Theory and Application of Multi–Formalism Modeling

Marco Gribaudo
Politecnico di Milano, Italy

Mauro Iacono
Seconda Università degli Studi di Napoli, Italy

A volume in the Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC) Book Series
Chapter 8

GPA: A Multiformalism, Multisolution Approach to Efficient Analysis of Large-Scale Population Models

Jeremy T. Bradley  
Imperial College London, UK

Marcel C. Guenther  
Imperial College London, UK

Richard A. Hayden  
Imperial College London, UK

Anton Stefanek  
Imperial College London, UK

ABSTRACT

This chapter discusses the latest trends and developments in performance analysis research of large population models. In particular, it reviews GPA, a state-of-the-art Multiformalism, Multisolution (MFMS) tool that provides a framework for the implementation of various population modelling formalisms and solution methods.

1 INTRODUCTION

A decade ago Sanders (Sanders 1999) noted that in spite of research advances in the performance modelling and analysis field, the work of performance analysts has by no means become easier since both the expectation in their work as well as the complexity of systems to be evaluated have also grown considerably. To equip performance analysts with expressive modelling formalisms and efficient solution methods, Sanders suggested a multiformalism, multisolution (MFMS) paradigm that would not only allow modellers to model systems in a natural composite manner using different formalisms in the same model, but also provide them with different analysis options for such heterogeneous composite models.

While MFMS research and tools development have led to a better range of software products for performance analysts, there are still quite a lot of open research challenges in the performance...
community with regards to improving formalisms and their solution techniques. Population models, which are the focus of this chapter, are one such area that has recently received a lot of attention in the literature due to the growing need to model and analyse crowd behaviour (Massink et al. 2011), biological systems (Ciochetta & Hillston 2009) as well as large, distributed communication systems with thousands of network participants (Stefanek et al. 2010). The challenge in analysing these models is the fact that the state-space of population models increases exponentially in the number of interacting individuals/agents, making it computationally expensive to solve such models using traditional Monte Carlo simulation techniques. Moreover, even moderate population models with only a few hundred agents often exceed the capabilities of traditional numerical state-space avoidance and largeness tolerance solution methods. As a consequence, novel mean-field analysable formalisms have been developed, which can efficiently handle models with large populations. The efficient mean-field/fluid analysis techniques look at the models from a macroscopic point of view instead of treating every component at an individual level. By aggregating the behaviour of individual components, it is often possible to derive a set of Ordinary Differential Equations (ODEs) whose solution expresses the evolution of probabilistic measures such as the means and higher order moments of populations.

Grouped PEPA Analyser (GPA) (Stefanek et al. 2010) is an advanced software solution for population modelling and mean-field analysis. Originally developed for the analysis of the Grouped PEPA process algebra (GPEPA) (Hayden & Bradley 2010, Hillston 2005) in 2009, the tool has since been extended to support a range of different population modelling formalisms and solution methods. As this change continues, the tool is slowly embracing more and more MFMS principles, with a strong focus on population modelling formalisms. In this chapter we give an overview of mean-field analysable population models and describe how the architecture of GPA facilitates the implementation of new formalisms and solution techniques for such models. The chapter is organised as follows; In Section 2 we define population modelling, introduce different classes of mean-field analysable formalisms in Section 2.1 and formally describe a Population CTMCs, the central intermediate representation used by GPA, in Section 2.2. In addition to this, Section 2.3 reviews related MFMS and population modelling tools. Section 3 and 4 describe the architecture of GPA and show how different formalisms and solution techniques were implemented. Finally, Section 5 discusses future extensions for GPA and we present our conclusions in Section 6.

2 BACKGROUND

Population models describe interactions between individuals, which are grouped into populations. Individuals can represent a number of different entities or agents such as people, telecommunication equipment or vehicles to name but a few. While the individual behaviour of agents can be described using a small set of rules, the simulation of population models becomes infeasible when looking at the interaction of thousands or millions of individuals. However, when grouping a large number of individuals into populations, it is possible to evaluate the effects of interactions using efficient mean-field analysis techniques rather than simulation.

2.1 Mean-Field Analysable Population Models

Before we discuss the population model classes that are currently supported by GPA, we would like to give a brief overview of different population model classes and measures that can be calculated with mean-field fluid techniques. To distinguish between different classes of population models, we consider different possibilities of
• Agent state-space.
• State sojourn time distribution.
• Distribution of the next state after each transition.
• The method of accumulation and feedback.

From the perspective of an individual, the state-space can be discrete, continuous or mixed. For instance, if we were to model a single mobile device, its state-space might consist of a continuous variable for the remaining battery charge, as well as a set of discrete states to keep track of whether the device is idle or in communication mode. Thus the state of an individual with discrete modes from a set D and continuous attributes can be expressed as a tuple in $D \times \mathbb{R}^c$ or a state-space of $(D \times \mathbb{R}^c)^n$ for a population of n devices. The abstraction from individual to population specific measures enables us to use a more compact state-space representation of $(\mathbb{N} \times (\mathbb{R}^c \to \mathbb{R}))^d$. Then each state is a d tuple with the i-th element being a tuple $(n_i, f_i)$ where $n_i$ is the number of devices in state $d_i$ and $f_i(x_1, \ldots, x_c)$ is the distribution function over the continuous quantities.

Throughout this chapter, we consider models that evolve in continuous time. The sojourn time is the time it takes for an agent to transit from one discrete mode to another. Traditionally, this is an exponentially distributed random variable, enforcing the Markov property and enabling various efficient analysis techniques. Furthermore these can be readily be generalised to the cases where the sojourn time follows a phase type distribution. However, realistic examples often require further distributions that cannot be efficiently represented as a phase type distribution. For example, deterministic and uniformly distributed delays and delays from distributions with heavy tails would require intractably large phase type representations. This forms a distinction between individual agents being represented by a Continuous Time Markov Chain (CTMC) in the exponential/phase type case and by a Generalised Semi-Markov Process (GSMP) in the general case. Moreover, in some models the sojourn time for an individual agent is undefined and only determined globally when the agent is synchronised with another agent.

After each transition, an agent changes its state from one mode to another. The second mode could follow a deterministic, probabilistic or a non-deterministic choice. In this framework, we directly represent only a deterministic choice. However, the probabilistic choice can in some cases be emulated by adding additional transitions with appropriately weighted rates.

In case the agent state space contains continuous variables, the way these evolve over time has to be defined. Traditionally, each of the individual agent continuous variables accumulates linearly over time, with rate depending on its current mode. These could also evolve in a different way, such as with non-linear rates or with added stochastic noise. In addition to individual agent continuous variables, models can feature global continuous variables, for example representing temperature, the total cost of the system etc. Both local and global continuous variables may also feed back into the sojourn time distribution parameters and affect the distribution of agent transitions. Time-inhomogeneous behaviour can be seen as a special case, with time being represented by a continuous variable evolving linearly with rate $\lambda$.

Not all the combinations of the above possibilities have been considered, especially in the context of mean-field/fluid analysis. Some examples of existing work include:

1. The agent state space is discrete, sojourn times are exponential and next mode chosen deterministically. Examples include the PEPA/GPEPA process algebras, chemical equations, BioPEPA, MASSPA spatial process algebra, Stochastic Petri Nets. It is possible to derive ODEs describing means (Hillston 2005) and higher moments (Hayden & Bradley 2010) of populations.
Additionally, the discrete state space is extended with global continuous state space, where the continuous variables accumulate in between discrete transitions according to a system of ODEs. It is possible to extend the ODEs for the discrete model with additional equations capturing the means and higher moments of the continuous variables - both in case of no feedback (Stefanek, Hayden & Bradley 2011) and feedback from the continuous variables (Stefanek et al. 2012).

3. The sojourn times can be deterministic in addition to exponential. It is possible to derive Delay Differential Equations for the mean of populations (Hayden 2012a, Bortolussi & Hillston 2012a).

4. Individual agents can contain continuous variables. It is possible to derive Partial Differential Equations (PDEs) for the mean populations and distribution of the values of the continuous quantities (Hayden 2012c, Chaintreau et al. 2009).

The GPA tool generalises techniques listed in (i) and (ii). It uses an intermediate representation, so-called Population Continuous-Time Markov Chains (PCTMCs), that capture features of the formalisms mentioned in (i). This representation is augmented with continuous variables to capture the formalisms in (ii). GPA implements the efficient mean-field/fluid techniques for PCTMC models. In this chapter we will provide an overview of how different formalisms can be translated into this intermediate representation and how the techniques can be used to answer additional questions about the models.

### 2.2 PCTMC

Population Continuous-Time Markov Chains (PCTMCs) consist of a finite set of populations $S$, $n = |S|$ and a set $E$ of transition classes. States are represented as an integer vector $\tilde{P}(t) = (P_1(t), \ldots, P_n(t)) \in \mathbb{N}^n$, with the $i$-th component being the current population level of species $S_i \in S$ at time $t$. A transition class $e \in E$ is a tuple $(r_e, \tilde{c}_e) \in E$ that describes a transition with negatively exponentially distributed delay $D$ at rate $r_e : \mathbb{Z}^n \rightarrow \mathbb{R}_+$, which sets the population vector $\tilde{P}(t + D)$ to $\tilde{P}(t) + \tilde{c}_e$. The analogue to PCTMCs in the systems biology literature are Chemical Reaction Systems, were $\tilde{P}(t)$ describes a molecule count vector and transition classes represent chemical reactions between the molecules with being the reaction rate function and $\tilde{c}_e$ the stoichiometric vector for a specific reaction. For notational convenience we write a transition class (or an event) as:

$$S_{i_1} + \ldots + S_{i_h} \rightarrow S_{j_1} + \ldots + S_{j_k} \text{ at } r_e(\tilde{P}(t))$$

(1)

where $S_i \in S$ represent different species that are involved in the event - of them on left hand side and $S_j$ on the right hand side of the above equation. The corresponding change vector is $\tilde{c}_e = (s_{i_1}^{\text{out}} - s_{i_1}^{\text{in}}, \ldots, s_{i_n}^{\text{out}} - s_{i_n}^{\text{in}}) \in \mathbb{Z}^n$ where $s_{i}^{\text{in}}$ represents the number of occurrences of a particular species on the left hand side of the event and the number of its occurrences on the right hand side. The rate of each transition depends on the current population level $\tilde{P}(t)$ and is equal to:

$$r_e(\tilde{P}(t)) \begin{cases} 0 & \text{if } P_i(t) < s_i^{\text{in}} \text{ for all } i - 1, \ldots, n. \\ \frac{r_e(t)}{P_i(t)} & \text{otherwise} \end{cases}$$

(2)

An important feature of PCTMC models is that approximations to the evolution of population moments of the underlying stochastic process can be represented by the following system of ordinary differential equations (ODEs) (Hayden & Bradley 2010, Hillston 2005, e.g.)
\[
\frac{d}{dt} \mathbb{E} \left[ T(\bar{P}(t)) \right] = \sum_{c \in \mathcal{C}} \mathbb{E} \left[ \left( T(\bar{P}(t) + \epsilon_c) - T(\bar{P}(t)) \right) c(\bar{P}(t)) \right]
\]

where \( T : \mathbb{Z}^n \rightarrow \mathbb{R} \) is a real valued function defined on the populations.

To obtain the ODEs describing the evolution of the mean of a population, all we need to do is to substitute \( T(P_i(t)) = P_i(t) \) in the above system of equations, where \( P_i(t) \) is the random variable representing the population count of species at time \( t \). The right hand side of this ODE can depend on further means (after some approximating assumptions are considered) which have to be included in the final system of ODEs. In the literature the resulting system of ODEs is often referred to as the mean-field approximation. Similarly, ODEs for higher joint moments can be obtained by choosing adequate \( T(\bar{P}(t)) \), e.g. \( T(\bar{P}(t)) = P_i(t) - \mathbb{E}[P_i(t)]^2 \) for the variance of \( P_i(t) \). Alternatively stochastic simulation (Gillespie 1977) can be used to evaluate PCTMCs. Like discrete event simulation for low-level protocol models, this latter simulation technique captures the stochastic behaviour of PCTMCs exactly, but does not scale for models with larger populations.

### 2.3 Related Tools

Before describing the features and architecture of GPA in detail, we ought to review existing population and multiformalism modelling and analysis tools. In doing so we also provide a high-level comparison between GPA and other popular tools that have been developed by the research community over the last decade. In their purest form, multiformalism modelling and solution tools should allow modellers to compose models build from smaller sub-models, each of which may have been defined in its own specific formalism (Sander 1999). Subsequently the composite model is analysed by the software in order to work which solution methods can be used to evaluate the performance measures selected by the user. Currently, the most sophisticated tool that follows this paradigm is Möbius (Deavours et al. 2002). Similar tools, although less well-known and versatile, are DrawNET++ (Gribaudo et al. 2005), SHARPE (Sahner et al. 1997) and SMART (Ciardo & Miner 1996). Tools such as GPA (Stefanek et al. 2010) and PRISM (Kwiatkowska et al. 2011) belong to a slightly less generic class of multiformalism, multisolution (MFMS) tools, as their focus is less on the evaluation of compositional models containing sub-models expressed in different formalisms. Instead, they provide a modelling and evaluation framework that easily allows programmers to add a range of formalisms and re-use existing solution methods. The reason for this difference is mainly because both GPA and PRISM were designed as specialised frameworks with a focus on large population models and probabilistic model checking, respectively. What is interesting though is that the current evolution of GPA is currently moving into the direction of the original multiformalism, multisolution modelling paradigm, suggested by Sanders (Sanders 1999).

To see why this is the case we need to briefly review the architecture of the Möbius framework. Möbius is currently the most powerful MFMS tool because of its well-defined, generic Abstract Functional Interface (AFI) (Deavours et al. 2002) in which virtually any low-level formalism, e.g. CTMCs, GSMPs, Coloured Stochastic PetriNets, can be implemented. Through the AFI, any two sub-models written in a formalism that is implemented using the interface, can be combined into a larger composite model and analysed using a single solver. This is made possible by properties, i.e. a form of meta-data, which are added to each formalism implementation of the AFI. Depending on its underlying formalism and nature, a sub-model exposes properties that determine suitable solution methods. Naturally, any model expressed in any formalism can be analysed using
stochastic simulation techniques. However, other, more efficient solvers, can be used on models that satisfy certain properties, for instance models that translate to CTMCs. When analysing composite models with two or more sub-models, Möbius can algorithmically find all suitable solvers by taking into account the nature of the composition and the individual properties of the sub-models. While this maximises the re-use of solvers for different formalisms, it also means that any novel properties required by newly implemented solution algorithms have to be added to the existing formalisms supported by Möbius.

While the PCTMC formalism that represents the current low-level interface of GPA is much less abstract than the AFI, it is still true to say that the low-level population formalism of GPA is slowly becoming more generic. While Möbius was designed to be a multiformalism, multisolution tool from the very beginning, the growing abstractness of GPA’s low-level formalism is an evolutionary development that is driven by the recent surge in research on fluid analysis techniques for large population models in the performance community. GPA started as a tool for the evaluation of the GPEPA formalism (Hayden & Bradley 2010) and was only later extended to cater for other formalisms such as MASSPA (Guenther & Bradley 2011), the Chemical Reaction style formalism (Section 2.2) and Unified Stochastic Probes defined over the GPEPA process algebra (Kohut et al. 2012). Ultimately, future versions of GPA will feature even more generic population formalisms than PCTMCs in order to support a range of non-Markovian population formalisms (Section 5) as well as composite models. As part of this change, a Möbius inspired property metadada language for efficient application of different solution methods across different formalisms, is likely to be added.

Having discussed different MFMS tools, we also need to look at other population modelling and analysis frameworks that support similar models as GPA does. Most notable are the tools that were written for the evaluation of Bio-PEPA (Ciocchetta & Hillston 2009), SCCP (Bortolussi & Policriti 2008) and Markovian Agent Models (MAM) (Gribaudo et al. 2008).

Bio-PEPA is an evolution of the PEPA process algebra aimed towards modelling of biological systems at various levels of abstraction, such as bio-chemical networks (Ciocchetta et al. 2010), epidemiological models (Ciocchetta & Hillston 2010) and emergent behaviour in crowd dynamics (Massink et al. 2011). The Bio-PEPA language is supported by the Bio-PEPA tool suite (Ciocchetta et al. 2009), consisting of the Bio-PEPA Eclipse plugin and the Bio-PEPA workbench. The Bio-PEPA Eclipse plugin contains a full tool chain that allows modellers to specify Bio-PEPA models and analyse them with fluid analysis and simulation. The Bio-PEPA workbench is an experimental tool that provides researchers with a way to export Bio-PEPA models into further representations that can be used with a range of simulators and ODE solvers, or for example the PRISM model checking tool. Some work has been done to show how Bio-PEPA can be used as an intermediate formalism, thus allowing the solution techniques provided by the Bio-PEPA tool suite to be applied to a wider range of models. For example, a subset of the models specified in the Systems Biology Graphical Notation Process Description can be translated into Bio-PEPA (Loewe et al. 2011).

While some of the tools like Bio-PEPA offer additional solution methods for PCTMCs that are currently not available in GPA, these could be added in principle. Moreover, much like it was shown how the MAM formalism can be implemented in GPA (Guenther & Bradley 2011), it would also be possible to add Bio-PEPA and SCCP grammars.
3 POPULATION MODEL DEFINITION AND ANALYSIS IN GPA

The high level architecture of GPA is shown in Figure 1. The central representation is based on the PCTMC formalism introduced in Section 4.2, to which all the other supported formalisms are translated. Currently, the supported formalisms are: a simple language similar to chemical equations, the GPEPA process algebra and MASSPA algebra for spatial models.

All formalisms use a common specification interface which defines the format of the input to GPA. The initial section contains a definition of all the parameter constants in the model. This is followed by the model definition which is specific to the formalisms. Defined models can be analysed using different explicitly specified solution techniques, described in Section 4. The analysis can compute various metrics of interest that are based around the moments of the populations in the model. In case of the GPEPA formalism, GPA additionally provides and implementation of the Unified Stochastic Probes formalism. This allows modellers to specify complex passage time measures based on the model behaviour. The GPEPA model is automatically augmented with additional states that are translated into new PCTMC populations.

Moments of these populations are then implicitly used to produce the desired passage time
probabilities. We describe the USP formalism and the extended analysis process in Section 4.2.2.

Furthermore, GPA extends the solution techniques to allow the analysis to produce metrics based around accumulations defined on the population levels. These can be useful when modelling continuous quantities such as energy consumption or temperature.

Finally, the metrics from the analysis can form objective functions and constraints in parameter optimisation experiments introduced in Section 4.2. In general, the resulting global optimisation problems are hard and computationally not feasible. However, when the mean-field analysis solution method is chosen, even naive sweeping algorithms can explore a large range of parameters. The architecture shown in Figure 1 is designed to support future extensions of PCTMCs (Section 5) such as population models with deterministically (Hayden 2012a) or generally timed delays (Hayden 2011) or rates that depend on a feedback from reward measures (Stefanek et al. 2011). As shown in Figure 1, the input of GPA starts with a declaration of all used parameters. These are of the form:

\[
\begin{align*}
\text{alpha} &= 0.1; \\
\text{beta} &= 0.2; \\
\end{align*}
\]

3.1 Simple Chemical Equations

GPA supports a formalism that directly represents PCTMCs in form of equations as in Equation (1). Each model is defined by a set of such equations and a list of initial population values. The population names used in the equations, rates and initial values definitions are uppercase strings enclosed in braces, such as \{S\}. Figure 2 shows a GPA definition of a model of a circadian clock (Engblom 2006). There are populations and equations (transition classes). At time \(t=0\), all the populations are except for \{Da\} and \{Dr\} equal to 0.

After the definition, various types of analysis can be performed on the model. For example, the underlying PCTMC can be simulated to obtain sample traces of population evolution or statistics from a large number of simulation traces. Figure 3 shows an example of a single trace and a mean of populations obtained from simulation traces.

Alternatively, GPA implements efficient fluid analysis solution techniques introduced in Section 4.1.2. Figure 4 shows the means of species for the same parameter values as the simulation analysis in Figure 3a. This model is an example where the first order mean-field analysis is not sufficient to capture oscillations in the populations and a second order analysis is required.

---

Figure 2. GPA syntax of a simple chemical equations model of a circadian clock
GPA has originally been developed as an implementation of the higher order fluid techniques in (Hayden & Bradley 2010). GPA models can be described in the Grouped PEPA (GPEPA) process algebra. The input file syntax closely follows the formal definition of GPEPA. Processes can be defined as a choice between a number of actions (the operator) or synchronised in parallel by the operator where is a set of actions:

\[
S := (\alpha, r) | S | S + S | C_s \\
P := P \triangleright \triangleleft P \triangleright C
\]

Figure 3. Single trace and a mean of populations of species R and C obtained from simulation traces

Figure 4. Means of species R and C for the same parameter values as the simulation analysis in Figure 3 (a)

3.2 GPEPA

GPA has originally been developed as an implementation of the higher order fluid techniques in (Hayden & Bradley 2010). GPA models can be described in the Grouped PEPA (GPEPA) process algebra. The input file syntax closely follows the formal definition of GPEPA. Processes can be defined as a choice between a number of actions (the operator) or synchronised in parallel by the operator where is a set of actions:

The initial part of the model definition contains a number of component definitions, written as equations with the component label on the left hand side and the corresponding PEPA process on the right.

The system equation is defined as a composition of different labeled groups. Each group contains a large number of identical components that do not directly communicate. Different types of components in the group are separates by the operator. Groups can be composed in parallel with the PEPA operator:

\[
G : G\{P[n] \ldots | P[n]\} \\
M : M \triangleright \triangleleft M
\]
where \( n \) is an integer and \( G \) a group label. For example, Figure 5 shows a simple Client/Server model. The model consists of a large number of clients and servers. Each client repeatedly requests some data from the group of servers and subsequently receives the data and performs an independent operation on the data. The servers answer requests and provide the data and are susceptible to occasional failures.

Figure 6 shows the means and standard deviation of some of the populations in the Client/Server model. This example demonstrates an improvement in accuracy when a second order closure based on the normal distribution is used.

We can inductively define a PCTMC semantics of the GPEPA process algebra. The set of populations of a GPEPA model \( M \) is:

\[
S \left( G \left( P_1[n_1] \ldots P_m[n_m] \right) \right) = \left\{ (G, P_1), \ldots, (G, P_m) \right\}
\]

\[
S(M_1 \triangleright A M_2) = S(M_1) \cup S(M_2)
\]

In order to inductively define the set of PCTMC transitions for composed GPEPA models, we label the transitions with the GPEPA actions. For a simple labeled GPEPA group \( G \left\{ P_1[n_1] \ldots P_m[n_m] \right\} \) the resulting PCTMC has a (labelled) transition class for each PEPA transition \( P_i \xrightarrow{(\alpha, r)} P'_i \)

\[(G, P_i) \rightarrow (G, P') \quad \text{at rate } N_{G, P_i}, \text{label : } \alpha\]

Now let the model \( M \) be a cooperation of two GPEPA models \( M_1 \bowtie A M_2 \) and let be the transition classes of PCTMCs corresponding to the models \( M_i \) respectively (Hayden & Bradley).

The set of transition classes for the PCTMC of \( M \) consists of:

1. All transition classes in \( C_1 \) and \( C_2 \) not labelled by an action in the cooperation set \( A \).
2. For each combination of transition classes in \( C_1 \) and \( C_2 \) labelled by an action \( \alpha \) in the cooperation set.

\[
\sum_{i=1}^{k} G_i, P_i \xrightarrow{r} \sum_{i=1}^{l} G_i', P'_i \quad \text{at rate } R_i, \text{label } \alpha
\]

\[
\sum_{i=1}^{k} H_i, Q_i \xrightarrow{r} \sum_{i=1}^{l} H_i', Q'_i \quad \text{at rate } R_i, \text{label } \alpha
\]

The transition class defined as:

\[
\sum_{i=1}^{k} G_i, P_i + \sum_{i=1}^{l} H_i, Q_i \xrightarrow{r} \sum_{i=1}^{k} G_i', P'_i + \sum_{i=1}^{l} H_i', Q'_i \quad \text{at rate } R_i, \text{label } \alpha
\]
where the rate $R$ is:

$$R = \frac{R_1}{r_{\alpha}(M_1)} \cdot \frac{R_2}{r_{\alpha}(M_2)} \cdot \min(r_{\alpha}(M_1), r_{\alpha}(M_2))$$

where $r_{\alpha}(M_i)$ is the GPEPA apparent rate of the action $\alpha$ in the model $M_i$:

$$r_{\alpha}(M_i) = \sum_{\text{at } r, \text{label } a, x} r$$

### 3.3 MASSPA

Similarly to GPEPA we can also inductively define a PCTMC semantics of the MASSPA process algebra. The MASSPA grammar presented here was changed slightly from the one used in (Guenther & Bradley 2011) in order to make it easier for the reader to compare it with the GPEPA grammar discussed above.

$$P ::= (r).P | (m, g_m, r).P | ?(m, a_m).P | P + P | C_{\nu} \mid \emptyset$$

$$M ::= M \triangleright_{\nu} M \mid P \oplus \llbracket N \rrbracket$$

where $C = \{\text{Channel}(P \oplus l, P \oplus k, m, u), \ldots\}$

where $P$ is a generic agent component, whose definition, e.g. transition rates, may be location dependent (the location placeholder $x$ in the model shown in Figure 7). The $(r)$ transitions represent local transitions within an agent that
occur at rate \( r \), \( !(m,g_m,r) \) local transitions, which fire \( g_m \) messages of type \( m \)?(m,a_m) are message induced transitions that execute with probability \( a_m \) when the agent receives a message of type \( m \).

The set of all message types defined among the components is \( M \). A MASSPA model \( M \) consists of a finite composition of local agent component instances \( P@l[N] \) where local components are like component groups in GPEPA. A local agent component \( P@l \) inherits the definition from the generic agent component \( P \) but it can override message labels and transition rates to make them location dependent. The finite set of all locations is denoted \( L \) and for each \( P@l[N] \) we have \( l \in L \). Model instances of local components such as \( P@l[N] \) operate in parallel subject to channel constraints defined in the set \( C \).

We say that a local agent \( P@l \) can send a message of type \( m \in M \) to \( Q@k \) if \( \text{Channel}(P@l, Q@k, m, u) \in C \). When translating a MASSPA model into a PCTMC the local agent component \( P@l \) associates with the population \( P@l \).

\[
S(P@l[N]) = D(P@l)
\]

where \( D(P@l) \) is the set derivative populations of local agent component \( P@l \), i.e. the set of populations associated with local agent components that can be reached from \( P@l \) through transitions. This might well be a subset of all components that are theoretically reachable according to the generic agent component definition \( P \) since some local transition rates at location \( l \) could be zero, thereby disabling potential transitions. Like in GPEPA the set of populations of a composite model is simply:

\[
S(M_1 \cup M_2) = S(M_1) \cup S(M_2)
\]

Next we need to inductively define the set of PCTMC transitions for a MASSPA model. We start by deriving the local PCTMC transitions. In a MASSPA model the occurrence of any \( !(m,g_m,r).P'@l \) and \( (r).P'@l \) transitions in the definition of a local agent component \( P@l \) creates the following PCTMC transition for a model instance \( P@l[N] \).

\[
P@l \rightarrow P'@l \text{ at rate } N_{P@l} \cdot r
\]

While the behaviour of a local transition is similar to a silent transition in GPEPA, the translation of message emission transitions semantically

---

**Figure 7. Location placeholder \( x \)**

(a) Diagram of the system equation.  
(b) GPA definition using GPEPA syntax.
implies that broadcast messages are only sent while an agent is sojourning in a particular state. To express a broadcast message that is sent as part of a state transition, the translation to PCTMC level would become significantly harder, despite the fact that the resulting first-order moment mean-field ODEs would not change.

If $M$ is a composite of two MASSPA models $M_1 \triangleright \triangleleft M_2$ and $T_i$ the set of PCTMC transitions of the PCTMCs corresponding to the models $M_i$, the set of transitions for the PCTMC of $M$ is the union of $T_1 \cup T_2 \cup T_{12}$, where $T_{12}$ represents the set of message induced transitions that occur when other agent states receive messages on channels that they listen to.

For each Channel($P@l$, $Q@k$, $m$, $u$), $k, l \in \mathcal{L}, m \in \mathcal{M}$ for which exist valid transitions $(m_g, r).P@l$ in the definition of local agent component $P@l$ and $(m_a).Q@k$ in the definition of $Q@k$ there is a PCTMC transition:

$$P@l + Q@k \rightarrow P@l + Q'@k$$

at rate $N_{P@l} \cdot N_{Q@k} \cdot r \cdot g_m \cdot a_m \cdot u$

To illustrate the MASSPA grammar, Figure 7 shows a simple spatial content distribution model implemented in MASSPA. The diagram conceptually shows the available message channels described by the process algebra definition. Due to space restrictions we did not include all constant definitions. While choosing any non-zero rate for $r_{stale}$ will suffice for most locations, in location A we have to have $r_{stale}@A = 0$ to reflect that there is a reliable source of information in the network. Figure 8 shows the ODE mean-field analysis results (Section 4.1.2) for a particular configuration of the model.

### 4 SOLVING PCTMC MODELS IN GPA

Figure 9 outlines the input and steps GPA takes to evaluate PCTMC models. Having defined a model in a PCTMC translatable formalism, modellers can specify the measures that they are interested in, for instance central moments and distributions of a population or passage time distribution bounds. The choice of solution method then depends on the chosen measures and the rates in the model. GPA currently does not assist modellers by suggesting the most suitable evaluation method given
GPA

Figure 9. The process of solving a PCTMC model in GPA

Given the model, measures and the solution method, GPA first translates the model into an internal PCTMC representation and consequently generates a source code for the model to speed up its evaluation. The nature of the generated code depends on the solution method, for instance for the mean-field solution (Section 4.1.2) the source code represents the system of ODEs, which can be integrated using standard ODE solvers. One important aspect of the code generation is that the source code for the solution method only needs to be generated once, even when multiple experiments are conducted with different start conditions and parameters. Moreover, while GPA supports generated Java code natively, users can alternatively choose other output languages such as Matlab and C++.

4.1 Solution Methods

In a GPA analysable model, the solvers are defined below the model definition using the syntax in Table 1.

Each solver can take pre- and post-processor arguments. Pre-processor options have an impact on how source code is generated for the PCTMC model and the chosen metrics, whereas post-processor arguments are passed to the solution algorithm, for instance to define numerical accuracy and the stop time for the evaluation.

4.1.1 Simulation

GPA provides two variants of the stochastic simulation technique that is also known as Gillespie’s simulation technique (Gillespie1977). The syntax for the solution methods is shown in Table 2.

Simulation uses replications many traces from time to stopTime and records the

Table 1.

```
SOLVER[(<PREPROC_OPTION>,)*](<POSTPROC_OPTION,>)* { 
  (<MEASURE,>)* -> <OUTPUT_FILE>? ;
}
```

Table 2.

```
Simulation (stopTime = <FLOAT>, stepSize = <FLOAT>, replications = <INT>) {...}
AccurateSimulation (stopTime = <FLOAT>, stepSize = <FLOAT>, CI = <FLOAT>,
  maxRelCIWidth = <FLOAT>, batchSize = <INT>) {...}
```
GPA

ensemble statistics for every interval of stepSize width. AccurateSimulation simulation uses the same simulation algorithm, but instead of specifying the number of replications for the ensemble statistic computation, users can explicitly specify a confidence interval CI and a maximum relative tolerable confidence interval width maxRelCIWidth which is checked for all measures every batchSize many replications. The simulation only stops once the confidence interval is tight enough (Guenther et al. 2012) The Gillespie implementation is generic, but for efficiency reasons the reaction rate computation and the population state update functions are automatically generated from the model definition, compiled and passed to the Gillespie evaluator. To speed up simulation even further, we are also currently looking into implementing a -leaping simulator (Gillespie & Petzold 2003).

4.1.2 Mean-Field Analysis

Mean-field ODE analysis of population moments using GPA can be done by adding the following code to the solver section (see Table 3.)

In contrast to the simulation solvers, the ODE solution allows the user to provide optional preprocessor commands such as maxOrder, the order of the highest moment that is to be computed and the momentClosure method that is to be used. By default maxOrder is equal to the highest order among the moments defined in the measures section of the solver ... and the default moment closure is the normal moment closure.

The default value for maxOrder is intuitive, since we generally need to compute at least the first n moments if we are interested in an n-th moment population measure. Occasionally, however, the accuracy of the n-th moment estimation becomes better as we compute higher-order, say n+x-th order moments (Guenther et al. 2012). This is the case when we have a PCTMC model with an unclosed, infinite, linear systems of moment approximating mean-field ODEs. This happens when there exist evolution rates \( r(P) \) (Section 2.2) with non-linear polynomials in the population counts, such as \( r(P) = P_1 P_2 \). This quadratic reaction rate, is commonly referred to as a mass-action type reaction and can often be found in PCTMCs representing biological systems, but also in the MASSPA and MAM formalisms. When expanding Equation (3) for such PCTMCs, moment ODEs will depend on higher-order moment ODEs. In a simple example with a transition class:

\[
S_1 + S_2 \rightarrow S_3 + S_4 \quad \text{at rate} \quad r(P) = P_1 P_2
\]

the ODE describing the mean of \( P_1 \), \( \mathbb{E}[P_1(t)] \) depends on a second order moment \( \mathbb{E}[P_1(t) P_2(t)] \) the ODE for this moment depends on third order moments and so on. To solve such infinite systems of coupled ODEs numerically, we need to approximate higher-order moments of some order using moments with order no larger than the order of the highest moment we wish compute, i.e. we close the system of moment ODEs at a particular order. By applying such a moment closure, we are changing a linear but infinite system of moment ODEs into a finite non-linear system of ODEs. While allowing us to solve the closed system, there are two drawbacks to applying a

*Table 3.*

<table>
<thead>
<tr>
<th>ODEs [momentClosure = &lt;STRING&gt; , maxOrder= &lt;INT&gt;]?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(stopTime = &lt;FLOAT&gt;, stepSize = &lt;FLOAT&gt;, density = &lt;FLOAT&gt;) } ... }</td>
</tr>
</tbody>
</table>
closure. Firstly, to compute specific non-linear replacement terms for all higher-order moments that exceed maxOrder, we have to make a decision as to which family of distributions the population random variables belong to. Secondly, adding non-linear terms makes the numerical integration more expensive and sometimes less stable. Before moving on to our description of GPA’s ODE generation and moment closure application, we briefly review the mean-field and the normal moment closure (Whittle 1957). For a detailed overview of moment closures that have been implemented in GPA please refer to (Guenther et al. 2012).

Mean-field analysis (Opper & Saad 2001) methods investigate the evolution of the mean of population vectors. The mean-field closure approximates higher-order moments such as \( \mathbb{E}[P_i(t) P_j(t) \ldots P_k(t)] \) by the product of the individual expectations \( \mathbb{E}[P_i(t)] \mathbb{E}[P_j(t)] \ldots \mathbb{E}[P_k(t)] \). In other words, the mean-field approach ignores the covariance between any two populations. This produces good approximations for population means, especially when the populations are high. However in some model, for instance in the circadian clock model the mean-field closure does not perform well (Engblom 2006, Stefanek, Guenther & Bradley 2011). Note that the approximation of population moments using fluid analysis is generally referred to as mean-field analysis in the literature irrespective of both the moment closure that is applied and the order of the moments that are approximated.

The normal moment closure (Whittle, 1957) can be applied to any system of ODEs originating from a PCTMC for which we want to find 2nd or higher-order moments. It assumes that the populations at each point in time are approximately multivariate normal and therefore all third- and higher-order moments can be expressed in terms of means and covariances. This relationship is captured by the Isserlis’ theorem (Isserlis 1918):

For \( \vec{P} \) multivariate normal with mean \( \vec{\mu} \) and covariance matrix \( \sigma_{ij} \) we have:

\[
\mathbb{E}\left[(\vec{P} - \vec{\mu})^n\right] = \mathbb{E}\left[(P_1 - \mu_1)^{n_1} \ldots (P_n - \mu_n)^{n_n}\right] = 0,
\]

if \( o(\vec{m}) \) is odd

\[
\mathbb{E}\left[(\vec{P} - \vec{\mu})^n\right] = \sum \prod \mathbb{E}(P_i - \mu_i)(P_j - \mu_j),
\]

if \( o(\vec{m}) \) is even

(4)

where sums through all the distinct partitions of \( 1, \ldots, n \) into disjoint sets of pairs i,j. If some elements in \( \vec{m} \) are greater than one, then certain pairs i,j will appear multiple times in the resulting sum. To obtain the raw moment, we need to expand the central moment in Equation (4) first and subsequently rearrange the equation. For example, instead of including an ODE for the third order joint raw moment \( \mathbb{E}[P_1(t)P_2(t)Z] \) we can close the expansion at second order by using the approximation:

\[
\mathbb{E}[P_1(t)P_2(t)] \approx 2\mathbb{E}[P_2(t)]\mathbb{E}[P_1(t)] + \mathbb{E}[P_1(t)]\mathbb{E}[P_2(t)^2] - 2\mathbb{E}[P_1(t)]\mathbb{E}[P_2(t)]^2
\]

which yields

\[
\mathbb{E}\left[(P_i(t) - \mu_i(t))(P_j(t) - \mu_j(t))^2\right] = 0
\]

as required, since the multivariate normal distribution is not skewed.

For efficient ODE generation, GPA generates only those ODEs that are actually required for the evaluation of the measures. Having parsed the moments required for the measures specified for the solver, GPA first generates all right-hand side terms of these moments. Before being added to the right-hand side, each term is first closed using the chosen moment closure method and any previously unknown population moments are added to the list of ODEs that still need to be generated. When there are no more new population moments, the generation algorithm terminates and returns a sys-
tem of ODEs, which can then be translated to Java, C++ or Matlab source code. Subsequently GPA fetches the post-processor parameters \( \text{stopTime}, \text{stepSize} \) and \( \text{density} \) and passes them to the integration algorithm along with the compiled version of the ODE source code. While, \( \text{stopTime} \) and \( \text{stepSize} \) have the same meaning as for the simulation solver explained in Section 4.1.1, the \( \text{density} \) argument currently defines the integration interval size of explicit Runge-Kutta method used by GPA. Naturally, it would be possible to provide other integration methods such as Euler’s method or potentially a parallel algorithm. To do this, all developers would need to do is implement the algorithm using the provided post-processor interface and further add another option such as \( \text{algoName} = \text{<STRING>} \) to enable users to choose between all available integration methods.

### 4.1.3 Computing Accumulated Rewards

In case of the previously mentioned Client/Server model, a useful metric of interest would be the total power consumption of all the servers throughout the analysed time interval. For example, we can assume that each server consumes \( c_1 \) and \( c_2 \) units of energy per unit of time in the states Server and Server_get respectively and none in the Server_broken state. In that case, the total energy can be written as an integral over the population processes:

\[
E(t) = c_1 \int_0^t S(u) \, du + c_2 \int_0^t S_2(u) \, du
\]

In general, other useful quantities can be expressed as linear combinations of the integrals of populations over time. It can be shown (Stefáněk et al. 2012) that the system of ODEs from the fluid analysis can be extended with additional ODEs describing the evolution of means and higher moments of these integrated processes. GPA implements this extension for both ODE and simulation solvers. The syntax for \( \int_0^t P(u) \, du \) is \( \text{acc}(P) \), so for example to plot the mean energy consumption above, the following expression can be used (see Table 4).

### 4.2 Experiments

The flexible architecture of GPA allows analysis methods to be combined, compared or repeated in a number of secondary experiment solvers.

#### 4.2.1 Optimisation

Fluid analysis is often much more computationally efficient than an equivalent stochastic simulation. This makes it possible to repeat the analysis for a large number of parameter configurations. GPA provides a command that can iterate over a range of values of the parameters and evaluate complex reward and constraint expressions. For example, in the client / server model, we can define a reward as the total income from running the system. We can define this as a combination of the number of completed client requests minus the energy above and the initial cost of servers where \#data is the syntax for the number of times the data action has taken place and income is the units of income for each request, \( e_{\text{cost}} \) the cost of consuming a unit of energy per unit time and \( S_{\text{cost}} \) the cost of each server (see Table 5.) Using techniques from (Hayden et al. 2012) we can also define an expression that captures the probability that a client finishes each request within a given time. A

\[
\begin{align*}
\text{Table 4.} \\
E[c_1 \ast \text{acc}(\text{Servers:Server}) + c_2 \ast \text{acc}(\text{Servers:Server_get})]
\end{align*}
\]
service level agreement could be that this probability has to be higher than an agreed constant (say at a given point in time, e.g. after units of time. A possible optimisation problem for a provider of such a system would be to explore the number of servers that are needed to satisfy the SLA. Additionally, we could assume that the provider can obtain cheaper servers at the expense of reliability, making the cost $S_{\text{cost}}$ inversely proportional to the break rate of servers. Figure 10 shows how this problem could be specified in GPA.

**4.2.2 Unified Stochastic Probes for GPEPA**

In case of model specification in GPEPA, GPA implements the Unified Stochastic Probes (USP) framework (Hayden et al. 2013). This allows modellers to observe additional passage times without explicitly modifying the GPEPA system model. The USP language can express a variety of passage time measures. For example, in the Client/Server model, a measure of interest could be the time it takes an individual client to execute its first think action. This can be achieved by defining a probe attached to a single client component. This probe starts its clock at the initial time and finishes when the observed client fires a think action. Probes are defined in a regular expression like language and multiple probes can be composed together. To define the placement of the probe (in this example attaching the probe to a single client), USP accepts a simple GPEPA model transformation language. Figure 11 shows the GPA definition of the probe and the respective model transformation. This probe definition shows an example of an individual passage time measurement. Additionally, the framework defines so-called global passage times. These can use clocks triggered by a mixture of actions and conditions on the global
state space. For example, a measure of interest could be the time it takes for half of the total $N_C$ clients to execute their first think action.

Internally, GPA follows the algorithms defined by the USP framework. Using the model transformation and definitions of probes, the original GPEPA model is translated into an intermediate GPEPA model that also allows passive cooperation. This is then translated into a standard GPEPA model so that the execution chain of GPA can be applied. As shown in Figure 1, the generated core for the primary solvers is re-used and the respective analysis executed to calculate certain expressions required by the USP algorithms. Finally, the results are transformed to provide approximations of various passage time measures. Figure 12 plots an example of an individual passage time and a global passage time computed by GPA.

### 4.2.3 Distribution Calculation

To give further insight into the behaviour of the underlying PCTMC of a model, GPA can calculate the probability distribution of population and reward based expressions. The example in Table 6 shows the syntax to calculate the distribution of the population of servers and the server utilisation at each time within the interval of the analysis.
GPA generates the code for primary solvers and then adds additional routines to keep track of the sample values of the given expressions. In the end, it displays the distributions, Figure 13.

5 FUTURE GPA EXTENSIONS

While the current version of GPA supports a large range of PCTMC specific analysis techniques and modeling features, our ultimate goal is to extend GPA’s capabilities to other types of population models. In this section we will discuss some planned future extensions to GPA, which will capture a number of population model classes that were mentioned in Section 2.

5.1 Population Models with Continuous States

One such extension is to consider models which are governed by some continuous quantities evolving over time in addition to the discrete populations. These can represent energy, temperature, continuous location in a space etc. One line of work has further extended the accumulated reward described in Section 4.1.3. Instead of considering the rewards as additional measures obtained from the analysis, they can be treated as part of the state space of the system. The population vector of the PCTMC can be augmented with a number of global continuous quantities that evolve over time as defined by a set of ODEs (Stefanek et al. 2012). These ODEs can depend on the population vectors and the transition rates of the PCTMC part of the model can in turn depend on the continuous quantities. It can be shown that the ODE techniques can be extended to provide means and higher moments of both populations and continuous quantities in such models. The current version of GPA supports this to some extend by allowing rates to depend on expressions that involve accumulated populations. In future, we plan to extend this to general accumulations defined by a set of additional ODEs.
Another line of work adds continuous quantities at individual rather than at global level. For example, Hayden et al. (Hayden 2012c) look at a wireless sensor network model where each of the large number of nodes in addition to the discrete state space contains a continuous value representing the current battery level. In such cases, the fluid analysis comes down to a system of Partial Differential Equations that capture the mean populations together with the distribution of the continuous quantities associated with individuals.

5.2 Time-Inhomogeneous Population Models

Another extension are time inhomogeneous population models. Generally these include any population model class, which feature rates or population changes that depend on time. In the near future we are intending to release an extension for GPA, which will allow modellers to define and evaluate a range of time-inhomogeneous PCTMCs (IPCTMCs) that were shown to be useful in (Guenther & Bradley 2013, Stefanek et al. 2013). A IPCTMC is a PCTMC that also features deterministic rate and population changes, which occur at deterministic times. More formally for rates this means that any reaction rate $r_i(\hat{P}(t))$ (Section 2.2) is now time dependent, i.e. $r_i(\hat{P}(t), t)$ becomes:

$$
\begin{cases}
    r_i(\hat{P}(t), t_1) & \text{if } P(t) \geq s_i \forall i = 1, \ldots, n \land t < t_1 \\
    r_i(\hat{P}(t), t_2) & \text{if } P(t) \geq s_i \forall i = 1, \ldots, n \land t_1 \leq t < t_2 \\
    \vdots & \\
    0 & \text{otherwise}
\end{cases}
$$

where $t_1, t_2, \ldots$ are deterministic times at which reaction rate changes occur. Moreover, we allow deterministically timed events that result in an affine transformation of the population vector $\hat{P}(t)$. In the following let us assume that if a deterministic population change occurs at time $t_j$ then no population changes occur due to random PCTMC events between $t_j - \delta t$ and $t_j$. Should no such interval exist then we informally assume that the deterministic event is triggered immediately after the random event. Let $D$ denote the set of all deterministic events, s.t. $(t_d, M) \in D$, where:

$$
M_{n,n+1} = \begin{pmatrix}
    \lambda_1 & 0 & \ldots & 0 & c_1 \\
    0 & \lambda_2 & \ldots & 0 & c_2 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & 0 & \ldots & \lambda_n & c_n
\end{pmatrix}
$$

and the updated population vector is computed using the following affine transformation:

$$
\hat{P}(t_d) = M \times \begin{pmatrix}
    1 \\
    0 \\
    \vdots \\
    0 \\
    0
\end{pmatrix} \times \hat{P}(t_d - \delta t) + \begin{pmatrix}
    0 \\
    0 \\
    \vdots \\
    0 \\
    0
\end{pmatrix}
$$

that way we can describe a reset of population $i$ to $c_i$, and a population jump of population $i$ by $c_i$ as:

$$
\text{Reset}_{n,n+1} = \begin{pmatrix}
    0 & 0 & \ldots & 0 & c_i \\
    0 & 1 & \ldots & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    1 & 0 & \ldots & 0 & c_i \\
    0 & 1 & \ldots & 0 & 0
\end{pmatrix}
$$

$$
\text{Jump}_{n,n+1} = \begin{pmatrix}
    0 & 0 & \ldots & 0 & 0 \\
    0 & 0 & \ldots & 0 & 0 \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    0 & 0 & \ldots & 0 & 0 \\
    0 & 0 & \ldots & 0 & 0
\end{pmatrix}
$$

To use IPCTMCs in (Guenther & Bradley 2013, Stefanek et al. 2013) we had to implement new GPA post-processors for both the simulation and the ODE solver. Although we are presently improving these methods, here is a brief description of the solution algorithm that we intend to release in a GPA version later this year. In IPCTMC simulation runs, deterministic events can be immediately applied to the current population.
vector at the time they occur. Thus a IPCTMC simulation can be build on top of the existing simulation code, the difference being that we need to apply the affine transformation of the population vector and rate changes every time a deterministic event occurs. Similarly, when using mean-field ODEs to estimate the moments of the IPCTMC process, we need to integrate the ODEs stepwise between subsequent events. This means that we can also reuse existing ODE integration algorithms and simply have to take care of applying the deterministic events. However, unlike individual simulation runs, mean-field ODEs keep track of population moments, which have to be recomputed whenever a deterministic event occurs at time $t_d$. For instance assuming that we a have two populations and that the following deterministic population change occurs at $t_d; x(t_d) = \lambda_x x(t_d - \delta t) + c_i$ and $Y(t_d - \delta t)$ then

$$
\mathbb{E}\left[ X(t_d) Y(t_d) \right] = \mathbb{E}\left[ \lambda_x X(t_d - \delta t) + c_i \right] Y(t_d - \delta t)
$$

and similarly for

$$
\mathbb{E}\left[ X(t_d)^2 \right]
$$

or any other uncentred joint moment. Since the transformation of the population vector is affine, we do not have to use moment closures (Guenther et al. 2012) in order to compute the new values of the moments at time $t_d$. However, an important issue that is subject to future research are solution methods that can handle boundary conditions for populations. For instance, if we had to ensure that a population must always stay positive, we would need to change our population vector update method for the simulation. Moreover, for the ODE solution of IPCTMCs such boundary conditions are even harder to achieve, as we would have to apply moment closures to estimate the probability that a deterministic IPCTMC event leaves the population vector within its legal boundaries before updating the population vector.

### 5.3 Population Models with Non-Markovian Delays

So far the population models supported by GPA all have an underlying CTMC representation. However, models often require deterministic or other non-exponentially distributed delays that cannot easily be approximated using phase-type distributions. Under these circumstances we require a population GSMP model (Section 2.1). It was recently shown that first order mean-field analysis can be used to solve a subclass of PGSMP models that supports deterministic delays (Bortolussi & Hillston 2012a, Hayden 2012b). Moreover, both papers hint at how the solution technique could be applied to PGSMPs with generally-timed delays. While incorporating PGSMPs into GPA is a straightforward task, it would ideally lead to a novel, more general abstraction layer between population formalisms, low-level state-space representations and available solution methods, thus helping GPA to embrace the MFMS paradigm further.

### 5.4 Model Checking

In Section 4.2.2 we illustrated how Unified Stochastic Probes (Hayden et al. 2013, Kohut et al. 2012) were integrated into GPA. While mean-field ODEs for population dependent global passage times can only be used to derive bounds of the underlying distribution, it was shown that passages time distributions of individuals can be computed exactly if the individual is assumed to be interacting with a mean field regime (Hayden et al. 2012). In the literature this is also occasionally referred to as fast simulation (Darling & Norris 2007). The efficient mean-field computation of
both global passage time bounds as well as individual passage time distributions, could help to address model checking problems for population models that were previously confined to the realm of simulation techniques, due to their vast state spaces. As an example Bortolussi et al. (Bortolussi & Hillston 2012b) show how mean-field passage time computations can be used to cover a subset of the Constraint Stochastic Logic (CSL) formalism. Fluid analysable subsets of CSL or similar model checking formalisms would make a useful extension to GPA. Moreover, if general model checking formalisms were to be added to GPA, it would be possible to write an algorithm that determines whether individual queries can be solved using mean-field analysis or if simulation techniques or state space exploration techniques are needed. This would then enable us to compare the performance of different solution techniques for various types of models and model checking queries and possibly help us to assist modellers in their choice of solver.

6 CONCLUSION

Our examples of mean-field analysable model classes in Section 2.1 show that recent research in the field has vastly increased the expressiveness of tractable formalisms. However, the popularity of tools such as Möbius and PRISM illustrates that only the availability of user-friendly software tools, truly brings theoretical advances to the modelling community. In view of the ever increasing size and complexity of systems, it is thus important to offer modellers a powerful population modelling and analysis framework. Even though GPA was initially designed as an evaluation tool for GPEPA, the intention was to provide modellers with the means that would encourage them to move away from simulation based analysis of large population models towards more efficient mean-field analysable formalisms. As the family of mean-field analysable population modeling continues to extend, we aim to implement new formalisms and solution techniques in GPA. However, the transformation from a single formalism tool into a general multiformalism multisolution population model analysis framework is ongoing work, as the examples in Section 2.1 show. Ultimately, we hope that future development will help to make GPA accessible to an even wider group of modellers and become the framework of choice for implementing future population modelling formalisms.

REFERENCES


