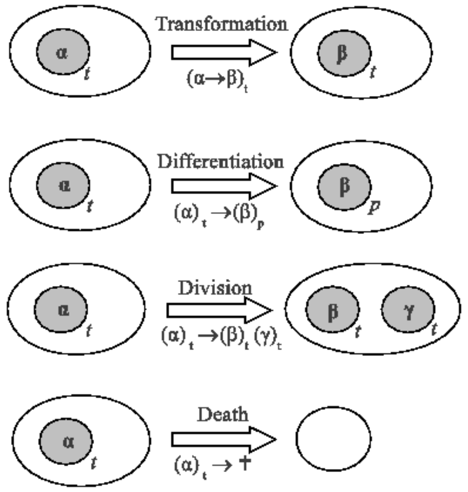

Transformation
$(\alpha \rightarrow \beta)_t$
Differentiation
$(\alpha)_t \rightarrow (\beta)_p$
Division
$(\alpha)_t \rightarrow (\beta)_t (\gamma)_t$
Death
$(\alpha)_t \rightarrow \dagger$

Figure 1. Evolution Rules in Population P systems with active cells.

- $((x : a); (y : b), in)_t$, that is, in the presence of an object $(x : a)$ the object $(y : b)$ is imported non-deterministically from a neigh boring cell;
- $((x : a), exit)_t$, that is, when present, the object $(x : a)$ is exported to the environment.

Communication rules are also shown diagrammatically in Fig. 2.

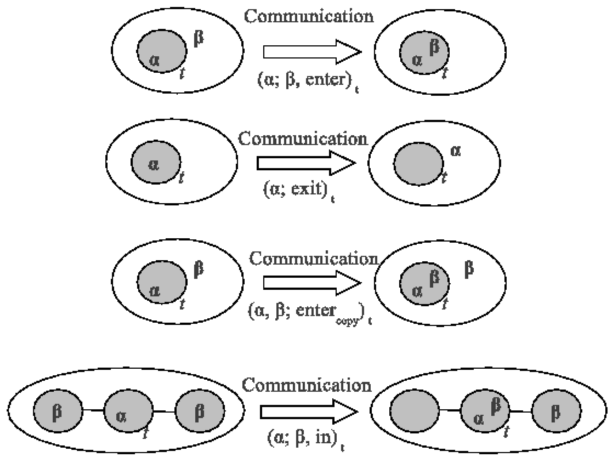


Figure 2. Communication Rules in Population P systems with active cells.

3.7 Computation

The computations step in a cell of a PPS consists of firstly applying in a non-deterministic order all the evolution rules that are triggered, and afterwards applying in a non-deterministic order the triggered communication rules. Since, however, this does not always serve the purposes of ABM, the partial order *O* augments the model, imposing a priority to the rules so that they applied in an order that fits the model at hand, e.g. $...r_i \succeq r_j....$ The computation in each cell is performed independently of other cells, as implied by the maximal and arbitrary parallelism characteristic of PPS.

4 POPULATION P SYSTEMS FOR MODELLING OF MAS

Formal methods such as PPS are particularly suited to modelling MAS [12]. Individual agents:

- perceive their environment by receiving stimuli as input;
- receive messages from other agents;
- revise their beliefs based on both the percepts as well as the information encoded in the received messages;
- react based on a specific set individual behavioural rules;
- engage in a deliberation process to decide on the next action;
- compile and send messages to other agents;
- act, with the effects of their action appearing in the environment.

Not all the above are present in every agent. For example, reactive agents do not deliberate, while "smarter" proactive agents do. Also, communication between simple biological agents is rather primitive and mostly done through the environment, in contrast to more elaborated direct communication that may follow a strict protocol. Therefore, in order to create a model of an agent, one would require non-trivial data structures, means of encoding rules and behaviour, representation of the internal state of the agent, etc. It is evident that PPS can provide such attributes in an agent (cell) through the set of objects and the set of evolution rules.

In a MAS, each agent operates in parallel with other agents, and a mode of interaction determines the way in which agents exchange messages. Agents have roles and are organised in a way that the communication flow is facilitated. The features of PPS are well suited for all the above; maximal/arbitrary parallelism, types of cells and bond making rules.

At MAS level, formal modelling would require ways to deal with exchange of messages between agents, a method for expressing the asynchronous computation of individuals, the addition and removal of agent instances "on the fly" and means for structuring and restructuring the organisation "on the fly" (structure mutation). Again, it is evident that PPS can provide such attributes for modelling MAS through communication rules, division and death rules as well as bond making rules.

Finally, in the current context of crowd behaviour, a MAS also requires special handling of spatial characteristics of agents, something that PPS can neatly provide. To summarise, PPS, through their characteristics, seems to offer an interesting and elegant way to formally model the features of spatial MAS such as those met in crowd behaviour.

5 CASE STUDY: METRO STATION CROWD BEHAVIOUR

The case study employed to demonstrate the application of the PPS approach concerns the simulation of underground metro station, as the latter is discussed in [1]. The scenario concerns a common everyday situation where passengers enter an underground station and try to board a wagon. The experimental setting includes an entrance area, where passengers initially entering the station are located, the platform, the wagon, the door area, an intermediate target for the passengers to reach before completing their boarding, and the seats (Fig. 3). Briefly, each passenger:

- Upon entering the station, selects one wagon door and moves towards it.
- The passenger then waits in the platform area for the doors to open.
- When the doors open the passenger moves to the door area in order to board the wagon.
- The passenger completes the boarding and is now located in the wagon.
- If the passengers perceives an empty seat they move towards it to get seated.

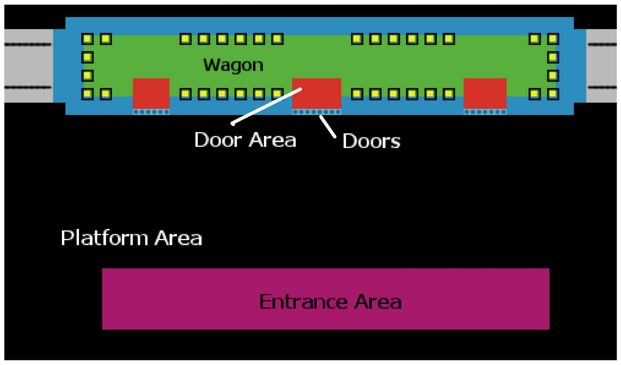


Figure 3. Areas in the underground simulation scenario.

For the experimental setting described above, we consider space to be discrete, divided to a grid, where each grid position can accommodate a single passenger. The latter move from a position to the next instantly. This is in accordance with almost all discrete models of pedestrian simulation reported in section 2. All the assumptions stated here are going to be reflected in the PPS model described in the following.

For the purposes of modelling the given scenario three distinct cell types are required. Cells of the first type $p \in K$ will be used to model the individual passengers. Each passenger cell should include objects that correspond to the area it is positioned, and beliefs, such as:

$(area : platformWaiting)(\Pi : (x, y))(goal : door_1)$

indicating that the particular passenger is currently waiting at the platform, in a particular position (x, y) with the goal of going towards the first door when the doors open.

Given a passenger in the above state, for them to perceive the doors have opened we need a communication rule that will retrieve the information (object $(doors : open)$) from the environment:

$((area : platformWaiting); (doors : open), enter_{copy})_p$

For the agent to move towards its intended direction, i.e. $door_1$ as specified by the $goal$ object, the neighbouring environment needs to be perceived for empty positions:

$((\Pi: (x, y))(goal: door_1);$
$\quad (door_1: (nx, ny))(empty: (nx, ny)), enter)_p$

such that (nx, ny) is a the neighbouring position that is closer or belonging to $door_1$.

After an empty position has been perceived, the passenger moves to this position (transformation rule), updating its environment to inform that its previous position is now empty (communication rule):

$((\Pi: (x, y))(empty: (nx, ny)) \rightarrow (empty: (x, y))(\Pi: (nx, ny)))_p$
$((empty: (x, y)), exit)_p$

Perceiving that it has arrived to the door area (communication rule), the agent updates its area (transformation rule):

$((\Pi : (x, y))(goal : door_1); (door_1 : (x, y)), enter_{copy})_p$
$((\Pi: (x, y))(area: platformWaiting)(goal: door_1)(door_1: (x, y))$
$\quad \rightarrow (\Pi : (x, y))(area : door_1)(goal : wagon))_p$

Passengers are removed from the system when they are in the entrance area exiting the station:

$((area: entranceExiting)(\Pi: (x, y))(entrance: (x, y)) \rightarrow \dagger)_p$

A second type of cell is required to model the arrival of new passengers at the entrance of the platform. The idea is that we have a generator cell (type $g \in K$) positioned at the entrance, that only contains one $divide$ object and two division rules. One rule will actually generate a new passenger at the entrance area whereas the other will be a "fake" division rule that does not generate a passenger. In this manner one of the two rules will be non-deterministically applied in each computation step possibly generating a new passenger. The two rules are:

$((divide) \rightarrow (divide)(\varepsilon))_g$, and
$((divide) \rightarrow (divide)$
$\quad ((state: entrance)(goal: dock)(\Pi: (x, y)))_p)_g$

where (x, y) is any position such that in the environment there are the objects $(entrance : (x, y))$ and $(empty : (x, y))$.

Finally, a third type of cells is required, to be responsible for the opening and closing of doors (type $d \in K$). One cell of the given type is necessary so that at specific intervals it expels to the environment the $(doors : open)$ and $(doors : close)$ objects with the use of two communication rules. To model the time intervals a transformation rule is used, which produces a copy of a $tick$ object at every step. Assuming, for example that doors have to open in 30 time units the rules would be:

$(tick \rightarrow tick^2)_d$, increasing the time (superscript denotes the multiplicity of the object in the multiset)

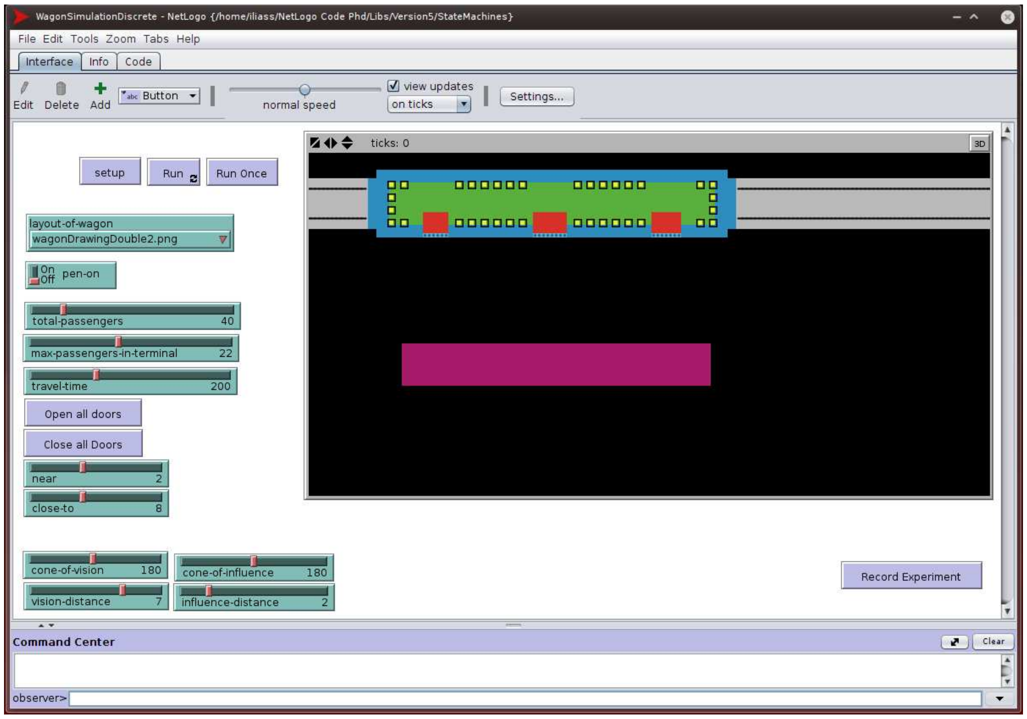
$(tick^{30} \rightarrow tick(doors : open))_d$, generating the $(doors : open)$ object and reseting the clock , and

$((doors : open); (doors : open), exit)_d$, for expelling to the environment so that the passengers are informed. A similar process takes place for closing the doors.

6 FROM FORMAL MODELLING TO SIMULATION

In the recent years a large number of agent simulation platforms have been proposed in the literature [15], differing in the programming language used, development environment, documentation, etc. Probably, Repast [16] and NetLogo [20] are the most widely adopted, and better supported platforms with a large community of active users. Out of the two, NetLogo was selected as the platform of choice for implementing the PPS model of the underground station, mainly due to its unparalleled simplicity in developing and maintaining simulation experiments.

NetLogo is an agent simulation platform, aiming at the simulation of complex social and biological phenomena, involving a large number of agents. The platform offers a complete simulation development environment, supporting the creation of a large number of agents (turtles) "living" in a 2D or 3D grid (patches), whose behaviour is controlled by a domain specific language. The platform



layout-of-wagon wagonDrawingDouble2.png
pen-on On
total-passengers 40
max-passengers-in-terminal 22
travel-time 200
near 2
close-to 8
cone-of-vision 180
vision-distance 7
cone-of-influence 180
influence-distance 2
view updates: on ticks

Figure 4. The underground station simulation scenario in NetLogo. Note that a number of controls allows modification of experiment parameters.

also supports easy GUI creation, thus greatly facilitating the development of a simulation experiment. It should be noted that it has been used as a tool for ABMS in a large number of cases, and in pedestrian simulation as well [5], [24]. Figure 4 depicts the NetLogo simulation environment of the case study of the paper.

Supporting direct execution of PPS defined agents in NetLogo presents a number of problems. Rules, as described in section 5, require support for one-way pattern matching (rewrite rules), a feature that is difficult to support in the functional-like domain specific language of NetLogo. Additionally, the declarative nature of the agent rule base presents a few difficulties when encoded in this programming paradigm. Therefore, we have chosen to implement the agent "PPS engine" in an NetLogo[4] extension, NetPrologo, that allows execution of Prolog code from the simulation platform.

This greatly facilitated the development of our agent model, since after the initialisation of the experiment, all relevant objects regarding the environment and the agents are asserted as Prolog facts, in the NetProlog engine. During simulation execution each agent runs the engine and updates its position and state according to the results of the execution.

Figure 5 presents a snapshot of the simulation showing agents boarding the metro wagon. Although in the figure all passengers are shown white, in the actual simulation, they are color coded depending on their state (seated, boarding etc.). In Fig. 6 all passengers have successfully boarded the metro wagon and some have managed to find a place to seat.

Although the NetProlog extension offers an easy and elegant way to implement the PPS model, we are considering the option of auto-

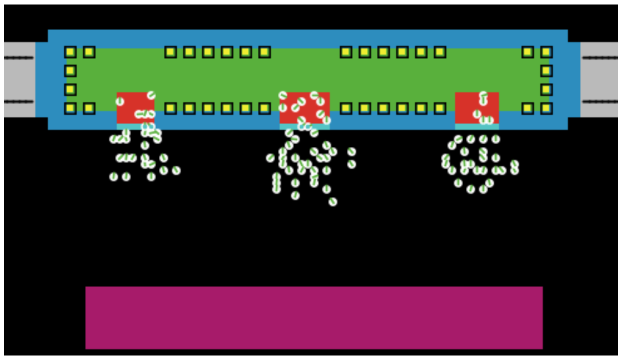


Figure 5. Passengers (shown in white) Boarding the Wagon.

[4] NetProlog is available at http://www.cs.us.es/˜fsancho/NetProLogo/

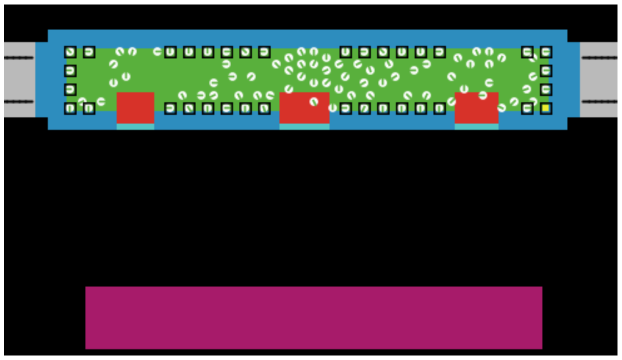


Figure 6. All Passengers on Board. Note that some passengers are seated, occupying the corresponding space in the wagon.

matically compiling rules to the NetLogo programming language, mainly for efficiency reasons, since it will reduce the overhead of updating the Prolog interpreter in each cycle.

7 CONCLUSIONS AND FURTHER WORK

With this work we aim at demonstrating how PPS with active cells is a suitable formalism for modelling MAS for the purposes of simulating crowd behaviour. We have used metro station boarding as an example to illustrate that PPS provide the necessary constructs both for modelling individual agents, with the use of objects that represent the agent's beliefs and evolution rules that update these beliefs, as well as for dealing with the structure reconfigurations that take place in such dynamic systems, with the use of bond-making, division and death rules. Additionally, although not explicitly demonstrated in the presented case study, PPS also allow the communication between agents through objects' exchange.

The initial NetLogo simulation described in this paper, acts as a proof of concept in using the PPS formalism to such a complex scenario. We are considering further developing tools and techniques for automatic translation of PPS specifications to NetLogo and possibly other platforms such as Repast, considering further issues such as simulation scalability, possibly through parallel/distributed execution. Additionally, we are considering the development of more complex pedestrian behaviours, as for example in panic situations in evacuation scenarios, where object exchange mentioned above could provide a flexible modelling approach.

REFERENCES

[1] S. Bandini, M.L. Federici, and G. Vizzari, 'Situated cellular agents approach to crowd modeling and simulation', *Cybernetics and Systems*, **38**(7), 729–753, (2007).

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

[3] V.J. Blue and J.L. Adler, 'Cellular automata microsimulation for modeling bi-directional pedestrian walkways', *Transportation Research Part B: Methodological*, **35**(3), 293–312, (2001).

[4] E. Bonabeau, 'Agent-based modeling: Methods and techniques for simulating human systems', *Proceedings of the National Academy of Sciences of the United States of America*, **99**(Suppl 3), 7280–7287, (2002).

[5] F. Camillen, S. Capri, C. Garofalo, M. Ignaccolo, G. Inturri, A. Pluchino, A. Rapisarda, and S. Tudisco, 'Multi agent simulation of pedestrian behavior in closed spatial environments', in *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pp. 375–380, (sept. 2009).

[6] J. Dijkstra and H. Timmermans, 'Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance', *Automation in Construction*, **11**(2), 135–145, (2002).

[7] M. Fukamachi and T. Nagatani, 'Sidle effect on pedestrian counter flow', *Physica A: Statistical Mechanics and its Applications*, **377**(1), 269–278, (2007).

[8] R.Y. Guo and H. J. Huang, 'A mobile lattice gas model for simulating pedestrian evacuation', *Physica A: Statistical Mechanics and its Applications*, **387**(2-3), 580–586, (Jan 2008).

[9] D. Helbing and P. Molnar, 'Social force model for pedestrian dynamics', *Physical Review E*, **51**, 4282, (1995).

[10] L. Huang, S. C. Wong, M. Zhang, C.W. Shu, and W.H.K. Lam, 'Revisiting Hughes' dynamic continuum model for pedestrian flow and the development of an efficient solution algorithm', *Transportation Research Part B: Methodological*, **43**(1), 127–141, (2009).

[11] R. L. Hughes, 'A continuum theory for the flow of pedestrians', *Transportation Research Part B: Methodological*, **36**(6), 507–535, (2002).

[12] P. Kefalas and I. Stamatopoulou, 'Modelling multi-agent systems: Experiences with membrane computing and future challenges', in *Proceedings of the 1st International Workshop on Application of Membrane Computing, Concurrency and Agent-based Modelling in Population Biology (AMCA-POP 2010)*, pp. 71–83, (2010).

[13] P. Kefalas and I. Stamatopoulou, 'Towards modelling of reactive, goal-oriented and hybrid intelligent agents using P systems', in *Proceedings of the 11th International Conference on Membrane Computing*, pp. 265–272, Berlin, Heidelberg, (2010). Springer-Verlag.

[14] Q. Lin, Q. Ji, and S. Gong, 'A crowd evacuation system in emergency situation based on dynamics model', in *Interactive Technologies and Sociotechnical Systems*, eds., Hongbin Zha, Zhigeng Pan, Hal Thwaites, Alonzo Addison, and Maurizio Forte, volume 4270 of *Lecture Notes in Computer Science*, 269–280, Springer Berlin / Heidelberg, (2006).

[15] C. Nikolai and G. Madey, 'Tools of the trade: A survey of various agent based modeling platforms', *Journal of Artificial Societies and Social Simulation*, **12**(2), 2, (2009).

[16] M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos, 'A declarative model assembly infrastructure for verification and validation', in *Advancing Social Simulation: The First World Congress*. Springer, Heidelberg, FRG, (2007).

[17] E. Papadimitriou, G. Yannis, and J. Golias, 'A critical assessment of pedestrian behaviour models', *Transportation Research Part F: Traffic Psychology and Behaviour*, **12**(3), 242–255, (2009).

[18] I. Petreska and P. Kefalas, 'Population p systems with moving active cells', in *Proceedings of the 12th International Conference on Membrane Computing, Fontainebleau, France*, pp. 421–432, (2011).

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

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

[21] H. Yue, H. Hao, X. Chen, and C. Shao, 'Simulation of pedestrian flow on square lattice based on cellular automata model', *Physica A: Statistical Mechanics and its Applications*, **384**(2), 567–588, (2007).

[22] N. Zarboutis and N. Marmaras, 'Searching efficient plans for emergency rescue through simulation: the case of a metro fire', *Cognition, Technology and Work*, **6**, 117–126, (2004).

[23] X. Zheng, T. Zhong, and M. Liu, 'Modeling crowd evacuation of a building based on seven methodological approaches', *Building and Environment*, **44**(3), 437–445, (2009).

[24] W. Zhu and H. Timmermans, 'Modeling pedestrian shopping behavior using principles of bounded rationality: model comparison and validation', *Journal of Geographical Systems*, **13**, 101–126, (2011).