MASSPA-Modeller: A Spatial Stochastic Process Algebra modelling tool

Marcel C. Guenther and Jeremy T. Bradley

Imperial College London, 180 Queen's Gate,
London SW7 2AZ, United Kingdom,
Email: {mcg05,jb}@doc.ic.ac.uk

Abstract. We introduce *MASSPA–Modeller*, a visual modelling tool for the recently developed spatial stochastic process algebra MASSPA which describes Markovian Agent Models (MAM)s. The major advantage of using a visual editor to generate MASSPA models is that the laborious task of modelling the communication between agents is partially automated by the tool. Furthermore the tool can separately check the correctness of local and spatial aspects of the model and thereby help users to find mistakes. For the analysis of the resulting models MASSPA–Modeller uses the powerful *GPA–analyser* engine. Additionally we briefly summarise the latest developments in spatial stochastic process algebras.

Keywords: Performance Analysis, Higher Moment Analysis, Spatial Stochastic Process Algebra, Spatial Modelling, MAM, MASSPA

1 Introduction

Performance analysis studies modelling techniques for predicting performance of computer and telecommunication systems. Moreover, it is also applied in other areas such as crowd modelling, systems biology and various other fields. Common analysis targets are the average time to complete a job or the probability distribution of client service times. Most models are described in high-level languages such as PEPA[1], a Stochastic Process Algebra (SPA), which translate to Continuous Time Markov Chains (CTMC)s. Alternatively Stochastic Petri nets can be used to describe CTMCs, but in this paper we will focus on process algebras. All CTMC models assume that any delay between two events in the underlying stochastic process is exponentially distributed.

Recent developments in fluid analysis methods [2,3] for CTMCs derived from high-level model descriptions such as SPAs have given modellers means to analyse extremely large models. Previously such models could only be analysed using stochastic simulation [4], as traditional analysis techniques, which are based on linear equation solvers, can only solve small models due to the well-known state space explosion problem. Fluid techniques exploit the fact that moments

Andrew V. Jones (Ed.): Proceedings of the 1st Imperial College Student Workshop (ICCSW '11), pp. 24–30. September 29th–30th 2011, London UK.

of stochastic processes, which describe state populations in lumped CTMCs, can be approximated by Ordinary Differential Equations (ODE)s. Today, there is a large number of formalisms for which the mapping from the high-level model description to ODEs has been defined, e.g. GPEPA [3], SCCP [5] and stochastic π-calculus [6]. More recent research has shown that it is even possible to approximate measures such as passage time distributions and reward vectors using ODEs [7,8].

Spatial modelling languages are useful extensions to their non-spatial counterparts, for instance when investigating crowd movements, disaster propagation or network topologies. Evaluation techniques for spatial models are similar to those for non-spatial models, however, spatial models tend to have larger CTMCs and allow analysis techniques that take into account the spatial nature of the model. Fluid analysis techniques have been applied to CTMCs that arise from spatial models described in Bio-PEPA [9] and the MAM formalism [10] for which we defined MASSPA, a Markovian Agent Spatial Stochastic Process Algebra [11]. The MAM formalism is a particularly interesting spatial modelling framework as it has been applied in a large number of different areas such as wireless sensor networks [10], fire propagation [12] and traffic modelling [13] to name but a few. Research in [11] shows that there is a generic mapping from MASSPA to a mass-action type reaction system. As a consequence any MAM model expressed in MASSPA grammar can now be analysed using the powerful *GPA–analyser*, which was originally developed for the analysis of massive GPEPA models [14].

Despite the existence of spatial SPAs (SSPA)s, complex spatial models can still be too large to be expressed in SSPAs by hand. To facilitate the creation of spatial models, visual editors such as DrawNet [15] and SeSam [16] have been developed for stochastic Petri nets and agent based simulations. Moreover, a recent extension to DrawNet enables users to describe the spatial aspects of MAMs, but it currently does not allow users to define sequential Markovian Agents [17]. MASSPA–Modeller, which is described in this paper, is the first spatial modelling tool for MAMs that allows users to define all aspects of MA models. It further enables users to perform higher moment and continuous reward vector analysis on spatial models. Moreover, it is the first hybrid tool that allows users to express local agent behaviour using SPA and spatial behaviour in a visual editor. This paper is organised as follows. First, we briefly describe the MAM formalism and MASSPA in Sect. 2. We then demonstrate the MASSPA-Modeller work-flow in Sect. 3 and finally present conclusions in Sect. 4.

2 MAM and MASSPA

The MAM formalism was first described by Gribaudo *et al.* in [10]. Each MAM consists of two parts, a definition for local agent behaviour and a model describing their distribution and interactions in space. Agents can evolve through local and message induced transitions. While local transitions are no different from

those in other formalisms, message induced transitions are novel. Every time an agent changes state it can emit a message of type M. At the same time another agent can listen for messages of type M and act on incoming messages. The perception function $u(\cdot)$, which is part of the model, defines all combinations of agents in all locations that can exchange messages (cf. [10,11]). In other words $u(\cdot)$ can be thought of as a directed graph where vertices are pairs of agent state populations and edges are message channels between them. Note that the MAM message exchange paradigm is an asynchronous form of communication, as receiving agents can decide to discard messages without blocking sending agents. Therefore Markovian Agents (MA)s are said to be autonomous. MASSPA [11], the Markovian Agent SSPA, simplifies the definition and evaluation of MAMs and makes comparisons with other formalisms easier. A MASSPA model consists of sequential agent, topology, agent population size and $Channel(\cdot)$ definitions (e.g. Fig. 1).

```
// Agent definitions
Agent OnOff {
  On  = !(2.0,M,1.0).Off;   Off = ?(M,1.0).On + (1.5).On;
};
// Spatial model definition
Locations = {(0),(1),(2)};
// Initial agent distributions
On@(0) = 100;   Off@(1) = 200;   Off@(2) = 150;
// Channel definition
Channel(On@(1),Off@(0),M) = 1/100;
Channel(On@(0),Off@(1),M) = 1/150;
Channel(On@(2),Off@(1),M) = 1/150;
Channel(On@(1),Off@(2),M) = 1/200;
```

Fig. 1. Simple MASSPA model for an `On`/`Off` agent model generated using MASSPA-Modeller. In this model agents in the `On` state emit `!(...)` messages of type `M` and agents in the `Off` state can turn `On` if they receive `?(...)` a message of this type or alternatively simply turn `On` at rate `1.5`. The perception function describes all message channels as well a scaling rate for each channel that will modulate the rate at which messages can be sent on a particular channel.

3 The MASSPA–Modeller

The example in Fig. 1 shows a fairly simple MASSPA model. However, as the number of agents states, locations and message types increases, the definition of the perception function can become large. Moreover, it becomes difficult to visualise the directions of channels if a model has hundreds or thousands of

channels. One of the main features of MASSPA–Modeller (`http://www.doc.ic.ac.uk/~mcg05`) is its ability to mitigate this by allowing users to define high-level channels such as: *Any state in location (0) can send any kind of message to any state in location (1).* Later this high-level channel description is used to auto-generate actual MASSPA channels.

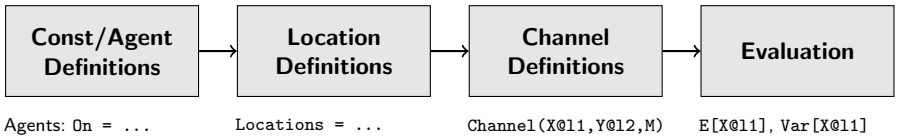

Fig. 2. MASSPA-Modeller work-flow

Having illustrated why a visual editor is necessary for the definition of complex MASSPA models, we now give a brief description of the MASSPA–Modeller work-flow, which is shown in Fig. 2. As a running example we will describe how the MASSPA model shown in Fig. 1 can be created in MASSPA–Modeller. The first step is to define constants, variables, functions and sequential agent definitions in the *Agents & Variables* tab. In our simple model we only define the latter, i.e. `Agent OnOff {On = !(2.0,M,1.0).Off; Off = ?(M,1.0).On + (1.5).On;}`, but in more elaborate models each transition rate could be represented as a expression of constants, variables and functions. Once this definition has been entered, the user can compile the agent definition. Errors and warnings will be displayed in the console below the editor tab.

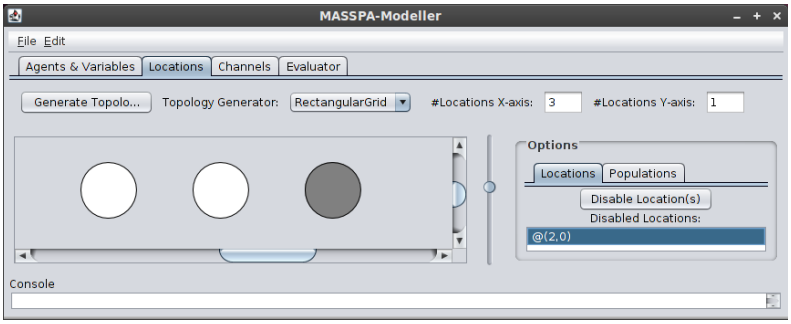

Fig. 3. Location definition in MASSPA–Modeller with one disabled location.

Having defined our agent(s), the *Locations* tab (see Fig. 3) can be used to generate different topologies such as rectangular or radial location grids. In our case we have a simple line consisting of 3 locations. Having generated a topology we can disable any unwanted location and define initial populations for every agent

state in each location. In our example we use this feature to create the following three initial populations `On@(0) = 100; Off@(1) = 200; Off@(2) = 150;`.

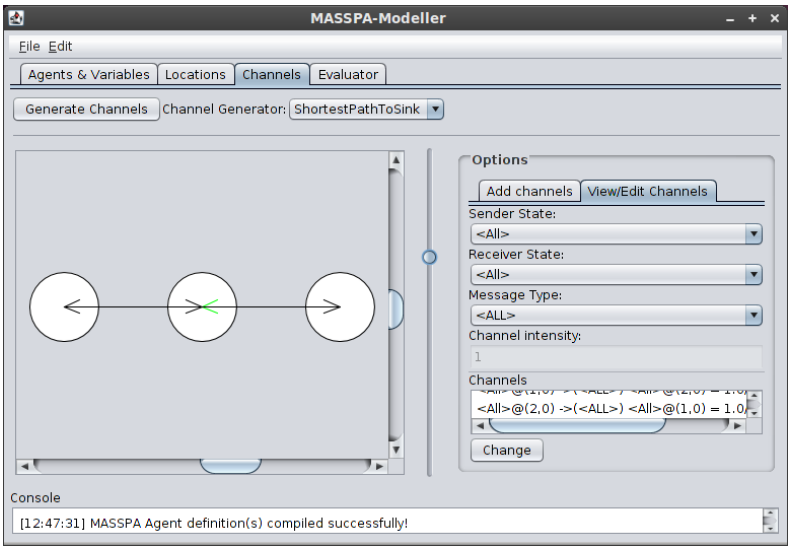

Fig. 4. Channel definition in MASSPA–Modeller.

Once users have create the topology and defined the initial populations, they can define the communication channels between locations or, if more fine grained communication control is needed, between specific sender and receiver populations (see Fig. 4). Channels can be generated using a predefined channel generator or by adding channels manually. Channel generators are useful for models with many locations, for instance when we want to create a source to sink style communication pattern with constraints on the maximum length of a single hop. In our example it is easiest to create the four channels manually in the editor.

The final tab allows users to generate the MASSPA model and to specify the evaluation method. Having generated the model, the evaluation method needs to be defined in GPA syntax [14], e.g. `ODEs(stopTime=60.0,stepSize=0.1, density=10,closure=MASSPA_infty){E[On@(2)],Var[On@(2)];}` uses fluid analysis to determine the mean and variance of `On` states in location `(2)` from time `0` to `60`. Pressing the *Evaluate* button will compile the model and use the GPA–analyser engine to perform the specified analysis and generate Fig. 5.

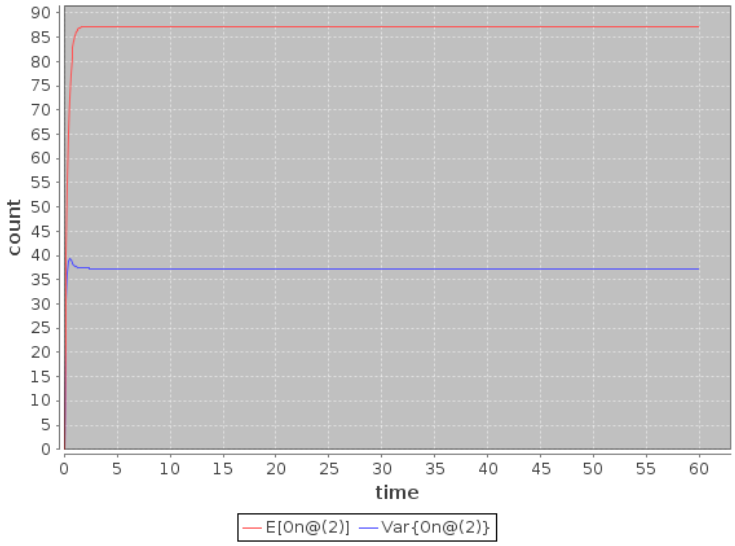

Fig. 5. Mean and variance for population `On@(2)`.

4 Conclusions

We have presented a new spatial modelling tool for MASSPA, which allows users to define all aspects of Markovian Agent models. Users can specify local agent behaviour using process algebra, while any spatial aspects of the model can be modelled using visual editors, which is more convenient than defining communication channels by hand. This combination between letting modellers define agents using process algebra and spatial aspects using an editor should be especially appealing to modellers who are familiar with non-spatial process algebras such as PEPA or GPEPA. In the future we might add this hybrid modelling approach to DrawNet, as this tool provides a much richer user interface for composing spatial models than MASSPA–Modeller. A particularly interesting challenge would be to create a WYSIWYG editor in DrawNet that allows users to define local agent behaviour as a labelled transition diagram or alternatively using MASSPA.

References

1. J. Hillston, "A Compositional Approach to Performance Modelling," *Cambridge University Press*, p. 158, 1996.
2. J. Hillston, "Fluid flow approximation of PEPA models," *Second International Conference on the Quantitative Evaluation of Systems QEST05*, pp. 33–42, 2005.

3. R. A. Hayden and J. T. Bradley, "A fluid analysis framework for a Markovian process algebra," *Theoretical Computer Science*, vol. 411, no. 22-24, pp. 2260–2297, 2010.
4. D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.
5. L. Bortolussi and A. Policriti, "Modeling Biological Systems in Stochastic Concurrent Constraint Programming," *Constraints*, vol. 13, no. 1-2, pp. 66–90, 2008.
6. A. Stefanek, *Continuous and spatial extension of stochastic pi-calculus*. Master thesis, Department of Computing, Imperial College London, 2009.
7. R. A. Hayden, A. Stefanek, and J. T. Bradley, "Fluid computation of passage time distributions in large Markov models," *submitted to Theoretical Computer Science*, 2010.
8. A. Stefanek, R. A. Hayden, and J. T. Bradley, "Fluid analysis of energy consumption using rewards in massively parallel Markov models," in *2nd ACMSPEC International Conference on Performance Engineering ICPE*, pp. 121–132, 2011.
9. V. Galpin, "Towards a spatial stochastic process algebra," in *Proceedings of the 7th Workshop on Process Algebra and Stochastically Timed Activities (PASTA)*, (Edinburgh), 2008.
10. M. Gribaudo, D. Cerotti, and A. Bobbio, "Analysis of On-off policies in Sensor Networks Using Interacting Markovian Agents," *6th IEEE International Conference on Pervasive Computing and Communications PerCom (2008)*, pp. 300–305, 2008.
11. M. C. Guenther and J. T. Bradley, "Higher moment analysis of a spatial stochastic process algebra.," in *8th European Performance Engineering Workshop - EPEW 2011*, 2011.
12. D. Cerotti, M. Gribaudo, A. Bobbio, C. Calafate, and P. Manzoni, "A Markovian Agent Model for Fire Propagation in Outdoor Environments," in *7th European Performance Engineering Workshop EPEW* (A. Aldini, M. Bernardo, L. Bononi, and V. Cortellessa, eds.), vol. 6342 of *Lecture Notes in Computer Science*, pp. 131–146, Springer Berlin Heidelberg, 2010.
13. D. Cerotti, M. Gribaudo, and A. Bobbio, "Presenting Dynamic Markovian Agents with a road tunnel application," in *IEEE International Symposium on Modeling Analysis Simulation of Computer and Telecommunication Systems MASCOTS*, pp. 1–4, IEEE, 2009.
14. A. Stefanek, R. Hayden, and J. Bradley, "A new tool for the performance analysis of massively parallel computer systems," *Eighth Workshop on Quantitative Aspects of Programming Languages QAPL 2010 March 2728 2010 Paphos Cyprus*, 2010.
15. A. Baravalle, G. Franceschinisy, M. Gribaudo, V. Lanfranchi, M. Iaconoth, N. Mazzoccath, and V. Vittoriniu, "DrawNET Xe: GUI and Formalism Definition Language," *Informatica*, 2003.
16. F. Klügl, R. Herrler, and M. Fehler, "SeSAm: Implementation of Agent-Based Simulation Using Visual Programming," *Components*, no. May, pp. 1439–1440, 2006.
17. D. Cerotti, E. Barbierato, and M. Gribaudo, "A tool suite for modelling spatial interdependencies of distributed systems with Markovian Agents," tech. rep., 2011.