Swarm Intell (2013) 7:201–228
DOI 10.1007/s11721-013-0079-6

On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics

Mieke Massink · Manuele Brambilla · Diego Latella · Marco Dorigo · Mauro Birattari

Received: 31 October 2012 / Accepted: 1 March 2013 / Published online: 11 April 2013

Abstract In this paper we analyse a swarm robotics system using Bio-PEPA. Bio-PEPA is a process algebra language originally developed to analyse biochemical systems. A swarm robotics system can be analysed at two levels: the macroscopic level, to study the collective behaviour of the system, and the microscopic level, to study the robot-to-robot and robot-to-environment interactions. In general, multiple models are necessary to analyse a system at different levels. However, developing multiple models increases the effort needed to analyse a system and raises issues about the consistency of the results. Bio-PEPA, instead, allows the researcher to perform stochastic simulation, fluid flow (ODE) analysis and statistical model checking using a single description, reducing the effort necessary to perform the analysis and ensuring consistency between the results. Bio-PEPA is well suited for swarm robotics systems: by using Bio-PEPA it is possible to model distributed systems and their space-time characteristics in a natural way. We validate our approach by modelling a collective decision-making behaviour.

Keywords Swarm robotics · Modelling · Bio-PEPA · Fluid flow analysis · Statistical model checking

M. Massink (✉) · D. Latella
Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo' (ISTI), CNR, Pisa, Italy
e-mail: massink@isti.cnr.it

D. Latella
e-mail: latella@isti.cnr.it

M. Brambilla · M. Dorigo · M. Birattari
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

M. Brambilla
e-mail: mbrambil@ulb.ac.be

M. Dorigo
e-mail: mdorigo@ulb.ac.be

M. Birattari
e-mail: mbiro@ulb.ac.be

1 Introduction

The analysis of large and complex swarm robotics systems (Sahin 2005; Brambilla et al. 2013) directly with robots is costly and time consuming. For this reason, one of the most common ways to analyse a swarm robotics system is through *physics-based simulations*. Using physics-based simulations, it is possible to realise a very detailed representation of the system to analyse. However, such a detailed representation could encumber the analysis process. In fact, through simulation it is difficult to abstract from the details of a system in order to identify its key components and parameters. Moreover, physics-based simulations are not well suited for the formal verification of properties.

To overcome these limitations, other approaches are usually employed (Brambilla et al. 2013). *Fluid flow analysis* (Lerman et al. 2005), based on macroscopic models and ordinary differential equations (ODEs) is commonly used for describing the collective behaviour of a system. *Stochastic simulations* (Dixon et al. 2011), based on microscopic models, focus instead on the behaviour of individual robots. Finally, *model checking* (Konur et al. 2012), based on Markov chains and mathematical logic, can be used to verify formal properties of a swarm robotics system.

These approaches allow a developer to obtain different "views" of the system behaviour. However, for each of these views, a different model is necessary: macroscopic models, microscopic models and Markov chains are just three examples of possible models. Producing such models greatly increases the effort necessary for the analysis process. Moreover, when dealing with different models, the issue of mutual consistency must be addressed.

In this paper, we present a novel approach to model swarm robotics systems based on Bio-PEPA (Ciocchetta and Hillston 2009), which allows one to obtain different consistent views of a system from the same formal specification.

Bio-PEPA is a process algebra language originally developed for modelling biochemical systems. Bio-PEPA has been adopted to analyse a number of biological systems (Ciocchetta and Hillston 2012) and disease spread (Benkirane et al. 2012), but it has also been used to analyse emergency egress (Massink et al. 2012c) and crowd dynamics (Massink et al. 2011b) which are systems characterised by a high number of individuals and lack of a centralised controller, aspects common also to swarm robotics systems. Bio-PEPA is well suited to analyse and develop swarm robotics systems; with Bio-PEPA it is possible to develop a specification at the microscopic level while providing also primitives for spatial description (e.g. *locations*) and for composition of individual robot behaviour specifications (e.g. *co-operation* operator). Moreover, Bio-PEPA allows one to easily define *species*, which can be used to characterise groups of robots with specific attributes and actions; for instance, *species* can be used to differentiate between groups of robots performing different tasks at the same location.

Other techniques developed for biochemical systems have been used in a number of papers (Napp et al. 2011; Mather and Hsieh 2012; Evans et al. 2010) to model swarm robotics systems due to the similarities between these two kinds of system. Examples of such similarities are: a large number of equal entities, lack of centralised control, group formations, different behaviours for different entities. Differently from other biochemical modelling approaches, Bio-PEPA allows one to perform different kinds of analysis using a single description of a system. As said, this aspect is also crucial for swarm robotics, where different analyses usually require different descriptions of the same system.

In this paper, we use Bio-PEPA to develop a formal specification and to analyse a collective decision-making behaviour which has been extensively studied in several papers (Montes de Oca et al. 2011; Scheidler 2011; Valentini et al. 2013). The case study consists of a swarm of robots that have to collectively identify the shortest path between two

possible choices. We validate our results against those presented in Montes de Oca et al. (2011).

This paper builds on the preliminary results of Massink et al. (2012b). Compared to our previous work, in this paper we present in more detail Bio-PEPA and its application to the analysis of swarm robotics systems. To further validate our approach, we extended the analysis of the case study, producing numerous new results. In particular, we extended the statistical model checking analysis (Sect. 5.2) and the fluid flow analysis (Sect. 5.3).

The outline of the paper is as follows. In Sect. 2, we present related work. In Sect. 3, we give a brief presentation of Bio-PEPA. In Sect. 4, we present the case study and its Bio-PEPA specification. In Sect. 5, we present and validate our results. Some conclusions are drawn in Sect. 6.

2 Related work

Swarm robotics systems can be observed at two different levels: the microscopic level and the macroscopic level. At the microscopic level we observe the individual robots and their interactions. At the macroscopic level we observe the swarm as a whole, that is, what results from the interactions of the individual robots. This duality is reflected also in how swarm robotics systems are modelled: models of swarm robotics systems can be developed both using a microscopic approach and a macroscopic approach. These different approaches result in different ways to analyse a swarm robotics system.

In this section, we present work on microscopic modelling, macroscopic modelling and model checking.

Microscopic modelling Microscopic models allow a researcher to study a swarm robotics system observing in detail the robot-to-robot and robot-to-environment interactions. Such interactions are the fundamental components of any swarm robotics system. The macroscopic behaviour of a swarm is in fact the result of these interactions.

Microscopic models give us a very detailed view of a system. However, this high level of detail is also the cause of the main limit of microscopic models: limited scalability. In order to describe the behaviour of a swarm it is necessary to replicate the description of the single individual for the number of individuals in the swarm. This results in models with a large number of components, which are computationally heavy to treat. Microscopic models are usually analysed via stochastic simulation (Brambilla et al. 2013; Lerman et al. 2005).

Macroscopic modelling Macroscopic models, instead, consider the time evolution of the swarm as a whole, ignoring the details of the individual robots composing it. Using macroscopic models it is possible, for example, to observe how the swarm evolves and reaches a common decision, but not the details of the contribution of each robot to such a decision.

The most common way to create a macroscopic model for a swarm robotics systems is by specifying a number of ordinary differential equations (ODEs) which model the distribution over time of the components of the system. ODEs are often called rate equations in swarm robotics. The analysis performed with ODEs is sometimes called *fluid flow analysis*, as the changes in the distribution of the robots can be modelled as changes in the distribution of liquids flowing from one location to another (Zarzhitsky et al. 2005).

This high level view allows one to analyse swarm robotics systems composed of thousands of robots in a computationally feasible way, making macroscopic models the most common modelling approach in swarm robotics (Brambilla et al. 2013). The price to pay for

this scalability is that it becomes difficult to analyse in detail the interactions of the robots. In general, macroscopic models focus on the average behaviour of the system, abstracting from local stochastic fluctuations. A review of macroscopic modelling in swarm robotics can be found in Lerman et al. (2005). A comparison between the microscopic and the macroscopic models of a swarm robotics system is presented in Martinoli et al. (2004).

Model checking Another way for specifying and analysing properties of interest of a swarm robotics system is through mathematical logic and model checking—see, for example, Baier et al. (1999). This approach has been applied with success to several different fields (Burch et al. 1990; Havelund et al. 2001) but it has not been explored extensively in swarm robotics. Some examples of model checking in swarm robotics can be found in the works by Konur et al. (2012) and Brambilla et al. (2012).

Model checking allows the user to automatically verify whether a given property is satisfied by a given model. The model, which is usually expressed through a Markov chain, can be either microscopic or macroscopic. The property is expressed using a mathematical logic.

Discussion In general, to perform different analyses of the same system, such as stochastic simulation, fluid flow (ODE) approximation and model checking, different models are necessary. Microscopic models are best suited for stochastic simulations; ODE macroscopic models are best suited for fluid flow approximation; Markov chain models may be used for stochastic model checking. The proliferation of different formalisms necessary to perform different analyses increases the effort necessary to study a system and may create problems of consistency between different models and related analysis results.

A way to overcome this problem is to use a high level meta-modelling language. In this paper, we use Bio-PEPA. Bio-PEPA allows us to perform different kinds of system analysis starting from a single specification.

3 Bio-PEPA

Bio-PEPA is a language that was originally developed by Ciocchetta and Hillston (2009) for the stochastic modelling and analysis of biochemical systems. Bio-PEPA is based on the Performance Evaluation Process Algebra (PEPA) (Hillston 1996).

For the purposes of the present paper, process algebras are formal specification languages for modelling concurrent/distributed systems behaviour (Bergstra et al. 2001). They are characterised by a formal definition of their syntax, which defines precisely the format of the terms of their language, that is, the *specifications*, and a formal definition of their operational semantics, which assign meaning to such specifications, associating them to mathematical objects. Furthermore, process algebras are equipped with mathematical theories for reasoning about system behaviour, as, for example, behaviour equivalences, which precisely characterise when two or more different specifications actually specify "the same" (strictly speaking, we should say "equivalent") behaviour.[1] The availability of support theories based

[1] Often, such equivalences are congruences, which is a crucial feature for compositional modelling. Intuitively, one would like to build specifications of large systems in a modular way, by composing specifications of sub-systems in order to get those of larger systems. If the specification of one such sub-system, say S_1, is proved to be *congruent* to another specification, say S_2, then we are allowed to replace S_1 with S_2 in any specification S which has S_1 as sub-component, *with the guarantee that* the overall behaviour of S *will not be affected*. Note that this is not guaranteed if one uses equivalences which are *not* congruences.

upon solid mathematical foundations, together with analysis techniques based on such theories and often supported by efficient software tools, makes process algebras particularly useful for the specification and analysis of behavioural aspects of concurrent/distributed systems.

Stochastic variants of process algebras, of which Bio-PEPA is an instance, have shown to be particularly suitable to model also performance aspects of systems (see, e.g. Hermanns et al. 2002; Aldini et al. 2010). Process algebras thus provide a formal and unambiguous framework for modelling and reasoning about system behaviour in a compositional way.

In Bio-PEPA, processes represent groups of similar entities. The interactions between groups of entities affect their population sizes. In the context of swarm robotics we will use this abstraction of "processes as groups of entities" to model groups of robots performing different behaviours. We will use the concept of interaction to model, for example, the movement of robots between different locations and for the formation of teams. The idea is that movement of robots between locations can be modelled as a simultaneous decrease of the size of a population situated in the location that is left and increase of the size of a population in the location of arrival. Regarding team formation, another feature, specific of Bio-PEPA, is particularly useful: the possibility to express the multiplicity of entities involved in single interactions, known as "stoichiometry" in the context of biochemistry. With this feature one can specify, for example, that three single robots can form a single team that subsequently is treated as a single entity in a new kind of "species".

Interactions have durations which are modelled as continuous random variables with negative exponential distributions. The use of negative exponential distributions is justified by the fact that Bio-PEPA semantics is based on Continuous Time Markov Chains (CTMCs) and fluid flow approximations thereof, characterised by systems of ordinary differential equations (ODEs). CTMCs are among the most successful frameworks in the field of quantitative system modelling and analysis, such as, for example, performance or dependability analysis. They are based on exponential distributions, which have useful and interesting mathematical properties. In practice, restriction to exponential distributions does not represent a real limitation since it has been shown that any random distribution can be approximated by suitable combinations of negative exponential ones, of course at the cost of larger models, which can be ameliorated by suitable exploitation of process algebra equivalences (Tschaikowski and Tribastone 2012). Finally, fluid limits and semantics are the key for actual scalability. Roughly speaking, for a CTMC modelling the behaviour of a population of agents, one can "scale it up" by means of making the population size increase. Under certain (scaling) conditions, the (limit) behaviour for an infinite population can be obtained via the solution of a suitable set of ODEs. Furthermore, good approximations of such "fluid" limit can be obtained for large, but finite, population sizes. In the case of Bio-PEPA, such a set of ODEs can be automatically generated for any Bio-PEPA system specification and constitutes what is usually called the ODE or "fluid flow" semantics of the language.

A further feature of Bio-PEPA is that the rates at which interactions occur can be defined as general functions of the population sizes of the groups involved in the interaction. This provides great flexibility in the definition of such rates.[2]

[2]This flexibility comes at a cost. Not all analysis methods, such as Gillespie's stochastic simulation algorithms, can yet deal with the full generality of rate functions or with interactions that depend on more than two species as input to the interaction. Similar caution is needed with the interpretation of numerical solutions of the sets of ordinary differential equations derived from a Bio-PEPA specification. We will address these issues in more detail in the analysis of the robot swarm decision-making strategy in Sect. 5.

We now briefly recall the aspects of the Bio-PEPA language that are directly relevant for the swarm robotics study that will be presented in the next section. The interested reader can find further details of Bio-PEPA in Ciocchetta and Hillston (2009).

Bio-PEPA specifications consist of two main kinds of component. The first kind is called the "*species*" *component*. Each species defines the behaviour of *all* the individuals belonging to it. Species components are composed together in order to build a model using the *parallel composition* operator with *synchronisation* on shared actions. The name "species" derives from the biochemical origins of Bio-PEPA; in the context of swarm robotics, the name "population" is usually preferred; in the following, we will consider "species" and "population" as synonyms.

The second kind of component is called *model component*. Model components define how species components are composed together in order to build a model using the *parallel composition* operator with *synchronisation* on shared actions.

The syntax of Bio-PEPA components is thus defined as follows, where S stands for a *species component* and P for a *model component*:

$$S ::= (\alpha, \kappa)\ \mathrm{op}\ S \mid S + S \mid C \quad \text{with } \mathrm{op} = \downarrow \mid \uparrow \mid \oplus \mid \ominus \mid \odot \text{ and } P ::= P \underset{\mathcal{L}}{\bowtie} P \mid S(x)$$

The *prefix combinator* "op" in the prefix term $(\alpha, \kappa)\ \mathrm{op}\ S$ represents the impact that action α has on species S. Specifically, $\downarrow$ indicates that the number of entities of species S decreases when α occurs, and $\uparrow$ indicates that this number increases. The amount of the change is defined by the (stoichiometry) coefficient κ. This coefficient captures the multiples of an entity involved in an interaction. The default value of κ is 1, in which case we might simply write α instead of (α, κ). Action durations are assumed to be random variables with negative exponential distributions, characterised by their *rates*. The rate of action α is defined by a so called functional rate or kinetic rate. Action rates are defined in the context section of a Bio-PEPA specification.

The symbol $\oplus$ denotes an *activator*, $\ominus$ an *inhibitor* and $\odot$ a generic *modifier*, all of which play a role in an action without being produced or consumed and have a defined meaning in the biochemical context. We will not use these operators in the present paper. The operator "+" expresses the choice between possible actions, and the constant C is defined by the equation $C \overset{def}{=} S$. The process $P \underset{\mathcal{L}}{\bowtie} Q$ denotes cooperation between components P and Q, the set $\mathcal{L}$ determines those actions on which the components P and Q are forced to synchronise. The shorthand $P \underset{*}{\bowtie} Q$ denotes cooperation on all actions that P and Q have in common. In $S(x)$, the parameter $x \in \mathbb{R}$ represents the initial amount of the species.

As a simple example, consider two groups of robots, R and B, identified by their respective colour, red and blue. Assume that we want to model the formation of a group T of teams each composed of two red robots and a blue one. Assume furthermore that the rate at which team formation occurs is regulated by a rate function that is proportional to the population size of red and blue robots. In Bio-PEPA this behaviour can be modelled as follows. The effect of the formation of a team, represented by action *mk_team*, on the three "species" can be defined as:

$$R \overset{def}{=} (mk\_team, 2)\downarrow R \qquad B \overset{def}{=} (mk\_team, 1)\downarrow B \qquad T \overset{def}{=} (mk\_team, 1)\uparrow T$$

The system can then be described by the following model component:

$$\left(R(r_0) \underset{\{mk\_team\}}{\bowtie} B(b_0)\right) \underset{\{mk\_team\}}{\bowtie} T(0)$$

where r_0 and b_0 denote the initial population sizes of the groups of red and blue robots, respectively, and 0 denotes that initially T is empty. What remains to define is the rate at

which the teams are formed, that is, the rate of action *mk_team*. This rate could be defined as the rate function: $f_{mk\_team} = r \times R \times B$.

So, operationally, what happens is that every time a *mk_team* action occurs, the three species that share this action are synchronised and two red robots and one blue are taken away from the species R and B, respectively, while at the same time one new team is generated and added to the species T increasing its population size by 1. How often the *mk_team* action occurs is given by its rate function, which in turn depends on the actual population sizes of the species R and B and a constant rate parameter r. Note also that, whenever one of these population sizes becomes zero, the rate function goes to zero too and the interaction can no longer take place.

In Bio-PEPA specifications, one can also use *location*s, which are meant to be a symbolic representation of physical space. Locations are specified by extending prefix terms with the notation @*location*. So, for instance, in order to specify that an action, say α, has an effect on population S that is *located* in location l, and in particular involves κ individuals of S, we write (α, κ) op $S@l$. Additionally, locations are used in model components in order to specify the initial size of the various populations in each location. Of course, each location used in a Bio-PEPA specification must be declared; thus, a Bio-PEPA *system* specification with *locations* consists of a set of species components, a model component, and a context containing definitions of locations, functional/kinetic rates, parameters, and so on.

Bio-PEPA is given a formal operational semantics based on Continuous Time Markov Chains (CTMCs) and on Ordinary Differential Equations (ODE) (Ciocchetta and Hillston 2008, 2009).

Bio-PEPA is supported by a suite of software tools which automatically process Bio-PEPA models and generate internal representations suitable for different types of analysis as described in detail in Ciocchetta and Hillston (2009) and Ciocchetta et al. (2009). These tools include mappings from Bio-PEPA to differential equations (ODE) supporting a fluid flow approximation (Hillston 2005), stochastic simulation models (Gillespie 1977), CTMCs with levels (Ciocchetta and Hillston 2008) and PRISM models (Kwiatkowska et al. 2011) amenable to statistical model checking. Consistency of the analyses is supported by a rich theory including process algebra, and the relationships between CTMCs and ODE.

4 Collective decision-making: a Bio-PEPA specification

To demonstrate the characteristics of Bio-PEPA, in this section we analyse a collective robot swarm decision-making system presented originally by Montes de Oca et al. (2011).[3] The goal of the swarm of robots is to perform foraging: the robots carry objects from a *start* area to a *goal* area. Unlike many foraging scenarios (Brambilla et al. 2013), the objects to be carried are too heavy for a single robot, thus cooperation is necessary: the robots form teams of three to be able to carry an object.

The start and the goal areas are connected by two paths: a short path and a long path. This scenario is thus very similar to the ants double bridge experiment (Goss et al. 1989). Similar to what ants do in the double bridge experiments, the robots have to collectively identify and choose the shortest path. Differently from what ants do, the robots do not use pheromones but a voting process based on the majority rule.

Each robot has a preferred path. When a group is formed in the start area (Fig. 1(a)), a vote takes place and the group chooses the path that is preferred by the majority of the

[3] Since an implementation of this system using real robots is not available, the physics-based simulation will be considered our *ground truth*, that is, not another analysis phase, but the subject of our analysis effort.

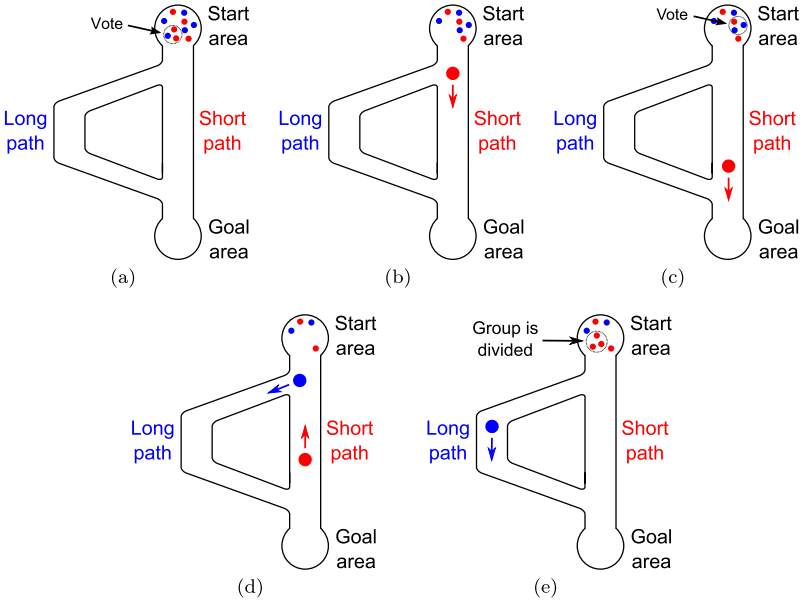

Fig. 1 The foraging scenario analysed in this paper. The robots start in the start area. Groups are formed and the path is chosen using the majority rule. In this figure two examples of the voting process are shown: (**a**) a group is formed; (**b**) the group has chosen the short path; (**c**) while the first group is active, another group is formed; (**d**) the second group has chosen the long path, at the same time the first group is coming back; (**e**) the first group is back in the start area and is disbanded

robots composing it (Fig. 1(b)). The chosen path also becomes the new preferred path for all the robots composing the group (Fig. 1(e)). For example, if two robots prefer the short path and one robot prefers the long path, the short path is chosen for the next run and the robot that preferred the long path changes its preference to the short path. Note that the voting process takes place only in the start area and no other event can change the preference of the robots. This means that robots come back to the start area following the same path taken for the outgoing trip. Figure 1 shows a schema of the scenario.

Since the robots taking the short path spend less time out of the start area than the robots taking the long path, their participation in the vote is, on average, more frequent. This results in the formation of more groups preferring the short path. If, initially, half of the robots have a preference for the short path and half for the long path, over time, all robots will converge on preferring the short path. More details are given in the work by Montes de Oca et al. (2011) and in Sect. 4.1.

We chose this collective decision-making behaviour as a case-study for Bio-PEPA since it displays several interesting characteristics common to many swarm robotics systems:

- Simplicity: the collective behaviour is simple enough that it is possible to analyse it without being hampered by the implementation details.
- Direct cooperation: the robots must form groups of three to carry the objects.
- Indirect cooperation: the vote process creates an opinion dynamics that lets the robots collectively choose the shortest path.
- Space and time aspects: space and time play an important role and must be carefully modelled. In particular, the voting process is spatially located in the start area and only the robot in the start area at a given moment can take part in it. Additionally, the time

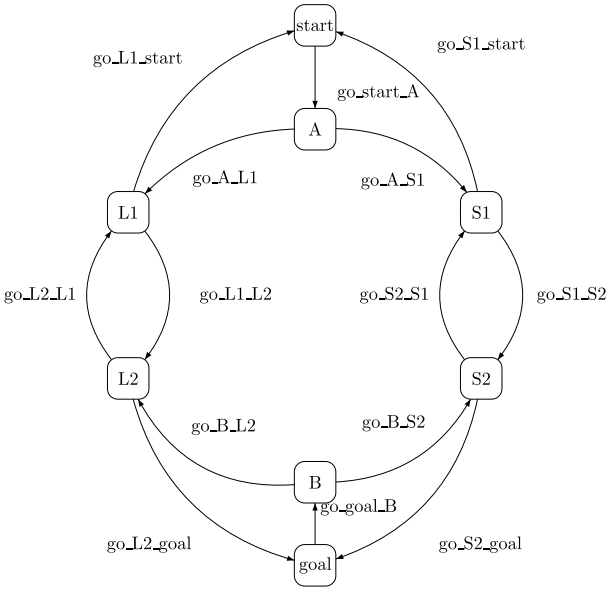

Fig. 2 Locations and transitions of robots in the Bio-PEPA specification

necessary for the robots to carry an object, which depends on the length of the chosen path, affects the opinion dynamics.

This system has been analysed in several other works: Montes de Oca et al. (2011) presented a simple fluid flow analysis and a Monte Carlo simulation, Scheidler (2011) presented a more complex fluid flow analysis, Valentini et al. (2013) presented an analysis based on absorbing Markov chains. In these works, each analysis was based on a different model. In our paper, we use a single Bio-PEPA description to perform three different kinds of analysis.

4.1 The Bio-PEPA specification

In the remaining part of this section we present the Bio-PEPA specification of the system. The full specification can be found in the supplementary material (Massink et al. 2012a). As shown in Fig. 2, the system is described by eight Bio-PEPA *locations*: two boundary locations, `start` and `goal`; a location A where robot teams select the short or long path to `goal` according to the decision taken when leaving `start` and, similarly, location B, where robot teams select the short or long path back to `start`, again according to the previously taken decision. We have then two locations for each path, $L1$ and $L2$ for the long path and $S1$ and $S2$ for the short one. We also define a set of Bio-PEPA *species* to specify the behaviour of the robots. For example in `start` we distinguish two species of robots: those that the last time returned via the short path, denoted as *Robo_start_fromS*, and those that returned via the long path, denoted as *Robo_start_fromL*. In the sequel we will refer to these two groups also as the *S-population* and the *L-population*, respectively. Similarly, other locations contain populations of teams of robots that move in the direction from the start area to the goal area and those that move in the opposite direction. For example, in location $S1$ we can have *Teams_S1_StoG* and *Teams_S1_GtoS*, where *StoG* denotes the direction from the start area to the goal area and *GtoS* the opposite direction.

The Bio-PEPA fragment below specifies the behaviour of a robot. Robots leave the start area in groups of three. Each group is randomly composed by either three robots from the

Fig. 3 Two components synchronised on action $S2L1$

$$\begin{aligned} Robo\_start\_fromS = {} & (allS, 3)\downarrow Robo\_start\_fromS@start+ \\ & (S2L1, 2)\downarrow Robo\_start\_fromS@start+ \\ & (S1L2, 1)\downarrow Robo\_start\_fromS@start+ \\ & (go\_S1\_start, 3)\uparrow Robo\_start\_fromS@start; \end{aligned}$$

$$\begin{aligned} Teams\_A\_S = {} & (allS, 1)\uparrow Teams\_A\_S@A+ \\ & (S2L1, 1)\uparrow Teams\_A\_S@A+ \\ & go\_A\_S1\downarrow Teams\_A\_S@A; \end{aligned}$$

S-population, three from the L-population or two from S and one from L or two from L and one from S. These combinations are modelled as four different actions: *allS*, *allL*, $S2L1$ and $S1L2$. In Bio-PEPA the formation of teams of robots is modelled by the coefficient that indicates how many entities are involved in an action. For example, upon action *allS* three robots of the S-population leave `start` (indicated by $(allS, 3)\downarrow$), to form an additional *team* of robots in choice point *A* (indicated by $(allS, 1)\uparrow$ in *Teams_A_S*) which is ready to take the short path when the team continues its journey towards the goal area (population *Teams_A_S@A*). Since action *allS* is shared between the species components *Robo_start_fromS* and *Teams_A_S* this movement occurs simultaneously with the rate of action *allS* that will be defined later on.

In a similar way, upon action $S2L1$, which is present in three components (*Robo_start_fromS*, *Teams_A_S* and *Robo_start_fromL*, of which the first two are shown in Fig. 3), all three components synchronise, resulting in two robots from the S-population and one from the L-population leaving the start area and forming at the same time 1 new team in choice point *A* in the population *Teams_A_S*, that is, those teams in choice point *A* that decided to take the short path. The synchronisation pattern of the components is given by the model component shown later on. The excerpt above only shows the behaviour of teams voting for the short path. The behaviour of those voting for the long path is similar and omitted for reasons of space. For the same reason also the behaviour of teams moving between different locations is not shown.

The actions denoting teams of robots leaving the start area need to occur with appropriate rates. For example, a group of three robots that are all from the S-population has a probability to occur equal to

$$pSSS = \frac{(RSS)}{(RSS) + (RSL)} \cdot \frac{(RSS - 1)}{(RSS - 1) + (RSL)} \cdot \frac{(RSS - 2)}{(RSS - 2) + (RSL)}$$

where, for the sake of readability, *Robo_start_fromL@start* is abbreviated by *RSL* and *RSS* abbreviates *Robo_start_fromS@start*. The rate with which the action occurs is the product of the probability of the action to occur and the rate of leaving the start area. A similar probability *pLLL* and rate can be defined for a group of three robots from the L-population.

The probability to extract two robots from the S-population and one from the L-population is:

$$pSSL = \frac{(RSS)}{(RSS) + (RSL)} \cdot \frac{(RSS - 1)}{(RSS - 1) + (RSL)} \cdot \frac{(RSL)}{(RSS - 2) + (RSL)}$$

Similarly, probabilities for *pSLS*, *pLSS*, *pLLS*, *pLSL* and *pSLL* can be defined. Therefore, the total probability that two, out of the three members of a team, vote for the short path is $pSSL + pSLS + pLSS$, while for the long path it is $pSLL + pLSL + pLLS$. Consequently, the rates of actions $S2L1$ and $S1L2$ can now be defined as $(pSSL + pSLS + pLSS) \cdot move$ and $(pSLL + pLSL + pLLS) \cdot move$, respectively. Note that the sum of these six probabilities and *pSSS* and *pLLL* amounts to 1. So the total rate at which teams of robots leave the

start area is constant and given by the parameter 'move'. The rate at which teams move from A to $S1$ and to $L1$ is also dependent on the number of teams present in A and are $walk\_normal \cdot Teams\_A\_S@A$ and $walk\_normal \cdot Teams\_A\_L@A$, respectively. The rate parameter $walk\_normal$ specifies the time it takes a robot team to move from choice-point A to the first section of a path.

The overall system definition shows the initial size of robot populations in each location. The overall robot behaviour is defined using cooperation on shared actions (see Sect. 3 for a definition and example):

$$
\begin{aligned}
&Robo\_start\_fromS@start(SS) \underset{*}{\bowtie} Robo\_start\_fromL@start(SL) \underset{*}{\bowtie} \\
&Teams\_A\_S@A(0) \underset{*}{\bowtie} Teams\_A\_L@A(0) \underset{*}{\bowtie} \\
&Teams\_S1\_StoG@S1(0) \underset{*}{\bowtie} Teams\_S1\_GtoS@S1(0) \underset{*}{\bowtie} \\
&Teams\_S2\_StoG@S2(0) \underset{*}{\bowtie} Teams\_S2\_GtoS@S2(0) \underset{*}{\bowtie} \\
&Teams\_L1\_StoG@L1(0) \underset{*}{\bowtie} Teams\_L1\_GtoS@L1(0) \underset{*}{\bowtie} \\
&Teams\_L2\_StoG@L2(0) \underset{*}{\bowtie} Teams\_L2\_GtoS@L2(0) \underset{*}{\bowtie} \\
&Teams\_goal\_fromS@goal(0) \underset{*}{\bowtie} Teams\_goal\_fromL@goal(0) \underset{*}{\bowtie} \\
&Teams\_B\_fromS@B(0) \underset{*}{\bowtie} Teams\_B\_fromL@B(0)
\end{aligned}
$$

where the number SS in $Robo\_start\_fromS@start(SS)$ (resp. SL) is the initial size of the robot S-population (resp. L-population) present in the start area ($@start$).

There is a further issue to consider which is the way to model the length of the paths. This can be done in two ways. The first is to model each path by two sections, as illustrated above, and vary the time it takes teams to traverse these sections by choosing a different rate for the movement between sections on the short and the long path, respectively. However, as also discussed in Montes de Oca et al. (2011), this model has the disadvantage that the duration of path traversal is essentially modelled by a *short* series of exponential distributions which in general approximates the average duration well, but not the variability. It therefore does not reflect very well real robot behaviour. An alternative is to choose the same rate for each section and to vary the number of sections on each path to model their difference in length. This way the traversal time of a path is modelled by a sequence of say m exponentially distributed random variables with rate λ, also known as an Erlang distribution, using the well-known method of stages (see Kleinrock 1975, p. 119).[4]

We model the two paths of the environment with eight sections for the short path and 15 sections for the long path. Each section takes, on average, ten time units to traverse by a robot team. This is modelled in the system by defining the rate $walk\_normal = 0.1$. Considering also the movements from the choice points to the path and those from the path to the start area and the goal area, the short path takes on average 100 time units to traverse, and the long one 170. This is comparable to the latency periods used in Montes de Oca et al. (2011) and provides a good approximation of the actual variability observed in robot movement. Other free variables of the model not provided in Montes de Oca et al. (2011) have been selected by us.

The analysis presented in Montes de Oca et al. (2011), that we will use to compare our results with in the next section, is based on the assumption that there is a constant number of teams, say k, active (that is, not present in the start area) at any time. The number k is a parameter of the model. In the Bio-PEPA model we use parameter min_start which

[4]The mean (variance, resp.) of an Erlang distribution with m phases of rate λ is m/λ (m/λ^2 resp.). Thus an appropriate choice of m and λ can guarantee the required values for the mean and variance, approximating a normal distribution.

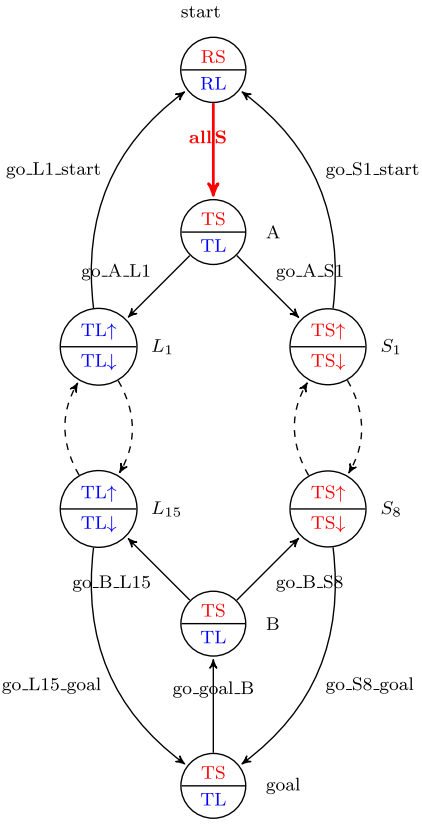

Fig. 4 Graphical representation of the full Bio-PEPA swarm decision-making model

specifies the minimum number of robots in the start area at any time. As we will see in the next section, after a short initial transitory period, the following holds in the model with a good approximation: $k = (32 - min\_start)/3$.[5]

A more detailed graphical representation of the complete Bio-PEPA specification is presented in Fig. 4. It presents the various locations, with eight locations on the short path (S_1 through S_8) and 15 on the long path (L_1 through L_{15}). In each location there are two populations. Their names have been abbreviated for reasons of presentation. Names starting by R indicate populations of robots, names starting by T refer to populations of teams. Names ending in S refer to populations of elements that are in favour of the short path, those ending in L refer to elements in favour of the long path. The arrows after the names of the populations in the locations on the paths indicate the direction of movement of the elements of the population, so those moving from the start area to the goal area are indicated by an arrow pointing downwards, whereas those moving from the goal to the start are indicated by an arrow pointing upwards. Also the actions that label the transitions between locations correspond to those in the Bio-PEPA specification; however, for reasons of readability, only one action is shown (*allS*) of all those between the start area and choice point A.

[5]In Bio-PEPA, one can make use of a predefined function H which takes a number as an argument. If this number is zero, H returns zero, otherwise it returns 1. To guarantee a minimum number *min_start* of robots in the start area, the rate of action $S2L1$ can then be defined as: $S2L1 = (pSSL + pSLS + pLSS) * move * H((RSS + RSL) - min\_start)$; the same must be done for the other related rates.

5 Analysis

For the analysis, in this section we consider a Bio-PEPA swarm decision-making specification with a population of 32 robots, unless stated otherwise. We furthermore consider the following parameters for the specification: initially $SS = 16$ and $SL = 16$, $move = 0.28$, $walk\_normal = 0.1$.

In the following, we illustrate three different forms of analysis of the same Bio-PEPA specification and compare their results with those validated in the literature (Montes de Oca et al. 2011). Good correspondence of the results would mean that Bio-PEPA is a viable formal language to model this kind of swarm robotics system but with the additional advantage that a single specification can be used for multiple kinds of analysis. This also means that, due to the precise and unambiguous mathematical semantics of the language, the results of the different analyses are formally related and coherent since they are systematically derived from the same specification.

In the following sections, the directly relevant aspects of each kind of analysis are recalled, in particular for what concerns its connection to Bio-PEPA. We omitted such a description for stochastic simulation because we assume readers to be familiar with this well-known and widely applied method in the context of swarm robotics. Following each description, the application of the method is illustrated on the Bio-PEPA swarm robotics model and results are discussed and compared with those in the literature.

5.1 Stochastic simulation

The first kind of analysis uses stochastic simulation to check the average number of active teams in the system over time for different assumptions on the minimal number of robots that are present in the start area. The Bio-PEPA tool suite relies on an implementation of Gillespie's stochastic simulation algorithm (Gillespie 1977). The original algorithm assumed that only interactions with at most two species were used in the model and that the rates were simple products of a constant and a population size. In the Bio-PEPA model this is indeed the case with the exception of the rate functions involved in the team formations, which are slightly more general, but which do not cause any problem.

Figure 5 presents two stochastic simulation results (average over 10 simulation runs) for $min\_start = 5$ (Fig. 5, left) and $min\_start = 2$ (Fig. 5, right), showing the number of robots on both paths and in the start area and also the number of teams on each path. The figure also shows that the number of active teams on the paths quickly increases to 9 (resp. 10) and then stabilises at that level. This means that the rate at which robots leave the start area,

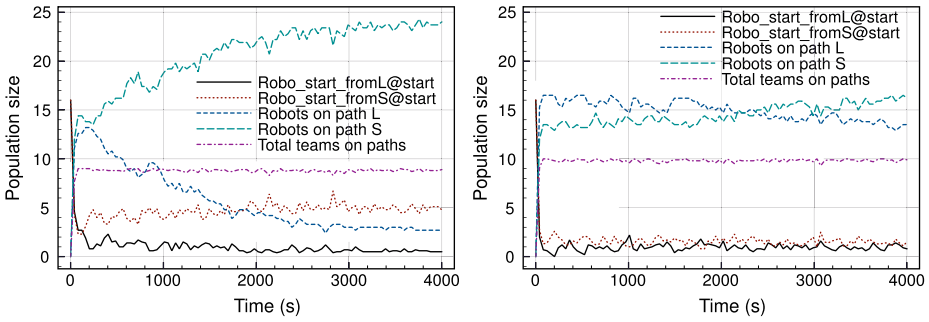

Fig. 5 Number of active teams for $min\_start = 5$ (*left*) and $min\_start = 2$ for $move = 0.28$ (*right*)

i.e., $move = 0.28$, is sufficiently high to quickly reach a situation that presents the desired number of active teams. This makes it possible to compare the results of this analysis with the results obtained with the physics-based simulation and Monte Carlo simulation reported in Montes de Oca et al. (2011), which will be discussed in Sect. 5.2.3.

5.2 Statistical model checking

Model checking has first been developed in a non-quantitative setting. Pioneers of this technique, starting their developments in the early 1980s, are, among others, Clarke et al. (2009) and Holzmann (1991). This verification technology provides algorithmic means to determine whether an abstract model of, for example, a hardware or software component, satisfies a formal property expressed as a temporal logic formula. Moreover, when the formula is found *not* to be satisfied, it can provide automatically a counterexample that illustrates the potential source of the problem.

At the heart of model checking algorithms are efficient and flexible search procedures used to verify behavioural properties on the finite, but potentially huge, state space, typically represented as a graph, of (a model of) a real-world system. Concurrent systems often produce huge state spaces which is due to the largely asynchronous composition of a considerable number of processes.

More recently, model checking techniques have been extended to deal with quantitative performance aspects of concurrent systems and related probabilistic versions of the temporal logics for the specification of quantitative properties have been formulated, such as the Continuous Stochastic Logic (Aziz et al. 2000; Baier et al. 1999). Efficient model checking methods for these stochastic variants are based on well-known numerical algorithms for the calculation of standard measures of continuous/discrete time Markov chains, like transient or steady state probabilities.

Although *stochastic model checking*, as this probabilistic variant is called, may generate very accurate answers, it relies on building the state space of the complete underlying Markov chain of the abstract system model, which (currently) restricts its realistic applicability to system models with a number of states in the order of 10^7.

A recently proposed related way to analyse large concurrent systems is via *statistical* model checking. In its most general form, statistical model checking is an analysis method in which a logic formula, formalising a probabilistic property of interest, is automatically checked against a set of randomly generated simulation runs of a high-level model of the system. The probability that the formula holds for the model is then estimated via *statistical analysis* rather than numerical analysis. This has various consequences. On the one hand, statistical model checking can deal with system models that have very large state spaces because only a set of paths need to be generated instead of the whole state space. On the other hand, in cases in which high accuracy is required the set of paths that need to be generated may be huge as well. So, in case of very large systems and when high accuracy is not the main issue, statistical model checking may be the right option.

Various statistical techniques have been implemented and added to existing stochastic model checkers such as PRISM (Kwiatkowska et al. 2011). Among these are techniques to approximate the probability with which a formula holds and techniques to establish whether such a probability is above or below a certain given bound. The former is based on various confidence interval methods, whereas the latter is based on hypothesis testing, in particular Wald's sequential probability ratio test (Younes et al. 2006).

For the analysis of properties of the Bio-PEPA model we will make use of two of the above techniques, in particular confidence interval methods to estimate probability and rewards. Before we present these two techniques in more detail, we first give an overview of

the type of properties of interest for the case study. Since we only deal with CTMCs, we will only review a relevant selection of Continuous Stochastic Logic properties. For a more complete overview and further details we refer the interested reader to Nimal (2010).

5.2.1 Performance properties

In the following we will encounter two types of performance property: (*bounded*) *until formulae* and *reward formulae*. The first type is given by the following grammar:

$$\begin{aligned} \mathit{property} ::= {} & P_{=?}[\mathit{proposition}\ U^{\leq t}\ \mathit{proposition}] \\ | \quad & P_{=?}[\mathit{proposition}\ U\ \mathit{proposition}] \\ | \quad & P_{\trianglelefteq b}[\mathit{proposition}\ U^{\leq t}\ \mathit{proposition}] \end{aligned}$$

where $\phi_1 U^{\leq t} \phi_2$ holds on a path σ of the model if ϕ_2 is true for a state on the path reached within time t and that until then ϕ_1 is true. More formally, for σ a path of the model,

$$\sigma \models \phi_1 U^{\leq t} \phi_2 \equiv \exists t_1 \leq t . \sigma(t_1) \models \phi_2 \wedge \forall t_0 < t_1 . \sigma(t_0) \models \phi_1$$

were $\sigma(t)$ is the state in σ occupied at time t and $\sigma \models \phi$ means that path σ satisfies formula ϕ. In the variant without time bound t it is required that eventually a state is reached in σ in which ϕ_2 holds, and that all preceding states satisfy ϕ_1. Again, more formally:

$$\sigma \models \phi_1 U \phi_2 \equiv \exists t_1 . \sigma(t_1) \models \phi_2 \wedge \forall t_0 < t_1 . \sigma(t_0) \models \phi_1$$

Note that in *statistical* model checking only paths of a maximum length, say ℓ, are considered: ℓ is one of the parameters of the model checking algorithm. This parameter should be sufficiently large for *unbounded until formulae* to make sense and needs to be considered carefully on a case by case basis. An alternative is to consider only *bounded until formulae*.

$P_{=?}[\phi]$ denotes the probability measure of the set of paths of the model that satisfy ϕ. $P_{\trianglelefteq b}[\phi]$ is the property stating that the probability measure of the set of paths satisfying ϕ is bounded by b, where b is a probability value and $\trianglelefteq \in \{<, \leq, >, \geq\}$.

Propositions are given by the following grammar, where $\wedge$ denotes conjunction, | disjunction and ! negation and the label needs to be defined separately:

$$\begin{aligned} \mathit{proposition} ::= {} & \mathit{label} \\ | \quad & \mathit{proposition} \wedge \mathit{label} \\ | \quad & !\,\mathit{proposition} \\ | \quad & \mathit{proposition} \mid \mathit{label} \end{aligned}$$

The second type of formulae that we will consider are *reward formulae*. *Reward formulae* make use of *reward structures* that are added to the abstract system model. *Reward structures* can be used, for example, to count the number of times that certain actions occur, such as the formation of a team. This is captured by a *reward structure* that accumulates the number of occurrences of the event of interest. They can also be used to record the amount of time that passes until a certain event occurs, for example the time until the first team reaches the goal area. This is captured by a *reward structure* that accumulates time with rate 1 in every state of the system. The accumulation of time stops as soon as the specified event occurs.

An example of a *reward structure* to accumulate time is shown in Fig. 6.

Fig. 6 *Reward structure* to accumulate time

```
reward "total_time"
   true : 1;
endreward
```

We will consider the following types of *reward formula*:

$$\begin{aligned} \mathit{expectation} ::= {} & R\{rwlabel\}_{=?}[F\ proposition] \\ | \ & R\{rwlabel\}_{\trianglelefteq r}[F\ proposition] \\ | \ & R\{rwlabel\}_{=?}[C^{\leq t}] \\ | \ & R\{rwlabel\}_{\trianglelefteq r}[C^{\leq t}] \end{aligned}$$

where *rwlabel* is the name of the *reward structure* in the model which the formula refers to. $R\{\text{“}id\text{”}\}_{=?}[F\ \phi]$ returns the expected reward, using *reward structure* “*id*”, based on the set of randomly generated paths σ, where for each σ the reward is accumulated until a state of σ is reached in which proposition ϕ holds. $R\{\text{“}foo\text{”}\}_{\trianglelefteq r}[F\ \phi]$ compares the expected reward with bound r. It is also possible to obtain the expected cumulative reward up to a certain point in time t, this is expressed by the formula $R\{\text{“}id\text{”}\}_{=?}[C^{\leq t}]$. To compare the expected reward with a given bound r the formula $R\{\text{“}id\text{”}\}_{\trianglelefteq r}[C^{\leq t}]$ is used.

5.2.2 *Confidence interval methods*

An overview of the statistical model checking approach is given in Fig. 7. Confidence interval methods in statistical model checking seek to provide an estimate of the probability that a given property holds for the paths of an abstract system model with a certain level of reliability. A confidence interval is an estimated interval of a certain width $2w$ such that, if the estimation is repeated a number of times, then the real probability lays within this interval $100 \times (1-\alpha)$ % of the times. The reliability parameter α is the level of confidence. Assume, for all $i \in \{1, \ldots, N\}$, that $\{Y_i\}_i$ is a set of realisations of the Bernoulli random variables X_i, where X_i is 1 if property ϕ on a randomly generated path σ of length k holds, and 0 otherwise. It is assumed that all Y_i are independent and identically distributed (i.i.d.) and normally distributed. Using the Central-limit theorem it is possible to derive a lower bound on the required number of paths N that need to be generated in order to provide an estimate of the probability with the required accuracy w and level of confidence α. It is also possible, given α and a desired number of paths N, to calculate the accuracy w. The latter is the approach we will follow in this paper in order to obtain results that can be compared with

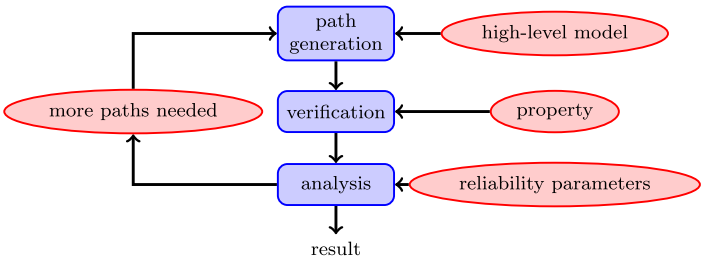

Fig. 7 Overview of the statistical model checking approach

those available in the literature that refer to a given number of sample paths. Several other methods are available as well, such as the asymptotic confidence interval method (ACI) and the approximate model checking technique (AMC) that use different bounds for the minimal sample size N. The latter also uses different notions of accuracy and confidence. For a detailed comparison of these methods we refer to Nimal (2010).

The confidence interval method has also been adapted to estimate the expected value of rewards, that is, for *reward formulae* of type $R_{=?}[\phi]$. Let Σ be a *reward structure* and ϕ a property over paths σ. The random variable $X_{\phi,\Sigma}(\sigma)$ can now be defined to produce a reward value, that is, it is of type $X_{\phi,\Sigma}(\sigma) \in \Omega \to \mathbb{R}^+$. It is assumed that the random variables are i.i.d. and normally distributed. For the rest the method is similar to the confidence interval method described above.

5.2.3 Results

The statistical model checking approaches described above are provided by various model checkers among which the model-checker PRISM described in Kwiatkowska et al. (2011). The Bio-PEPA specification developed in Sect. 4 can be translated automatically into a model expressed in the PRISM input language by the Bio-PEPA tool suite described in Ciocchetta et al. (2009). The translation approach itself is described in Ciocchetta and Hillston (2009). The resulting PRISM specification can be found in the supplementary material provided by Massink et al. (2012a). The PRISM model is a stochastic model having a CTMC as underlying mathematical structure.

One of the principal properties of interest for robot swarm decision making concerns the convergence aspects of the decision strategy. The first concern is whether convergence on one of the paths occurs at all. In principle, mixed decision situations could occur in which the swarm does not converge entirely on a single path. We will show that such a situation occurs with zero probability. A second concern is whether convergence on a single path always occurs eventually, that is, the system does not enter in some form of oscillating behaviour that prevents convergence. Convergence on the short path (*Convergence_on_S*) can be defined as the situation in which each of the 32 robots is either in a team on the short path, or in the S-population in the choice points, the start area or the goal area. In terms of the population sizes in the various locations, convergence on the short path can be formalised as the following proposition:

$$
\begin{aligned}
&3 * (Teams\_S1\_StoG@S1 + \cdots + Teams\_S8\_StoG@S8) + \\
&3 * (Teams\_S1\_GtoS@S1 + \cdots + Teams\_S8\_GtoS@S8) + \\
&3 * Teams\_goal\_fromS@goal + Robo\_start\_fromS@start + \\
&3 * (Teams\_A\_S@A + Teams\_B\_fromS@B) = 32
\end{aligned}
$$

"*Convergence_on_L*" can be defined similarly, but requiring that the above sum is equal to 0 instead of 32.

The formula to obtain an estimate of the probability that the system eventually converges either on the long or on the short path can now be expressed in terms of the formulae that were introduced before:

$$
P =? \left[true\ U\ (\text{“}Convergence\_on\_L\text{”}\ |\ \text{“}Convergence\_on\_S\text{”}) \right] \tag{1}
$$

Recall that $P =?$ is used to compute a probability, and U reads as "until".

For 100 sample paths, a confidence level $\alpha = 0.01$ and a maximum sample path length of 20,000 we obtain the result that for each k ranging from 1 to 10 the system converges to the short or the long path with probability 1. In fact, convergence takes place in each of the sample paths, so mixed decision situations do not occur.

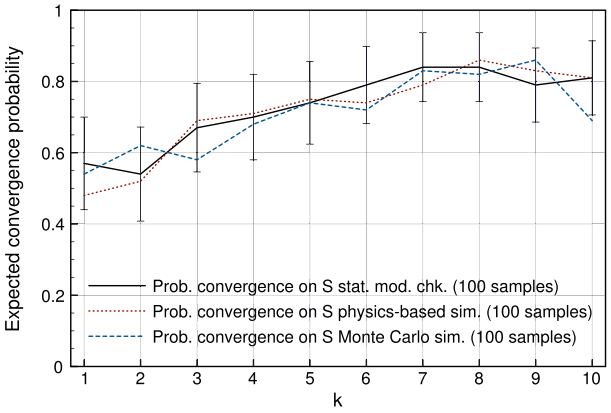

Fig. 8 Probability of convergence on the short path (100 samples). k is the number of active teams in the system

```
reward "teams"
  [go_A_S1] true : 1;
  [go_A_L1] true : 1;
endreward
```

Fig. 9 *Reward structure* to count team formations

The next question of interest is then what is the probability that the system converges on the short path. More precisely, this question should be formulated as "what is the probability that the system did not converge on the long path until it converges on the short path". The latter can be expressed as:

$$P =? [!\text{“}Convergence\_on\_L\text{”}\ U\ \text{“}Convergence\_on\_S\text{”}] \tag{2}$$

where that ! stands for negation.

The analyses of Eq. (2) for a number of teams k ranging from 1 to 10 is shown in Fig. 8 as a solid line. The analyses have been based on 100 random sample paths, a confidence level $\alpha = 0.01$ and a maximal sample path length of 20,000. In the figure the widths of the confidence interval are shown as vertical bars. The results are compared to those obtained via physics-based simulation and Monte Carlo simulation of the same case-study reported in Montes de Oca et al. (2011) and shown as dotted and dashed lines, respectively. The latter are close to the results obtained with the Bio-PEPA specification and well within the error-margins.

The expected number of teams formed until convergence has taken place on the short or the long path can then be analysed by statistical model checking using the logic *reward formula*:[6]

$$R\{\text{“}teams\text{”}\} =? \left[F\ (\text{“}\_Convergence\_on\_S\text{”} | \text{“}\_Convergence\_on\_L\text{”})\right] \tag{3}$$

The formula refers to a *reward structure* "teams" that counts the number of teams that were formed. In terms of the Bio-PEPA model, the formation of teams is directly related to the occurrence of the actions 'go_A_S1' and 'go_A_L1', that is, when teams move from choice point A to one of the paths. The specific *reward structure* required is shown in Fig. 9. Essentially this represents the fact that every time action 'go_A_S1' or 'go_A_L1' occurs, the total number of teams formed so far is incremented by 1.

[6]See Sect. 5.2.1.

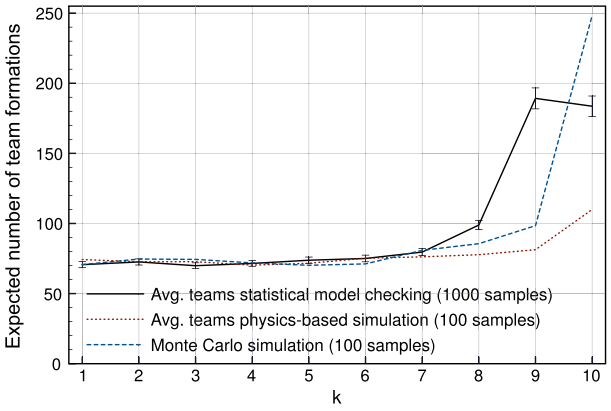

Fig. 10 Expected number of team formations until convergence (1,000 samples). k is the number of active teams in the system

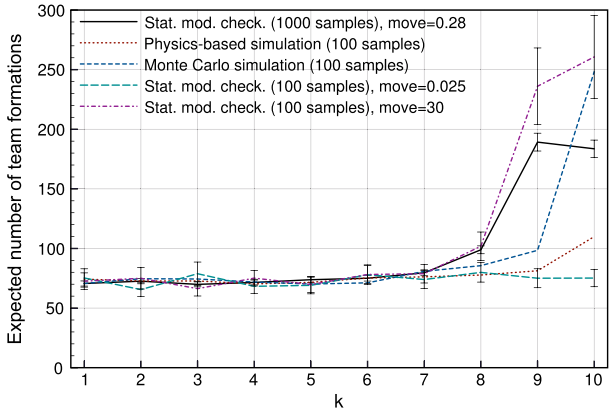

Fig. 11 Expected number of team formations until convergence for different rates at which teams leave the start area in the Bio-PEPA model. k is the number of active teams in the system

Figure 10 shows results on the expected number of team formations until convergence on the short or long path (Eq. (3)) using 1000 samples, $\alpha = 0.01$ and maximal path length of 20,000. The width of the confidence intervals are shown as error-bars.

The results obtained by stochastic model checking, physics-based simulation and Monte Carlo simulation are consistent for values of k up to 7. They diverge for higher values of k. The divergence can be explained by the differences in the underlying models that are used. The Monte Carlo simulations are obtained from an ODE model in which it is assumed that, at any point in time, a constant fixed fraction of the total population is in the start area. Such a fixed fraction can only be maintained if, upon arrival of a team in the start area, a new team forms and leaves the start area immediately. In the Bio-PEPA model this can be approximated by choosing a high rate for the parameter 'move'. In fact, as can be observed in Fig. 11, for $move = 30$ the results of the Bio-PEPA model follow a similar tendency as the results for the Monte Carlo simulation. An explanation for this tendency is that, for high values of k, the system needs more team formations to converge. This is due to the fact that when k is high, a robot team returning to the start area can influence the opinion only of the few robots that are in the start area: five robots for $k = 9$ and only two for $k = 10$.

For $k = 9$ there is a further divergence between the results obtained by Monte Carlo simulation and stochastic model checking. This can most likely be explained by the fact that Monte Carlo simulations start from an initial state in which a large fixed fraction of the population is already out of the start area and distributed over the paths in a particular

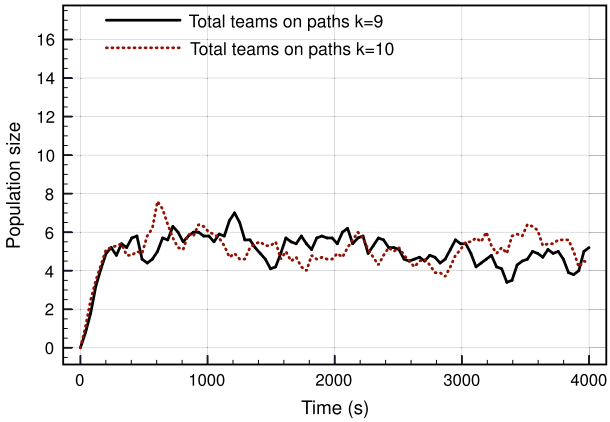

Fig. 12 Number of active teams for $min\_start = 5$ ($k = 9$) and $min\_start = 2$ ($k = 10$) for $move = 0.025$ (average over 10 independent simulation runs)

proportion. The number of team formations needed to reach such a state is not considered in the Monte Carlo simulation. On the other hand, in the Bio-PEPA model (and in the physics-based model) all robots are initially in the start area and subsequently distribute over the two paths. This results in many different intermediate distributions over the paths, which are likely to have an effect on the average number of team formations needed to reach convergence. Furthermore, for $k = 10$ (and $k = 9$ to a somewhat lesser extent) border effects might arise: the system is stretched to an extreme situation in which, at any time, only two robots remain in the start area. This small number is a source of strong stochastic fluctuations that might cause 'accidental' convergence earlier than what one could expect given the size of the population.

The physics-based simulation is based on the assumption that the teams leave the start area on average every 40 seconds, until a number of k teams are active. In the Bio-PEPA specification, this can be modelled by letting $move = 0.025$. The formation of teams is suspended whenever there are k teams active and is resumed when teams return to the start area. For this value of $move$, statistical model checking produces results that are comparable with those produced by the physics-based simulation (as shown in Fig. 11). This can be explained by observing that in the model used for the physics-based simulation when k is high, the average number of active teams is actually substantially lower than the nominal value k. This can also be made visible using simulation of the Bio-PEPA specification as shown in Fig. 12 for an average of the number of active teams over 10 simulation runs for $k = 9$ and $k = 10$ and $move = 0.025$. As a consequence, the number of robots in the start area is larger than the nominal $N - 3k$, which in turn means that there are more robots that provide implicitly feedback on which of the two paths is the shortest. This explains why the expected number of teams formed until convergence obtained with statistical model checking does not differ much from those obtained with physics-based simulation (for $move = 0.025$).

The difference between physics-based simulation and statistical model checking for higher values of the parameter *move* can be explained by looking at the early phases of the experimental runs. In the early phases there are more robots in the start area and they leave that area relatively quickly before feedback from returning teams can be taken into account. This is possibly leading to larger stochastic fluctuations before the system converges on one of the paths, resulting in more team formations.

A similar analysis using the same formula as used for the expected number of teams (see Fig. 9), but substituting *teams* with the reward structure *total_time*, gives the expected time

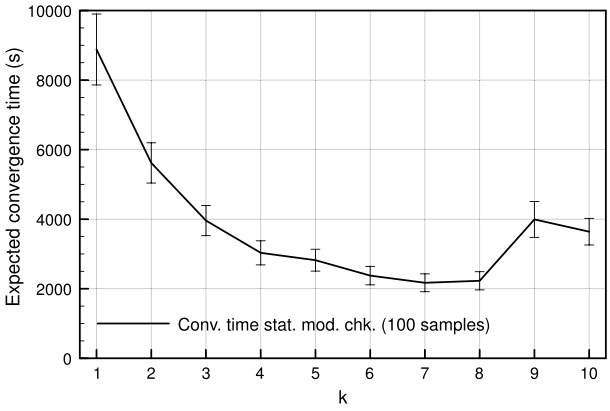

Fig. 13 Expected convergence time (100 samples), $move = 0.28$

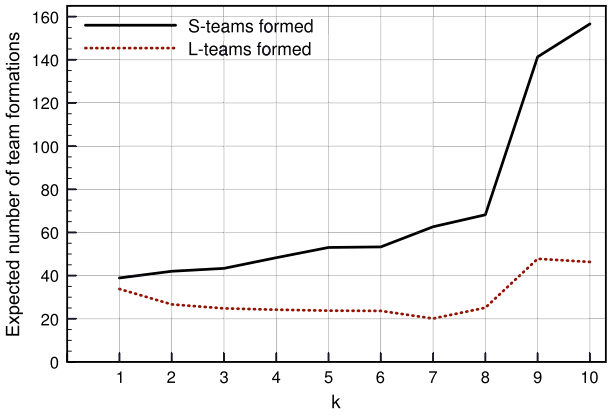

Fig. 14 Expected S-teams and L-teams formed until convergence ($move = 0.28$)

until convergence (for $move = 0.28$). Figure 13 shows the expected convergence time. No data from the literature concerning this aspect are available for comparison.

The total model-checking time to produce the data in Fig. 8 was ca. 10 minutes, those in Fig. 10 ca. 48 minutes and those in Fig. 13 ca. 5 minutes.[7]

By separating the *reward structure* in Fig. 9 into one for the expected number of teams that decide to take the short path (S-teams) and one for those that decide to take the long path (L-teams) the contribution of each kind can be made visible using a *reward formula* similar to that shown in Eq. (3). The result is shown in Fig. 14. For any value of k the number of S-teams is always higher than the number of L-teams. This can be explained by the fact that initially the S-population and the L-population in the start area have equal size and moreover that the probability that the system converges on the short path is more than 50 % in all cases.

5.3 Fluid flow analysis

The third kind of analysis we consider is a fluid flow approximation of the ODE underlying the Bio-PEPA specification. Based on the Bio-PEPA syntax, the underlying ODE model

[7]Model-checking was performed on an iMAC with a 3.2 GHz Intel core i3 processor and 4 GB memory running the MacOS X operating system.

can be generated automatically and in a systematic way, as shown in Hillston (2005) and in Ciocchetta and Hillston (2009), using the Bio-PEPA tool suite presented in Ciocchetta et al. (2009). This provides yet another view on the behavioural aspects of the system. One can, for example, explore numerically the sensitivity of the system to initial values and discover stationary points and other aspects related to stability analysis.

The derivation of ODEs from a Bio-PEPA specification is based on the following steps (see Ciocchetta and Hillston 2009):

1. definition of the stoichiometry $(n \times m)$ matrix D, where n is the number of species and m is the number of actions. The entries of the matrix D are obtained in the following way. For each species component C_i the prefix sub-terms C_{ij}, that is, those of the form $(\alpha_j, \kappa_{ij}) \circ_{\mathrm{P}} S_i @ l$, are considered. Such sub-terms represent the change of the species i as a consequence of action j. If the term contributes to an increase of the population size of the species then the entry is $+\kappa_{ij}$, if it contributes to a decrease then the entry is $-\kappa_{ij}$;
2. definition of the functional rate $(m \times 1)$ vector $\bar{v}_f(t)$ containing the functional rate of each action;
3. association of the variable $x_i(t)$, the expected value of the population size at time t, with each component C_i and the definition of the $(n \times 1)$ vector $\bar{x}(t)$.

The ODE system is then obtained as

$$\frac{d\bar{x}(t)}{dt} = D \times \bar{v}_f(t)$$

with initial population sizes x_{i_0} , for $i = 1, \ldots, n$.

To illustrate these steps, consider the slightly extended small toy Bio-PEPA example introduced in Sect. 3 in which teams can also be dissolved into individual red and blue robots as follows:

$$\begin{aligned} R &\stackrel{def}{=} (mk\_team, 2)\downarrow R + (dis, 2)\uparrow R \\ B &\stackrel{def}{=} (mk\_team, 1)\downarrow B + (dis, 1)\uparrow B \\ T &\stackrel{def}{=} (mk\_team, 1)\uparrow T + (dis, 1)\downarrow T \end{aligned}$$

with the following model component:

$$\left(R(r_0) \underset{\{mk\_team, dis\}}{\bowtie} B(b_0)\right) \underset{\{mk\_team, dis\}}{\bowtie} T(t_0)$$

If we let the functional rates for this toy example be $mk\_team = 0.002 * R * B$ and $dis = 0.2 * T$ we obtain the following ODE:

$$\begin{aligned} \frac{dR(t)}{dt} &= -2.0 \cdot r \cdot R(t) \cdot B(t) + 2.0 \cdot s \cdot T(t) \\ \frac{dB(t)}{dt} &= -1.0 \cdot r \cdot R(t) \cdot B(t) + 1.0 \cdot s \cdot T(t) \\ \frac{dT(t)}{dt} &= +1.0 \cdot r \cdot R(t) \cdot B(t) - 1.0 \cdot s \cdot T(t) \end{aligned}$$

where $r = 0.002$ and $s = 0.2$, to be solved with respect to the initial condition $r_0 = 200$, $b_0 = 100$ and $t_0 = 500$. The numeric solution of this ODE for the above mentioned initial values is shown in Fig. 15.

We now return to the real swarm robotics case study in Bio-PEPA, which leads to a model composed of 54 ordinary differential equations, and discuss various aspects of the relation between stochastic simulation and fluid approximation results. In Fig. 16 we can

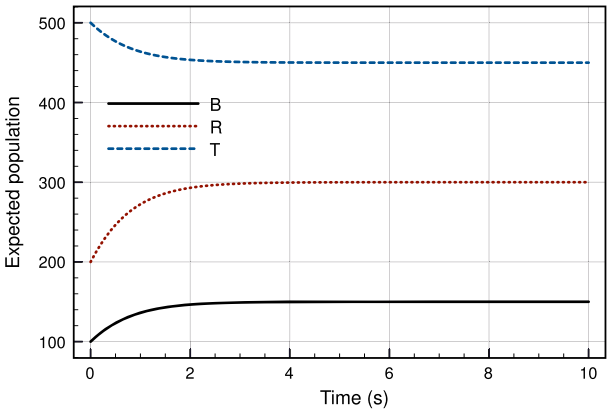

Fig. 15 Expected population sizes of R, B and T over time (ODE) for the small toy example

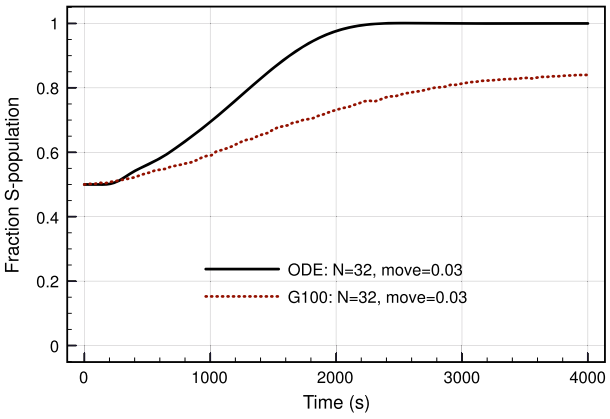

Fig. 16 Fraction of the S-population

observe the total fraction of robots in the S-population over time, that is, both those present in the start area and those in the teams.[8] Clearly, the fluid approximation for a model with initially 32 robots in the start area, of which 16 would vote for the short path and 16 for the long path, predicts that the system converges in 100 % of the cases to the short path for the given initial values. Stochastic simulation over 100 independent runs (G100) shows that such convergence happens only in 85 % of the cases, which corresponds to what we found with statistical model checking for a comparable value of k (see Fig. 8). The difference can be explained by the larger effect of stochastic fluctuations that occur in stochastic simulations of the system when the population is small. The probability that the system 'accidentally' converges on the long path is in that case relatively high. In fact, if a somewhat larger population is considered, a good correspondence can be observed between the fluid approximation and stochastic simulation over 1,000 independent runs (G1000), as shown in Fig. 17 for $N = 320$.

For large populations the probability that the system 'accidentally' converges to the long path tends to zero. In fact, single simulation trajectories tend to approximate the deterministic ODE solution very well for a finite time horizon when the specification satisfies certain

[8] To guarantee continuity of the ODE model, the H-function has been removed and replaced by setting $move = 0.03$ to approximate a scenario in which $k = 7$.

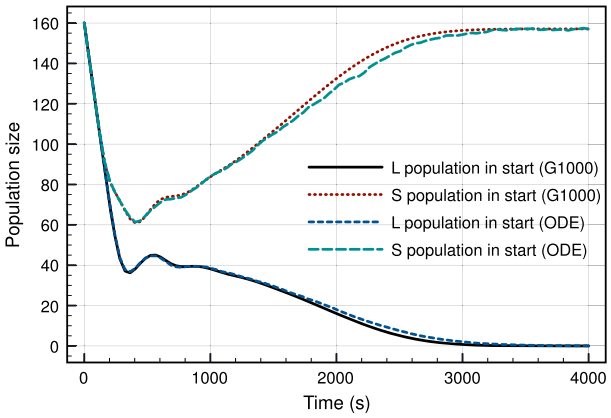

Fig. 17 Fluid approximation (ODE) versus the mean of 1,000 simulation trajectories (G1000), for $NS = NL = 160$. Parameters are $move = 0.03 * 10$ and $walk\_normal = 0.1$

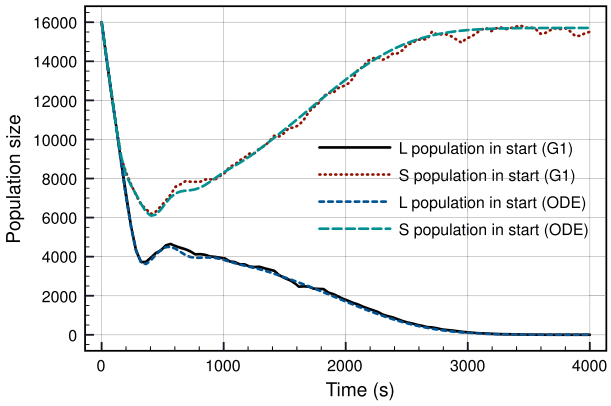

Fig. 18 Fluid approximation (ODE) versus single simulation trajectory (G1), for $NS = NL = 16{,}000$. Parameters are $move = 0.03 * 1{,}000$ and $walk\_normal = 0.1$

scaling conditions and the population considered in the simulation is sufficiently large. An example is shown in Fig. 18 for $N = 32{,}000$. This is a well-studied phenomenon (Kurtz 1970): it has also been applied recently in the context of stochastic process algebra (Tribastone et al. 2012), for an analysis of the double bridge experiment with ants in Bio-PEPA (Massink and Latella 2012) and for the analysis of crowd dynamics (Massink et al. 2011a).

Note that for large populations the model abstracts from the increased risk of collisions between robots on both paths, or, in other words, it is assumed that the size of the paths is scaled in such a way that the number of collisions is proportionally the same as in the model with 32 robots. This model can provide interesting insights in the behaviour of the decision-making strategy as such, that is, abstracting from accidental stochastic fluctuations that occur with small populations. An example of such analysis is given in Fig. 19 which shows a number of ODE trajectories for different initial values of the S-population (*NS*) and the L-population (*NL*) in the start area. The trajectories start from the points indicated on the diagonal and end in one of the two stationary points of the system indicated by a cross at (0, 15,710) and at (3,110, 0). Clearly, the system is bi-stable. For some initial value combination of *NS* between 12,000 and 14,000 and *NL* between 20,000 and 22,000, such that $NS + NL = 32{,}000$, a sudden shift takes place from trajectories converging on the long path to trajectories converging on the short path. A further interesting observation can be made with the help of the graph in Fig. 18. Different phases of behaviour can be distinguished. There is a first phase in which robots leave the start area at a constant rate. This can be

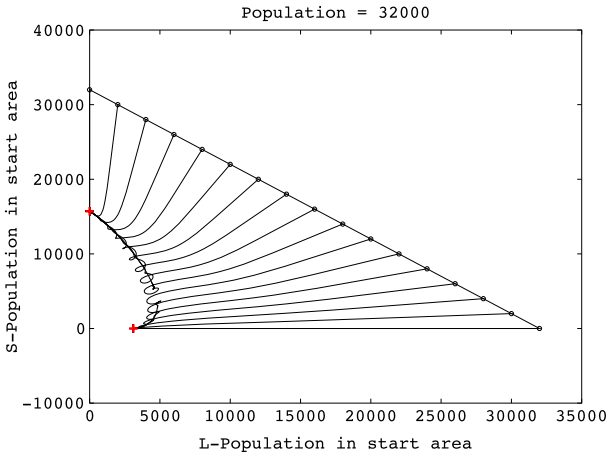

Fig. 19 Phase-space diagram of S-population versus L-population in the start area for a population of 32,000 robots. ODE trajectories for different initial values of NS and NL starting from the diagonal line and finishing in one of the two stationary points indicated by a cross at (0, 15,710) and at (3,110, 0). Parameters are $move = 0.03 * 1{,}000$ and $walk\_normal = 0.1$

observed up to ca. time 200. After that, robots start to return to the start area, first from the short path and later on from the long path, providing feedback to the population in the start area. At about time 600 it can be observed that the feedback is starting to have effect on the decision on which path to take, and an increasing number of teams take the short path rather than the long path with the consequence that the S-population in the start area continues to increase, while the L-population continues to decrease. The various phases in behaviour can also be observed in Fig. 19 where the change due to the arrival of feedback leads to small circle-like shapes in the curves.

Both in Figs. 18 and 19 the number of robots in the start area stabilises around 15,710 in case of convergence on the short path, and around 3,110 in case of convergence on the long path. That means that in the former case about 50% of the total population resides in the start area and that, on average, 5,430 teams circulate on the short path. In the latter case, there are far fewer robots in the start area and on average 9,630 teams circulate on the long path.

Note that Fig. 19 has been obtained via an automatic translation of the Bio-PEPA specification into SBML (Bornstein et al. 2004), which is a standard markup language widely used in systems biology, and then via another translator[9] from SBML into the Octave (Eaton 2002) or equivalently into the Matlab language (Gilat 2004). Such a tool-chain allows further numerical exploration of the generated ODEs with powerful applied mathematics tool suites.

6 Conclusions

In this paper, we analysed a swarm robotics system using Bio-PEPA. The behaviour analysed is a decision-making behaviour originally presented in Montes de Oca et al. (2011). Bio-PEPA (Ciocchetta and Hillston 2009) is a language based on the process algebra PEPA. It was originally developed for the stochastic modelling and analysis of biochemical systems. By using Bio-PEPA we were able to model the swarm robotics system at the microscopic level addressing issues like direct and indirect cooperation, team formation, heterogeneous team behaviours, voting, and certain spatial and temporal aspects.

[9]See http://www.ebi.ac.uk/compneur-srv/sbml/converters/SBMLtoOctave.html.

The main advantage of the use of Bio-PEPA is that it allows the researcher to perform a variety of analyses starting from a single microscopic specification. Among the possible analyses, we performed stochastic simulation, fluid flow (ODE) approximation and statistical (stochastic) model checking. The possibility to perform different analyses from the same specification reduces the effort necessary for the analysis process, while preserving the mutual consistency of the results.

In the presented analysis of the collective decision-making behaviour, we show that using Bio-PEPA we obtain results compatible with those obtained using other approaches, such as the results presented in Montes de Oca et al. (2011) via physics-based simulation and Monte Carlo simulation.

Our long term goal is to extend Bio-PEPA to facilitate the modelling and analysis process of swarm robotics systems. We believe that this could promote a more widespread uptake of modelling and analysis in swarm robotics.

Currently, Bio-PEPA provides relatively limited mechanisms to model and analyse more sophisticated spatial and temporal concepts. In future work, we plan to address this. We also plan to develop formal methods to further explore non-linear behavioural aspects using numerical techniques. Of particular interest are a further integration of formal modelling and the generation of phase diagrams and bifurcation diagrams to obtain insight in the stability aspects of non-linear systems. Furthermore, the development of advanced model-checking techniques for swarm robotics that exploit fluid approximation along the lines of the work presented in Bortolussi and Hillston (2012) is of direct interest too.

Another open problem that we plan to tackle is the gap between Bio-PEPA models and physics-based simulations. Currently, there is no direct link between a Bio-PEPA model and a physics-based simulations of the same system, neither from model to simulation, nor from simulation to model. This passage must be done manually relying on ingenuity and experience. As future work, we plan to create ways to partially or completely automatise these passages. We think that this could greatly stimulate the use of Bio-PEPA, as it would reduce the effort necessary to model and analyse a system.

Acknowledgements The research leading to the results presented in this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013)/ERC grant agreement no. 246939, and by the EU project ASCENS, 257414. Manuele Brambilla, Mauro Birattari and Marco Dorigo acknowledge support from the F.R.S.-FNRS of Belgium's Wallonia-Brussels Federation. Diego Latella has been partially supported by Project TRACE-IT—PAR FAS 2007–2013—Regione Toscana. The authors would like to thank Stephen Gilmore and Allan Clark (Edinburgh University) for their help with the Bio-PEPA tool suite and templates.

References

Aldini, A., Bernardo, M., & Corradini, F. (2010). *A process algebraic approach to software architecture design*. Heidelberg: Springer.

Aziz, A., Sanwal, K., Singhal, V., & Brayton, R. (2000). Model checking continuous time Markov chains. *ACM Transactions on Computational Logic*, *1*(1), 162–170.

Baier, C., Katoen, J.-P., & Hermanns, H. (1999). Approximate symbolic model checking of continuous-time Markov chains. In *Lecture notes in computer science: Vol. 1664. Concur '99* (pp. 146–162). Heidelberg: Springer.

Benkirane, S., Norman, R., Scott, E., & Shankland, C. (2012). Measles epidemics and PEPA: an exploration of historic disease dynamics using process algebra. In D. Giannakopoulou & D. Méry (Eds.), *Lecture notes in computer science: Vol. 7436. FM 2012: formal methods* (pp. 101–115). Berlin: Springer.

Bergstra, J., Ponse, A., & Smolka, S. (Eds.) (2001). *Handbook of process algebra*. Amsterdam: Elsevier.

Bornstein, B., Doyle, J., Finney, A., Funahashi, A., Hucka, M., Keating, S., Kovitz, H. K. B., Matthews, J., Shapiro, B., & Schilstra, M. (2004). Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (SBML) project. *Systems Biology*, *1*, 4153.

Bortolussi, L., & Hillston, J. (2012). Fluid model checking. In M. Koutny & I. Ulidowski (Eds.), *Lecture notes in computer science: Vol. 7454. CONCUR* (pp. 333–347). Berlin: Springer.

Brambilla, M., Pinciroli, C., Birattari, M., & Dorigo, M. (2012). Property-driven design for swarm robotics. In *Proceedings of 11th international conference on autonomous agents and multiagent systems (AAMAS 2012)* (pp. 139–146). IFAAMAS.

Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, *7*(1), 1–41.

Burch, J., Clarke, E., McMillan, K., & Dill, D. (1990). Sequential circuit verification using symbolic model checking. In *Proceedings of the 27th design automation conference* (pp. 46–51). Washington: IEEE Press.

Ciocchetta, F., & Hillston, J. (2008). Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. *Electronic Notes in Theoretical Computer Science*, *194*(3), 103–117.

Ciocchetta, F., & Hillston, J. (2009). Bio-PEPA: a framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, *410*(33–34), 3065–3084.

Ciocchetta, F., & Hillston, J. (2012). Bio-PEPA http://www.biopepa.org. Last checked on October 2012.

Ciocchetta, F., Duguid, A., Gilmore, S., Guerriero, M. L., & Hillston, J. (2009). The Bio-PEPA tool suite. In *Proceedings of the 6th international conference on quantitative evaluation of SysTems (QEST 2009)* (pp. 309–310). Washington: IEEE Computer Society.

Clarke, E. M., Emerson, E. A., & Sifakis, J. (2009). Model checking: algorithmic verification and debugging. *Communications of the ACM*, *52*(11), 74–84.

Dixon, C., Winfield, A., & Fisher, M. (2011). Towards temporal verification of emergent behaviours in swarm robotic systems. In *Lecture notes in computer science: Vol. 6856. Towards autonomous robotic systems* (pp. 336–347). Heidelberg: Springer.

Eaton, J. W. (2002). *GNU octave manual*. London: Network Theory Ltd.

Evans, W., Mermoud, G., & Martinoli, A. (2010). Comparing and modeling distributed control strategies for miniature self-assembling robots. In *IEEE international conference on robotics and automation (ICRA)* (pp. 1438–1445).

Gilat, A. (2004). *MATLAB: an introduction with applications* (2nd ed.). New York: Wiley.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, *81*(25), 2340–2361.

Goss, S., Aron, S., Deneubourg, J.-L., & Pasteels, J. (1989). Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, *76*, 579–581.

Havelund, K., Lowry, M., & Penix, J. (2001). Formal analysis of a space-craft controller using spin. *IEEE Transactions on Software Engineering*, *27*(8), 749–765.

Hermanns, H., Herzog, U., & Katoen, J.-P. (2002). Process algebra for performance evaluation. *Theoretical Computer Science*, *274*(1–2), 43–87.

Hillston, J. (1996). *Distinguished dissertation in computer science: A compositional approach to performance modelling*. Cambridge: Cambridge University Press.

Hillston, J. (2005). Fluid flow approximation of PEPA models. In *Proceedings of the 2th international conference on quantitative evaluation of SysTems (QEST 2005)* (pp. 33–43). Washington: IEEE Computer Society.

Holzmann, G. J. (1991). *Design and validation of computer protocols*. Upper Saddle River: Prentice-Hall

Kleinrock, L. (1975). *Queueing systems: Vol. 1. Theory*. New York: Wiley.

Konur, S., Dixon, C., & Fisher, M. (2012). Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, *60*(2), 199–213.

Kurtz, T. (1970). Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, *7*, 49–58.

Kwiatkowska, M., Norman, G., & Parker, D. (2011). PRISM 4.0: verification of probabilistic real-time systems. In *Lecture notes in computer science: Vol. 6806. Proceedings of 23rd international conference on computer aided verification (CAV'11)* (pp. 585–591). Heidelberg: Springer.

Lerman, K., Martinoli, A., & Galstyan, A. (2005). A review of probabilistic macroscopic models for swarm robotic systems. In *Lecture notes in computer science: Vol. 3342. Swarm robotics* (pp. 143–152). Heidelberg: Springer.

Martinoli, A., Easton, K., & Agassounon, W. (2004). Modeling swarm robotic systems: a case study in collaborative distributed manipulation. *International Journal of Robotics Research*, *23*(4–5), 415–436.

Massink, M., & Latella, D. (2012). Fluid analysis of foraging ants. In M. Sirjani (Ed.), *Lecture notes in computer science: Vol. 7274. Coordination* (pp. 152–165). Heidelberg: Springer.

Massink, M., Latella, D., Bracciali, A., & Hillston, J. (2011a). Modelling non-linear crowd dynamics in Bio-PEPA. In D. Giannakopoulou & F. Orejas (Eds.), *Lecture notes in computer science: Vol. 6603. FASE* (pp. 96–110). Heidelberg: Springer.

Massink, M., Latella, D., Bracciali, A., & Hillston, J. (2011b). Modelling non-linear crowd dynamics in Bio-PEPA. In *Lecture notes in computer science: Vol. 6603. Fundamental approaches to software engineering* (pp. 96–110). Heidelberg: Springer.

Massink, M., Brambilla, M., Latella, D., Dorigo, M., & Birattari, M. (2012a). Analysing robot swarm decision-making with Bio-PEPA: complete data. Supplementary information page at http://iridia.ulb.ac.be/supp/IridiaSupp2012-012/.

Massink, M., Brambilla, M., Latella, D., Dorigo, M., & Birattari, M. (2012b). Analysing robot swarm decision-making with Bio-PEPA. In *Lecture notes in computer science: Vol. 7461. Swarm intelligence* (pp. 25–36). Heidelberg: Springer.

Massink, M., Latella, D., Bracciali, A., Harrison, M., & Hillston, J. (2012c). Scalable context-dependent analysis of emergency egress models. *Formal Aspects of Computing*, *24*(2), 267–302. doi:10.1007/s00165-011-0188-1. Published online: 03 July 2011.

Mather, T., & Hsieh, M. (2012). Ensemble synthesis of distributed control and communication strategies. In *IEEE international conference on robotics and automation (ICRA)* (pp. 4248–4253).

Montes de Oca, M. A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., & Dorigo, M. (2011). Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making. *Swarm Intelligence*, *5*(3–4), 305–327.

Napp, N., Burden, S., & Klavins, E. (2011). Setpoint regulation for stochastically interacting robots. *Autonomous Robots*, *30*, 57–71.

Nimal, V. (2010). *Statistical approaches for probabilistic model checking*. MSc mini-project dissertation, Oxford University Computing Laboratory

Sahin, E. (2005). Swarm robotics: from sources of inspiration to domains of application. In *Lecture notes in computer science: Vol. 3342. Swarm robotics* (pp. 10–20). Heidelberg: Springer.

Scheidler, A. (2011). Dynamics of majority rule with differential latencies. *Physical Review E*, *83*, 031116.

Tribastone, M., Gilmore, S., & Hillston, J. (2012). Scalable differential analysis of process algebra models. *IEEE Transactions on Software Engineering*, *38*(1), 205–219.

Tschaikowski, M., & Tribastone, M. (2012). Exact fluid lumpability for Markovian process algebra. In M. Koutny & I. Ulidowski (Eds.), *Lecture notes in computer science: Vol. 7454. CONCUR 2012—concurrency theory: 23rd international conference* (pp. 380–394). Heidelberg: Springer.

Valentini, G., Birattari, M., & Dorigo, M. (2013). Majority rule with differential latency: an absorbing Markov chain to model consensus. In *European conference on complex systems (ECCS'12)*.

Younes, H. L. S., Kwiatkowska, M. Z., Norman, G., & Parker, D. (2006). Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer*, *8*(3), 216–228.

Zarzhitsky, D., Spears, D., Thayer, D., & Spears, W. (2005). Agent-based chemical plume tracing using fluid dynamics. In M. Hinchey, J. Rash, W. Truszkowski, & C. Rouff (Eds.), *Lecture notes in computer science: Vol. 3228. Formal approaches to agent-based systems* (pp. 146–160). Heidelberg: Springer.