QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

http://www.quanticol.eu

TR-QC-01-2013

On-the-fly PCTL Fast Mean-Field Model-Checking for Self-organising Coordination - Preliminary version

Revision: 0.0; Dec 17, 2013

Author(s): Diego Latella (CNR), Michele Loreti (IMT) and Mieke Massink (CNR)

Publication date: Dec. 17, 2013

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: `Jane.Hillston@ed.ac.uk`
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK

Contents

1 Introduction

Typical self-organising collective systems consist of a large number of interacting objects that coordinate their activities in a decentralised and often implicit way. These features play an important role in making such systems robust and able to flexibly and autonomously adapt to changing circumstances while maintaining an acceptable level of operation and optimisation. This is a desirable feature of systems in natural environments and can be found in many instances, where perhaps the most well-known and most studied examples are the behaviour in ant and bee colonies [25, 59]. These desirable features of self-organisation, among others, have recently attracted the interest of researchers in the field of engineering (see e.g. [12] and more recently e.g. [28, 49, 25, 62, 58]). There are numerous examples of man-made systems, existing or planned, that incorporate some forms of self-organisation, for the better or the worse. A familiar example is perhaps car traffic, where drivers tend to keep sufficient distance from other drivers and obstacles and follow a limited set of traffic rules. However, at the same time drivers cannot easily avoid the emergence of traffic jam. Other examples concern the development of a fully decentralised smart electric grid [27, 56], with many local producers and consumers, or smart transportation systems, that provide real-time information to travellers and adapt to changing requests. Examples are public transportation [14] and shared bike systems [21, 30]. Often the humans involved in such systems are not simply 'end-users' but they are intrinsic and autonomous interacting elements of the system, constantly receiving feedback and providing input while e.g. deciding where to take a shared bike and deciding the most convenient destination to leave it, which may depend on the time of the day, the personal needs, potential incentives and the actual situation of parking lots.

From a conceptual point of view, the notion of self-organisation is intuitively clear, but at the same time it is difficult to provide a precise mathematical definition that covers all phenomena that could arise from it. This is particularly so because instances of what could be seen as self-organisation can be found in many different scientific disciplines, ranging from physics, to biology, sociology and economics to mention a few. In the context of pattern formation in biological systems, Camazine et al. [13] provide the following definition: "Self-organisation is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system components are executed using only local information, without reference to the global pattern". Although much more articulated working definitions can be found in the literature nowadays (see e.g. Frei and Di Marzo Serugendo [28]), the above definition nevertheless covers a number of typical aspects of the examples of self-organisation that we address in this paper. First, the system of concern is composed of many interacting, and we would add autonomous, components. Second, the components use only local information on the current state of the system, without reference to the global pattern. Third, this global pattern is assumed to manifest itself over time as a result of the interactions among the components and the specific, a priori rules of behaviour that these components obey to. Although the above definition has its origin in the field of biological systems, the same ingredients can be found in most engineered self-organised systems.

The pervasive nature of engineered self-organised systems, along with the increasing critical dependence of people on their continuous reliable operation, means that it is extremely important to develop a priori analysis techniques of their design and to investigate all aspects of their behaviour, including quantitative and emergent aspects, before they are put into operation. The development of such analysis techniques is a very active field of research and covers very diverse approaches, among which various simulation techniques, deterministic approximation, stability analysis, to mention a few. In this article we propose and explore a new analysis technique that combines model-checking—a successful verification technique from the area of temporal logics and process algebra—and mean-field approximation techniques.

Process algebras, and in particular their stochastic extensions, have been designed and extensively and successfully used to model and specify behavioural aspects of concurrent systems, including non-functional properties [39, 11]. Examples of their application can be found in diverse areas ranging from communication protocols [26] to control theory [57]. A process algebraic design framework is complementary to temporal logics model checking techniques. Such techniques, and their stochastic extensions (see for instance [3, 2, 44]), have also proved to be extremely useful in the design of concurrent and distributed systems many of which involving decentralised strategies. Recent extensions of stochastic process algebras are suitable to define large scale reactive systems composed of interacting classes of independent autonomously behaving components [40, 61, 16]. Examples show that the latter are sufficiently expressive to model individual-based behaviour that leads to interesting emergent behaviour at the global level (see for instance [52, 51, 7, 50]). These extensions exploit mean field or fluid approximation techniques [19, 10]. Such techniques work by approximating the discrete state space of stochastic or probabilistic population models by a continuous one while abstracting from the identity of the individual objects. For example, if an individual object can evolve between three different discrete states, and we have a model with many individuals, then a typical (approximate) continuous state consists of a triple in which each element records the fraction of the population of individuals that are in a particular local state, respectively. When the population is very large, the average stochastic dynamics of the system behaviour can be approximated in continuous time or in discrete time. In the first case the behaviour can be approximated by a (deterministic) solution of a set of *differential* equations, and in the latter by the (deterministic) solution of a set of *difference* equations. In both cases the asymptotic correctness of the approximations is guaranteed by limit theorems, see e.g. [19] for approximations in continuous time, and [47] for correctness of the approach in discrete time (the interested reader is referred to [9] for an introduction to the field of deterministic approximation of collective system behaviour). Although mean field approximation provides a computationally efficient way to obtain an impression of the average dynamics of the full system over time, the behaviour of individuals in the system is kind of lost. Such individual behaviour, that evolves in the context of the overall system, is however often of interest. For example, in the case of an epidemics outbreak, one may be interested in knowing what is the probability that a certain individual gets infected within a certain amount of time in the presence (or absence) of measures to prevent spreading of a virus. Clearly, the behaviour of a single individual is affected by the behaviour and status of the other individuals in the system. Interestingly, it has been shown that the probabilistic behaviour of an individual in the context of the overall system can indeed be approximated by combining a model of the individual with an approximation of the average behaviour of the system. In some sense, the individual is affected by the rest of the system only through the 'average' behaviour of the latter. This combined approach is also known as *fast simulation* and has been proven to be asymptotically correct. Fast simulation has been developed both in a continuous time setting (see for example [19]) and in a discrete time setting (see for example [47]).

In particular, in the discrete time setting, the average system dynamics can be computed by a deterministic iterative procedure [47]. It is this feature that we extensively exploit in the development of an on-the-fly probabilistic mean field model-checker for the analysis of properties of individual behaviour in the context of large self-organised, collective systems that is the main contribution of the present paper.

Model checking has been widely recognised as a powerful approach to the automatic verification of concurrent and distributed systems. It consists of an efficient procedure that, given an abstract model $\mathcal{M}$ of the system, decides whether $\mathcal{M}$ satisfies a logical formula Φ, typically drawn from a temporal logic. Traditionally, model checking approaches are divided into two broad categories: *global* approaches that determine the set of *all* states in $\mathcal{M}$ that satisfy Φ, and *local* approaches that, given a state s in $\mathcal{M}$, determine whether s satisfies Φ [18, 6].

Global symbolic model checking algorithms are popular because of their computational efficiency and can be found in many model checkers, both in a qualitative (see e.g. [17]) and in a stochastic setting (see e.g. [2, 44]). The set of states that satisfy a formula is constructed recursively in a *bottom-up* fashion following the syntactic structure of the formula. Depending on the particular formula to verify, usually the underlying model can be reduced to fewer states before the algorithm is applied. Moreover, as is shown e.g. in [2] for stochastic model checking, the global model checking algorithm can be reduced to combinations of existing well-known and optimised algorithms for Continuous Time Markov Chains (CTMCs) such as transient analysis [2]. Despite their success, the scalability of model-checking algorithms have always been a concern due to the potential combinatorial explosion of the state space that needs to be searched.

Local model checking algorithms have been proposed to mitigate the state space explosion problem using a so called 'on-the-fly' approach (see e.g. [18, 6, 41, 32]). On-the-fly algorithms are following a *top-down* approach that does not require global knowledge of the complete state space. For each state that is encountered, starting from a given state, the outgoing transitions are followed to adjacent states, constructing step by step local knowledge of the state space until it is possible to decide whether the given state satisfies the formula.

The main contribution of this paper is the design, implementation and application of a novel model-checking procedure, based on an original combination of local, *on-the-fly* model-checking techniques and mean field approximation in discrete time [47]. The procedure can be used to verify bounded PCTL (Probabilistic Computation Tree Logic) [36] properties of *selected individuals* in the context of systems consisting of a large number of similar, but independent, interacting objects; a limited form of *global* system properties can be treated as well. The procedure is scalable in the sense that it is insensitive to the size of the population the system consists of.

The asymptotic correctness of the on-the-fly model-checking procedure is proven and a prototype implementation of the model-checker, FlyFast, is applied to a selection of simple and more elaborate case studies from the field of computer epidemics (also discussed in [9]), public transportation, in particular bike sharing [21], and population dynamics. To the best of our knowledge, this is the first proposal and implementation of an *on-the-fly mean field* model-checker for *discrete time, probabilistic, time-synchronous* models; fast simulation has has been recently applied in the work by Hillston and Bortolussi [8], but in the context of continuous time, stochastic, interleaving models, as we will discuss in more detail in Sect. 2.

Following the approach in [47] we consider a model for interacting objects, where the evolution of each object is given by a finite state discrete time Markov chain[1]. The transition matrix of each object may depend on the distribution of states of all objects in the system. Each object can be in one of its local states at any point in time and all objects proceed in discrete time and in a clock-synchronous fashion. When the number of objects is large, the overall behaviour of the system in terms of its 'occupancy measure', i.e. the fraction of objects that are in a particular local state at a particular time, can be approximated by the (deterministic) solution of a *difference equation* which is called the 'mean field'[2]. This convergence result has been extended in [47] to obtain a 'fast' way to stochastically simulate the evolution of a selected, limited number of specific objects in the context of the overall behaviour of the population.

We show that the deterministic iterative procedure of [47], to compute the average overall behaviour of the system and that of individual selected objects in the context of the overall system, combines well with an *on-the-fly* probabilistic model-checking procedure for the verification of bounded PCTL formulas addressing properties of selected objects of interest, and a limited form of global system properties[3]. An on-the-fly recursive approach also provides a natural way to address nested path formulas and time-varying truth values of such formulas. The model-checking algorithm presented in this paper is parametric w.r.t. the semantic interpretation of the model specification language. In particular we present two different instantiations of the algorithm; one based on the standard,

[1]These models are also known as SIO-models (System of Independent Objects) [9]. These are time-synchronous models in which each object performs a probabilistic step in each discrete time unit, possibly returning to the same state. This is a class of models that is frequently encountered in various research disciplines ranging from telecommunication to computational biology. The objects interact in an indirect way, via the global state of the overall system.

[2]The term 'mean field' has its origin in statistical physics and is sometimes used with slightly different meanings in the literature. Here we intend the meaning as defined in [47], i.e. a deterministic approximation of the occupancy measure of a probabilistic population model.

[3]Note that the transition probabilities of these selected objects at time t may depend on the occupancy measure of the system at t and therefore also the truth-values of the formulas may vary in time.

exact probabilistic semantics of a simple probabilistic population description language, and the other one on the *mean-field approximation in discrete time* of such a semantics. The latter is the main contribution of the present paper. The considered PCTL formulas can be extended along the lines proposed in [42, 43] with properties that address the overall status of the system. We show some simple instances of that as an example.

A preliminary version of this work has appeared in [45]. The current version includes detailed correctness proofs and further case studies that illustrate the potential and limitations of the approach for the verification of self-organised systems. Furthermore, a discussion of open issues and future extensions of the approach is provided.

The paper is organised as follows: Section 2 discusses related work, Section 3 provides preliminary definitions and introduces time bounded PCTL and on-the-fly probabilistic model-checking. Section 4 introduces a simple population description language and a running example from the field of computer epidemics. Section 5 introduces fast mean-field probabilistic model-checking and provides related correctness theorems. Section 6 shows the application of the FlyFast model-checker on several case studies from the field of self-organised systems. Finally, Section 7 provides a discussion on possible extensions, limitations and open issues followed by the conclusions in Section 8.

2 Related Work

For qualitative model checking, local model-checking algorithms have been shown to have the same worst-case complexity as the best existing global procedures for the above mentioned logics. However, in practice, they have better performance when only a subset of the system states need to be analysed to determine whether a system satisfies a formula. Furthermore, local model-checking may still provide some results in case of systems with a very large or even infinite state space where global model checking approaches would be impossible to use. In the context of stochastic model checking several on-the-fly approaches have been proposed, among which [23] and [35]. The former is a probabilistic model checker for bounded PCTL formulas. The latter uses an on-the-fly approach to detect a maximal relevant search depth in an infinite state space and then uses a *global* model-checking approach to verify bounded CSL [1, 2] formulas in a continuous time setting on the selected subset of states. An on-the-fly approach by itself however, does not solve the challenging scalability problems that arise in truly large parallel systems, such as collective adaptive systems, e.g., gossip protocols [15], self-organised collective decision making [55], computer epidemics [10] and foreseen smart urban transportation systems and decentralised control strategies for smart grids.

To address this type of scalability challenges in probabilistic model-checking, recently, several approaches have been proposed. In [38, 34] approximate probabilistic model-checking is introduced. This is a form of statistical model-checking that consists in the generation of random executions of an *a priori* established maximal length. On each execution the property of interest is checked and statistics are performed over the outcomes. The number of executions required for a reliable result depends on the maximal error-margin of interest. The approach relies on the analysis of individual execution traces rather than a full state space exploration and is therefore memory-efficient. However, the number of execution traces that may be required to reach a desired accuracy may be large and therefore time-consuming. The approach works for general models, i.e., not necessarily populations of similar objects, but is not independent of the number of objects involved.

To analyse properties of large scale mobile communication networks mean field approximations in discrete time have also been used e.g., in Bakshi et al. [4]. In that work an automatised method is proposed and applied to the analysis of dynamic gossip networks. A general convergence result to a deterministic difference equation is used, similar to that in [47], but not its extension to analyse individual behaviour in the context of a large population, nor its exploitation in model-checking algorithms.

In Chaintreau et al. [15], mean field convergence in *continuous time* is used to analyse the distribution of the age of information that objects possess when using a mix of gossip and broadcast for information distribution in situations where objects are not homogeneously distributed in space. An overview of mean field interaction models for computer and communication systems by Benaïm et al. can be found in [5].

Preliminary ideas on the exploitation of mean field convergence in *continuous time* for model-checking mean field models, and in particular for an extension of the logic CSL, were informally sketched in a presentation at QAPL 2012 [42], but no model-checking algorithms were presented. Follow-up work on the above mentioned approach can be found in [43] which relies on earlier results on fluid model checking by Bortolussi and Hillston [8]. In the latter a *global CSL* model-checking procedure is proposed for the verification of properties of a selection of individuals in a population, which relays on fast simulation results. This work is perhaps closest related to our work; however their procedure exploits mean field convergence and fast simulation [19, 31] in a *continuous* time setting rather than in a discrete time setting and is based on an *interleaving* model of computation, rather than a clock-synchronous one; furthermore, a *global* model-checking approach, rather than an on-the-fly approach is followed. Consequently, the underlying model-checking algorithms and related correctness proofs are fundamentally different from those presented in this paper.

The modelling language used in [8] is PEPA. Earlier work by Stefanek et al. [60, 37] on the use of mean field convergence in *continuous time* for grouped PEPA has investigated the quality of the convergence results when the related differential equations are derived directly from the process algebraic model. Also the use of higher order moments in models, such as variance and skewness, has been explored which provide more information on the behaviour of a large model than the average behaviour alone.

3 Time bounded PCTL and On-the-fly Model-Checking

In this section we recall the definition of the *time bounded* fragment of the Probabilistic Computation Tree Logic [36] (PCTL)[4] and we present an on-the-fly model-checking algorithm. The algorithm is parametric in the sense that it can be used for different languages and semantic interpretations. In this paper we use two instantiations of the algorithm; one is on a DTMC semantics of a simple language of object populations (Sect. 4) and the other is on a mean-field approximation semantics of the same language, for "fast model-checking" (Sect.5). For the sake of readability, we present only a schema of the algorithm for time *bounded* PCTL, that is the same as that proposed in [23]. The interested reader is referred to [46] where a novel algorithm is defined and implemented for *the full* logic.

3.1 Time bounded PCTL

Given a set $\mathscr{P}$ of atomic propositions, the syntax of PCTL is defined below, where $a \in \mathscr{P}$, $k \geq 0$ and $\bowtie \in \{\geq, >, \leq, <\}$:

$$\Phi ::= a \mid \neg\,\Phi \mid \Phi \,\vee\, \Phi \mid \mathcal{P}_{\bowtie p}(\varphi) \qquad \text{where } \varphi ::= \mathcal{X}\,\Phi \mid \Phi\,\mathcal{U}^{\leq k}\,\Phi.$$

PCTL formulas are interpreted over *state labelled* DTMCs. A state labelled DTMC is a pair $\langle \mathcal{M}, \ell \rangle$ where $\mathcal{M}$ is a DTMC with state set $\mathcal{S}$ and $\ell : \mathcal{S} \to 2^{\mathscr{P}}$ associates each state with a set of atomic propositions; for each state $s \in \mathcal{S}$, $\ell(s)$ is the set of atomic propositions true in s. In the following, we assume $\mathbf{P}$ be the one step probability matrix for $\mathcal{M}$; we abbreviate $\langle \mathcal{M}, \ell \rangle$ with $\mathcal{M}$, when no confusion can arise. A path σ over $\mathcal{M}$ is a non-empty sequence of states $s_0, s_1, \cdots$ where $\mathbf{P}_{s_i, s_{i+1}} > 0$ for all $i \geq 0$. We let $Paths_{\mathcal{M}}(s)$ denote the set of all infinite paths over $\mathcal{M}$ starting from state s. By $\sigma[i]$ we denote the element s_i of path σ. Finally, in the sequel we will consider DTMCs equipped with an initial state s_0, i.e. the probability mass is initially all in s_0. For any such a DTMC $\mathcal{M}$, and for all $t \in \mathbb{N}$ we let the set $L_{\mathcal{M}}(t) = \{\sigma[t] \mid \sigma \in Paths_{\mathcal{M}}(s_0)\}$.

We define the satisfaction relation on $\mathcal{M}$ and the logic in Table 1.

[4]For notational simplicity we call the fragment PCTL as well.

$s \models_{\mathcal{M}} a$	iff	$a \in \ell(s)$
$s \models_{\mathcal{M}} \neg\Phi$	iff	not $s \models_{\mathcal{M}} \Phi$
$s \models_{\mathcal{M}} \Phi_1 \vee \Phi_2$	iff	$s \models_{\mathcal{M}} \Phi_1$ or $s \models_{\mathcal{M}} \Phi_2$
$s \models_{\mathcal{M}} \mathcal{P}_{\bowtie p}(\varphi)$	iff	$\mathbb{P}\{\sigma \in Paths_{\mathcal{M}}(s) \mid \sigma \models_{\mathcal{M}} \varphi\} \bowtie p$
$\sigma \models_{\mathcal{M}} \mathcal{X}\ \Phi$	iff	$\sigma[1] \models_{\mathcal{M}} \Phi$
$\sigma \models_{\mathcal{M}} \Phi_1 \mathcal{U}^{\leq k} \Phi_2$	iff	$\exists\, 0 \leq h \leq k$ *s.t.* $\sigma[h] \models_{\mathcal{M}} \Phi_2 \ \wedge\ \forall\, 0 \leq i < h\ .\ \sigma[i] \models_{\mathcal{M}} \Phi_1$

Table 1: Satisfaction relation for Time Bounded PCTL.

```
Check( $s$ : proc, $\Phi$ : formula)=
match $\Phi$
with
|$a \rightarrow$ (lab_eval $s$ $a$)
|$\neg\Phi_1 \rightarrow \neg$Check($s, \Phi_1$)
|$\Phi_1 \vee \Phi_2 \rightarrow$ Check($s, \phi_1$) $\vee$ Check($s, \Phi_2$)
|$\mathcal{P}_{\langle\text{relop}\rangle p}(\varphi) \rightarrow$ CheckPath($s, \varphi$)$\langle$relop$\rangle p$
```

Table 2: Function Check

3.2 On-the-fly PCTL Model-Checking Algorithm

In this section we introduce a local on-the-fly model-checking algorithm for time-bounded PCTL formulas. The basic idea of an on-the-fly algorithm is simple: while the state space is generated in a stepwise fashion from a term s of the language, the algorithm considers only the relevant prefixes of the paths while they are generated. For each of them it updates the information about the satisfaction of the formula that is checked. In this way, only that part of the state space is generated that can provide information on the satisfaction of the formula and irrelevant parts are not taken into consideration.

In the case of probabilistic process population languages, for large populations, a mean-field approximated semantics can be defined. In Sect. 5 we show how a drastic reduction of the state space can be obtained, by using the same algorithm on such semantic models. We call such a combined use of on-the-fly model-checking and mean-field semantics "Fast model-checking" after "Fast simulation", introduced in [47].

The algorithm abstracts from any specific language and different semantic interpretations of a language. We only assume an abstract interpreter function that, given a generic process term, returns a probability distribution over the set of terms. Below, we let proc be the (generic) type of *probabilistic process terms* while we let formula and path_formula be the types of *state-* and *path-* PCTL formulas. Finally, we use lab to denote the type of *atomic propositions*.

The abstract interpreter can be modelled by means of two functions: next and lab_eval. Function next associates a list of pairs (proc, float) to each element of type proc. The list of pairs gives the terms, i.e. states, that can be reached in one step from the given state and their one-step transition probability. We require that for each s of type proc it holds that $0 < p' \leq 1$, for all $(s', p') \in \mathsf{next}(s)$ and $\sum_{(s',p')\in\mathsf{next}(s)} p' = 1$. Function lab_eval returns for each element of type proc a function associating a bool to each atomic proposition a in lab. Each instantiation of the algorithm consists in the appropriate definition of next and lab_eval, depending on the language at hand and its semantics.

The local model-checking algorithm is defined as a function, Check, shown in Table 2. On atomic state-formulas, the function returns the value of lab_eval; when given a non-atomic state-formula, Check calls itself recursively on sub-formulas, in case they are state-formulas, whereas it calls function CheckPath, in case the sub-formula is a path-formula. In both cases the result is a Boolean value that indicates whether the state satisfies the formula.

Function CheckPath, shown in Table 3, takes a state $s \in$ proc and a PCTL path-formula $\varphi \in$

```
CheckPath( $s$ : proc , $\varphi$ : path_formula )=
match $\varphi$ with
| $\mathcal{X}\,\Phi \rightarrow$ let $p$ = 0.0 and $lst$ = next($s$) in
      for $(s',p') \in lst$ do if Check($s',\Phi$) then p $\leftarrow p+p'$
      done;
      $p$
| $\Phi_1\,\mathcal{U}^{\leq k}\,\Phi_2 \rightarrow$ if Check($s,\Phi_2$) then 1.0
         else if Check($s,\neg\Phi_1$) then 0.0
         else if $k>0$ then
                begin
                    let $p$ = 0.0 and $lst$ = next($s$) in
                    for $(s',p') \in lst$ do
                       $p \leftarrow p+p' *$ CheckPath($s',\Phi_1\,\mathcal{U}^{\leq k-1}\,\Phi_2$)
                    done;
                    $p$
                end
             else 0.0
```

Table 3: Function CheckPath

path_formula as input. As a result, it produces the probability measure of the set of paths, starting in state s, which satisfy path-formula φ. Following the definition of the formal semantics of PCTL, two different cases can be distinguished. If φ has the form $\mathcal{X}\,\Phi$ then the result is the sum of the probabilities of the transitions from s to those next states s' that satisfy Φ. To verify the latter, function Check is recursively invoked on such states. If φ has the form $\Phi_1\,\mathcal{U}^{\leq k}\,\Phi_2$ then we first check if s satisfies Φ_2, then 1 is returned, since φ is trivially satisfied. If s does not satisfy Φ_1 then 0 is returned, since φ is trivially violated. For the remaining case we need to recursively invoke CheckPath for the states reachable in one step from s, i.e. the states in the set $\{s' | \exists p' : (s',p') \in \text{next}(s)\}$. Note that these invocations of CheckPath are made on $\varphi' = \Phi_1\,\mathcal{U}^{\leq k-1}\,\Phi_2$ if $k>0$. If $k \leq 0$ then the formula is trivially not satisfied by s and the value 0 is returned.

Let s be a term of a probabilistic process language and $\mathcal{M}$ the complete discrete time stochastic process associated with s by the formal semantics of the language. The following theorem is easily proved by induction on Φ [46].

Theorem 1. *$s \models_{\mathcal{M}} \Phi$ if and only if Check$(s,\Phi) =$ true.* •

4 Modelling language

In this section we define a simple population description language. The language is essentially a textual version of the graphical notation used in [47]. A *system* is defined as a population of N identical interacting processes or objects[5]. At any point in time, each object can be in any of its finitely many states and the evolution of the system proceeds in a *clock-synchronous* fashion: at each clock tick each member of the population must either execute one of the transitions that are enabled in its current state, or remain in such a state.[6]

Syntax. Let $\mathcal{A}$ be a denumerable non-empty set of *actions*, ranged over by $a, a', a_1, \ldots$ and $\mathcal{S}$ be a denumerable non-empty set of *state constants*, ranged over by $C, C', C_1, \ldots$ An *object specification* Δ is a set $\{D_i\}_{i \in I}$, for finite index set I, where each *state definition* D_i has the form $C_i := \sum_{j \in J_i} a_{ij}.C_{ij}$, with J_i a finite index set, states $C_i, C_{ij} \in \mathcal{S}$, and $a_{ij} \in \mathcal{A}$, for $i \in I$ and $j \in J_i$. Intuitively, the notation

[5]In [47] *object* is used instead of *process*. We consider the two terms synonyms here.

[6]For the purpose of the present paper, language expressivity is not a main concern.

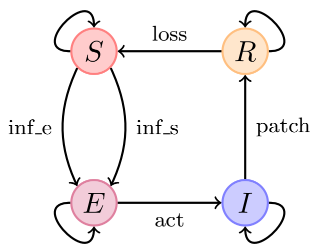


Figure 1: Epidemic model object.

$\sum_{j\in J_i} a_{ij}.C_{ij}$ is to be intended as the n-ary extension of the standard process algebraic binary non-deterministic choice operator. We require that $a_{ij} \neq a_{ij'}$, for $j \neq j'$ and that for each state constant C_{ij} occurring in the r.h.s. of a state definition D_i of Δ there is a *unique* $k \in I$ such that C_{ij} is the l.h.s. of D_k.

Example 1 (An epidemic model [9]). *We consider a network of computers that can be infected by a worm. Each node in the network can acquire infection from two sources, i.e. by the activity of a worm of an infected node (`inf_sus`) or by an external source (`inf_ext`). Once a computer is infected, the worm remains latent for a while, and then activates (`activate`).*

When the worm is active, it tries to propagate over the network by sending messages to other nodes. After some time, an infected computer can be patched (`patch`), so that the infection is recovered. New versions of the worm can appear; for this reason, recovered computers can become susceptible to infection again, after a while (`loss`). The object specification of the epidemic model is the following (see Fig. 1):

```
S := inf_ext.E + inf_sus.E
E := activate.I
I := patch.R
R := loss.S
```

The set of all actions occurring in object specification Δ is denoted by $\mathcal{A}_\Delta$. Similarly, the set of states is denoted by $\mathcal{S}_\Delta$, ranged over by $c, c', c_1 \cdots$. In Example 1, we have

$$\mathcal{A}_{\texttt{EM}} = \{\texttt{inf\_ext}, \texttt{inf\_sus}, \texttt{activate}, \texttt{patch}, \texttt{loss}\}$$

and $\mathcal{S}_{\texttt{EM}} = \{\texttt{S}, \texttt{E}, \texttt{I}, \texttt{R}\}$. A system is assumed composed of N interacting instances of an object. Interaction among objects is modelled probabilistically, as described below. Each action in $\mathcal{A}_\Delta$ is assigned a probability value, that may depend on the global state of the system. This is achieved by means of a *probability function definition*, that takes the following form: $a :: E$, where $a \in \mathcal{A}_\Delta$ and E is an expression, i.e. an element of Exp, defined according to the following grammar:

$$E ::= v \mid \mathsf{frc}\, C \mid \langle\text{uop}\rangle\, E \mid E\,\langle\text{bop}\rangle\, E \mid (E)$$

where $v \in [0, 1]$ and for each state C, $\mathsf{frc}\, C$ denotes the fraction of objects, over the total number of objects N, in the system, that are currently in state C. Operators $\langle\text{uop}\rangle$ and $\langle\text{bop}\rangle$ are standard arithmetic unary and binary operators.

Example 2 (Probability function definitions). *For the* epidemic model *of Example 1 we assign the following probability function definitions:*

```
inf_ext :: $\alpha_e$;
inf_sus :: $\alpha_i * (\mathsf{frc}\, I)$;
activate :: $\alpha_a$;
patch :: $\alpha_r$;
loss :: $\alpha_s$;
```

where α_e, α_i, α_a, α_r and α_s are model parameters in $[0,1]$, with $\alpha_e + \alpha_i \leq 1$.

A *system specification* is a triple $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$ where Δ is an object specification, A is a set of probability function definitions containing exactly one definition for each $a \in \mathcal{A}_\Delta$, and $\mathbf{C}_0 = \langle c_{0_1}, \ldots, c_{0_N} \rangle$ is the *initial system state*, with $c_{0_n} \in \mathcal{S}_\Delta$, for $n = 1 \ldots N$; we say that N is the *population size*[7]; in the sequel, we will omit the explicit indication of the size N in $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$, and elements thereof or related functions, writing simply $\langle \Delta, A, \mathbf{C}_0 \rangle$, when this cannot cause confusion.

Semantics. Let $\langle \Delta, A, \mathbf{C}_0 \rangle$ be a system specification. We associate with Δ the Labelled Transition System (LTS) $\langle \mathcal{S}_\Delta, \mathcal{A}_\Delta, \rightarrowtail \rangle$, where $\mathcal{S}_\Delta$ and $\mathcal{A}_\Delta$ are the states and labels of the LTS, respectively, and the transition relation $\rightarrowtail \subseteq \mathcal{S}_\Delta \times \mathcal{A}_\Delta \times \mathcal{S}_\Delta$ is the smallest relation induced by rule (1).

$$\frac{C := \sum_{j \in J} a_j.C_j \quad k \in J}{C \overset{a_k}{\rightarrowtail} C_k} \tag{1}$$

In the following we let $\mathcal{U}^S = \{\mathbf{m} \in [0,1]^S | \sum_{i=1}^S \mathbf{m}_{[i]} = 1\}$ be the unit simplex of dimension S; furthermore, we let $c, c', C, C' \ldots$ range over $\mathcal{S}_\Delta$ and for generic vector $\mathbf{v} = \langle v_1, \ldots, v_r \rangle$ we let $\mathbf{v}_{[j]}$ denote the j-th component v_j of $\mathbf{v}$, for $j = 1, \ldots, r$. A (system) *global state* is a tuple $\mathbf{C}^{(N)} \in \mathcal{S}_\Delta^N$. W.l.g., we assume that $\mathcal{S}_\Delta = \{C_1, \ldots, C_S\}$ and that a total order is defined on state constants $C_1, \ldots, C_S$ so that we can unambiguously associate each component of a vector $\mathbf{m} = \langle m_1, \ldots, m_S \rangle \in \mathcal{U}^S$ with a distinct element of $\{C_1, \ldots, C_S\}$. With each global state $\mathbf{C}^{(N)}$ an *occupancy measure* vector $\mathbf{M}^{(N)}(\mathbf{C}^{(N)}) \in \mathcal{U}^S$ is associated where $\mathbf{M}^{(N)}(\mathbf{C}^{(N)}) = \langle M_1^{(N)}, \ldots, M_S^{(N)} \rangle$ with

$$M_i^{(N)} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{1}_{\{\mathbf{C}_{[n]}^{(N)} = C_i\}} \tag{2}$$

for $i = 1, \ldots, S$, and the value of $\mathbf{1}_{\{\alpha=\beta\}}$ is 1, if $\alpha = \beta$, and 0 otherwise.

A probability function definition $a :: E$ associates a real value to action a by evaluating E in the current global state, via the interpretation function $\mathscr{E}$. In practice the occupancy measure representation of the state is used in $\mathscr{E}$. The expressions interpretation function $\mathscr{E} : \mathrm{Exp} \to \mathcal{U}^S \to \mathbb{R}$ is defined as usual:

$\mathscr{E}[\![v]\!]_\mathbf{m} = v$
$\mathscr{E}[\![\mathsf{frc}\, C_i]\!]_\mathbf{m} = \mathbf{m}_{[i]}$
$\mathscr{E}[\![\langle \mathrm{uop} \rangle\, E]\!]_\mathbf{m} = \langle \mathrm{uop} \rangle\, (\mathscr{E}[\![E]\!]_\mathbf{m})$
$\mathscr{E}[\![E_1 \langle \mathrm{bop} \rangle E_2]\!]_\mathbf{m} = (\mathscr{E}[\![E_1]\!]_\mathbf{m}) \langle \mathrm{bop} \rangle (\mathscr{E}[\![E_2]\!]_\mathbf{m})$
$\mathscr{E}[\![(E)]\!]_\mathbf{m} = (\mathscr{E}[\![(E)]\!]_\mathbf{m})$

The set A of probability function definitions characterises a function π with type $\mathcal{U}^S \to \mathcal{A}_\Delta \to \mathbb{R}$ as follows: for each $a :: E$ in A, we have $\pi(\mathbf{m}, a) = \mathscr{E}[\![E]\!]_\mathbf{m}$.

[7]Appropriate syntactical shorthands can be introduced for describing the initial state, e.g. $\langle \mathtt{S[2000]}, \mathtt{E[100]}, \mathtt{I[200]}, \mathtt{R[0]} \rangle$ for 2000 objects initially in state S etc.

For a system specification of size N, we define the *object transition matrix* as follows: $\mathbf{K}^{(N)} : \mathcal{U}^S \times \mathcal{S}_\Delta \times \mathcal{S}_\Delta \to \mathbb{R}$, with

$$\mathbf{K}^{(N)}(\mathbf{m})_{c,c'} = \begin{cases} \sum_{a:c \overset{a}{\rightarrowtail} c'} \pi(\mathbf{m}, a), & \text{if } c \neq c', \\ 1 - \sum_{a \in I(c)} \pi(\mathbf{m}, a), & \text{if } c = c'. \end{cases} \tag{3}$$

where $I(c) = \{a \in \mathcal{A}_\Delta | \exists c' \in \mathcal{S}_\Delta : c \overset{a}{\rightarrowtail} c' \neq c\}$. We say that a state $c \in \mathcal{S}_\Delta$ is *probabilistic* in $\mathbf{m}$ if $0 \leq \sum_{a \in I^*(c)} \pi(\mathbf{m}, a) \leq 1$ where set $I^*(c)$ is defined as follows: $I^*(c) = I(c) \cup \{a \in \mathcal{A}_\Delta | c \overset{a}{\rightarrowtail} c\}$. Note that whenever all states in $\mathcal{S}_\Delta$ are probabilistic in $\mathbf{m}$, matrix $\mathbf{K}^{(N)}(\mathbf{m})$ is a one step transition probability matrix. We define the (system) *global state transition matrix* $\mathbf{S}^{(N)} : \mathcal{U}^S \times \mathcal{S}_\Delta^N \times \mathcal{S}_\Delta^N \to \mathbb{R}$, as

$$\mathbf{S}^{(N)}(\mathbf{m})_{\mathbf{C},\mathbf{C}'} = \Pi_{n=1}^N \mathbf{K}^{(N)}(\mathbf{m})_{\mathbf{C}_{[n]},\mathbf{C}'_{[n]}}.$$

Note that whenever all states in $\mathcal{S}_\Delta$ are probabilistic in $\mathbf{m}$, matrix $\mathbf{S}^N(\mathbf{m})$ is a one step transition probability matrix modelling a possible single step of the system as result of the parallel execution of a single step of each of the N instances of the object. In this case, the $S^N \times S^N$ matrix $\mathbf{P}^{(N)}$ with

$$\mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \mathbf{S}^{(N)}(\mathbf{M}^{(N)}(\mathbf{C}))_{\mathbf{C},\mathbf{C}'} \tag{4}$$

is the one-step transition matrix of a (finite state) DTMC, namely the DTMC of the system composed on N objects specified by Δ. In this case, we let $\mathbf{X}^{(N)}(t)$ denote the Markov process with transition probability matrix $\mathbf{P}^{(N)}$ as above and $\mathbf{X}^{(N)}(0) = \mathbf{C}_0^{(N)}$, i.e. with initial probability distribution $\boldsymbol{\delta}_{\mathbf{C}_0^{(N)}}$, where $\mathbf{C}_0^{(N)}$is the initial system state and $\boldsymbol{\delta}_{\mathbf{C}_0^{(N)}}$ is the Dirac distribution with the total mass on $\mathbf{C}_0^{(N)}$. With a little bit of notational overloading, we define the 'occupancy measure DTMC' as $\mathbf{M}^{(N)}(t) = \mathbf{M}^{(N)}(\mathbf{X}^{(N)}(t))$; for $\mathbf{m} = \mathbf{M}^{(N)}(\mathbf{C})$, for some state $\mathbf{C}$ of DTMC $\mathbf{X}(t)$, we have:

$$\mathbb{P}\{\mathbf{M}^{(N)}(t+1) = \mathbf{m}' \mid \mathbf{M}^{(N)}(t) = \mathbf{m}\} = \sum_{\mathbf{C}':\mathbf{M}^{(N)}(\mathbf{C}')=\mathbf{m}'} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \tag{5}$$

Note that the above definition is a good definition; in fact, if $\mathbf{M}^{(N)}(\mathbf{C}) = \mathbf{M}^{(N)}(\mathbf{C}'')$, then $\mathbf{C}$ and $\mathbf{C}''$ are just two permutations of the same local states. This implies that for all $\mathbf{C}'$ we have $\mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \mathbf{P}^{(N)}_{\mathbf{C}'',\mathbf{C}'}$.

PCTL local Model-checking. For the purpose of expressing system properties in PCTL, we partition the set of atomic propositions $\mathscr{P}$ into sets $\mathscr{P}_1$and $\mathscr{P}_g$. Given system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$, we extend it with a *state labelling function definition* that associates each state $c \in \mathcal{S}_\Delta$ with a (possibly empty) finite set $\ell_1(c)$ of propositions from $\mathscr{P}_1$. We extend ℓ_1 to global states with $\ell_1(\langle c_1, \ldots, c_N \rangle) = \ell_1(c_1)$; this way, we can express *local* properties of the first object in the system, in the *context* of the complete population[8].

In order to express also (a limited class of) *global* properties of the population, we use set $\mathscr{P}_g$. The system specification is further enriched by associating labels $a \in \mathscr{P}_g$ with expressions bexp in the class BExp of restricted boolean expressions. We assume a sublanguage of function specifications be given[9] and for function symbol F, $\mathscr{E}[\![F]\!]_{\mathbf{m}} : [0,1]^q \mapsto \mathbb{R}$ continuous in $[0,1]^q$, with $\mathscr{E}[\![F]\!]_{\mathbf{m}} = \mathscr{E}[\![F]\!]_{\mathbf{m}'}$ for all $\mathbf{m}, \mathbf{m}' \in \mathcal{U}^S$; then BExp is the set of expressions of the form $F(E_1, \ldots, E_q) \langle \text{relop} \rangle\, r$, where each E_j is of the form $\mathsf{frc}\, C$, $\langle \text{relop} \rangle \in \{>, <\}$, $r \in \mathbb{R}$ and $\mathscr{E}[\![F(E_1, \ldots, E_q)]\!]_{\mathbf{m}} = \mathscr{E}[\![F]\!]_{\mathbf{m}}(\mathscr{E}[\![E_1]\!]_{\mathbf{m}}, \ldots, \mathscr{E}[\![E_q]\!]_{\mathbf{m}})$.

[8]Of course, the choice of the *first* object is purely conventional. Furthermore, all the results which in the present paper are stated w.r.t. the *first* object of a system, are easily extened to finite subsets of objects in the system. For the sake of notation, in the rest of the paper, we stick to the *first object* convention.

[9]The specific features of the sublanguage are not relevant for the purposes of the present paper and we leave out their treatment for the sake of simplicity.

Below, we define the state global labelling function ℓ_g:

$$\ell_g(\langle c_1, \dots, c_N \rangle) = \{a \in \mathscr{P}_g \mid \mathscr{E}[\![\text{bexp}_a]\!]_{\mathbf{M}^{(N)}(\langle c_1,\dots,c_N\rangle)} = \text{tt}\}.$$

We obtain the state labelled DTMC $\mathcal{D}^{(N)}(t)$ from $\mathbf{X}^{(N)}(t)$, with transition matrix $\mathbf{P}^{(N)}$ above, by enriching it with labelling function $\ell_{\mathcal{D}^{(N)}}$ such that $\ell_{\mathcal{D}^{(N)}}(\mathbf{C}) = \ell_1(\mathbf{C}) \cup \ell_g(\mathbf{C})$.

The definition of $\mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$ as well as that of the satisfaction relation $\models_{\mathcal{D}^{(N)}}$ are obtained by instantiating those given in Sect. 3.1 to $\mathcal{D}^{(N)}$. For $\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, $\sigma[j]_{[n]}$ denotes the n-th local state of global state $\sigma[j]$.
For model-checking a system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$ we instantiate proc with[10] $\mathcal{S}_\Delta^N$ and lab with $\mathscr{P}_1 \cup \mathscr{P}_g$. Function next is instantiated to the function $\mathsf{next}_{\mathcal{D}^{(N)}}$, where

$$\mathsf{next}_{\mathcal{D}^{(N)}}(\mathbf{C}) = [(\mathbf{C}', p') \mid \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = p' > 0].$$

Given a vector $\mathbf{C}$, $\mathsf{next}_{\mathcal{D}^{(N)}}(\mathbf{C})$ computes a list corresponding to the positive elements of the row of matrix $\mathbf{P}^{(N)}$ associated with $\mathbf{C}$. Of course, only those elements of $\mathbf{P}^{(N)}$ that are necessary for $\mathsf{next}_{\mathcal{D}^{(N)}}$ are actually computed. Function $\mathsf{lab\_eval}$ is instantiated with the function $\mathsf{lab\_eval}_{\mathcal{D}^{(N)}} : \mathcal{S}_\Delta^N \times \mathcal{A}_\Delta \to \mathbb{B}$ with $\mathsf{lab\_eval}_{\mathcal{D}^{(N)}}(\mathbf{C}, a) = a \in \ell_{\mathcal{D}^{(N)}}(\mathbf{C})$.

Example 3 (Properties)**.** *For the epidemic model of Example 1 we can consider the following properties, where $i, e, r \in \mathscr{P}_1$ are labelling states I, E and R, respectively, and LowInf $\in \mathscr{P}_g$ is defined as $(\mathit{frc}\, I) < 0.25$:*

P1 the worm will be active in the first component within k steps with a probability that is at most p: $\mathcal{P}_{\leq p}(\ \mathit{true}\ \mathcal{U}^{\leq k}\ i\ )$;

P2 the probability that the first component is infected, but latent, in the next k steps while the worm is active on less then 25% of the components is at most p: $\mathcal{P}_{\leq p}(\mathit{LowInf}\, \mathcal{U}^{\leq k}\ e\ )$;

P3 the probability to reach, within k steps, a configuration where the first component is not infected but the worm will be activated with probability greater than 0.3 within 5 steps is at most p:

$$\mathcal{P}_{\leq p}(\ \mathit{true}\ \mathcal{U}^{\leq k}\, (!e \wedge !i \wedge \mathcal{P}_{>0.3}(\ \mathit{true}\ \mathcal{U}^{\leq 5}\ i\ ))).$$

In Fig. 2 the result of exact PCTL model-checking of Ex. 1 is reported. On the left the probability of the set of paths that satisfy the path-formulas used in the three formulas above is shown for a system composed of eight objects each in initial state S, for k from 0 to 70. On the right the time needed to perform the analysis using PRISM [44] and using exact on-the-fly PCTL model checking are presented[11], showing that the latter has comparable performance. Worst-case complexity of both algorithms are also comparable. The local model-checker has been instantiated with the model defined by the (exact) operational semantics of the language, where each state $\mathbf{C} \in \mathcal{S}_\Delta^N$ is a *global* system state. In Sect. 5 we instantiate the procedure with the mean-field, approximated, semantics of the language, leading to a scalable, 'fast', model-checker, insensitive to the population size.

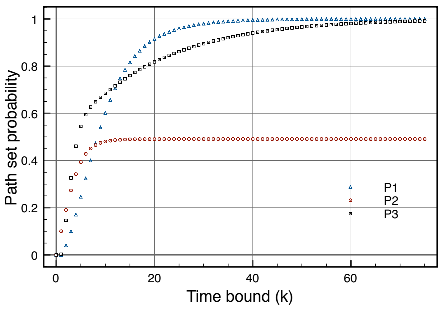
5 Fast Mean-field Model-checking

In this section we present the main result of the paper. We first provide an informal account of the relevant notions and methodology, followed by a detailed technical description.

Recall that we are interested in specifications $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$ of systems characterised by large population sizes N, and their models $\mathcal{D}^{(N)}(t)$ (see Fig. 3). We are aiming at model-checking properties

[10] Strictly speaking, the relevant components of the algorithm are instantiated to *representations* of the terms, sets and functions mentioned in this section. For the sake of notational simplicity, we often use the same notation both for mathematical objects and for their representations.

[11] We use a 1.86GHz Intel Core 2 Duo with 4 GB. State space generation time of PRISM is not counted. The experiments are available at `http://rap.dsi.unifi.it/$\sim$loreti/OFPMC/`).



	PRISM	Exact on-the-fly
$P1$	$108.479s$	$29.587s$
$P2$	$51.816s$	$3.409s$
$P3$	$216.952s$	$85.579s$

Model parameter values:
$\alpha_e = 0.1,\ \alpha_i = 0.2,\ \alpha_r = 0.2$
$\alpha_a = 0.4,\ \alpha_s = 0.1$

Figure 2: Exact model-checking results (left) and verification time (right).

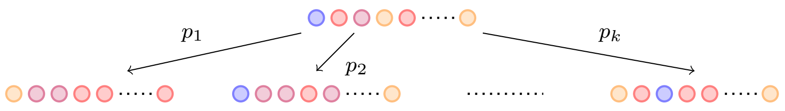


Figure 3: A fragment of $\mathcal{D}^{(N)}(t)$ transition matrix $\mathbf{P}^{(N)}$.

of the behaviour of an individual object within the context of such systems; w.l.g., we focus on the *first* object of $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$. Furthermore, we are interested in properties of the whole system behaviour as well. Given the fact that we are concerned with systems with very large population sizes, we need to abstract from some details of system global states. In particular, we first of all abstract from the identities of all individual objects, except the first one. This is done by using the occupancy measure vector $\mathbf{M}^{(N)}(\mathbf{C}^{(N)})$, as defined in (2) on page 10, for representing each global state $\mathbf{C}^{(N)}$. In practice, the abstraction is performed by means of a mapping $\mathcal{H}^{(N)}$ which associates states $\mathbf{C}^{(N)}$ of $\mathcal{D}^{(N)}$ to pairs $\langle c, \mathbf{m} \rangle$: c is the current state of the first object, i.e. $\mathbf{C}^{(N)}_{[1]}$, while $\mathbf{m} = \mathbf{M}^{(N)}(\mathbf{C}^{(N)})$. This abstraction procedure defines a new labelled DTMC, which we call $\mathcal{HD}^{(N)}(t)$ (see Fig. 4). As we will see, the abstraction preserves the satisfaction of PCTL formulas.

On the other hand, abstraction alone is not yet satisfactory from the point of view of the size of the state space, when one is interested in *very* large population sizes. In fact, one of the major sources of non-determinism, which in turn results in an exponential increase of the state space size, is the intrinsically non-deterministic (actually, stochastic) nature of the occupancy measure vector. Here is where *deterministic approximation* comes into play; we use in fact a well known convergence result by Le Boudec et al. (Theorem 4.1 of [47]) which provides us with a function $\boldsymbol{\mu}$ of time such that, for N large enough, *every* occupancy measure vector of $\mathcal{HD}^{(N)}(t)$ at any fixed time $\tilde{t}$ can be *deterministically* approximated by $\boldsymbol{\mu}(\tilde{t})$. This brings to a drastic reduction in the size of the state space. Furthermore, the deterministic approximation of the occupancy measure vector can be efficiently computed by iteration on t and can be combined on-the-fly with the probabilistic information concerning the first object of the system in the model-checking procedure, following an approach inspired by *Fast Simulation* in Theorem 5.1 of [47]. More specifically, such a combination gives rise to

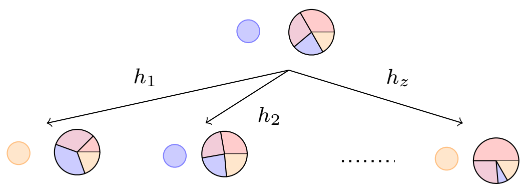


Figure 4: A fragment of $\mathcal{HD}^{(N)}(t)$ transition matrix $\mathbf{H}^{(N)}$.

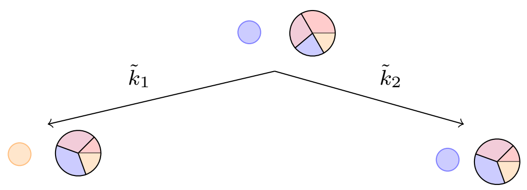


Figure 5: A fragment of $\mathcal{HD}(t)$ transition matrix.

a new state labelled DTMC, $\mathcal{HD}(t)$, which is shown to be the limit of $\mathcal{HD}^{(N)}(t)$, when N goes to infinity (see Fig. 5). Interestingly, for N large enough, the approximation preserves, almost surely, the satisfaction of PCTL formulas; this constitutes the major result of the present paper. On the practical side, we can instantiate the on-the-fly model-checking algorithm on $\mathcal{HD}(t)$, thus achieving *Fast, on-the-fly* bounded PCTL model-checking.

5.1 The Abstraction Step

Given a system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$ with initial state $\mathbf{C}_0$,we want to focus on the behaviour of the first object, starting in the initial state $\mathbf{C}_{0[1]}$, when in execution with all the other objects for (very) large population size N. We define a mapping $\mathcal{H}^{(N)} : \mathcal{S}_\Delta^N \to (\mathcal{S}_\Delta \times \mathcal{U}^S)$ such that $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) = \langle \mathbf{C}_{[1]}^{(N)}, \mathbf{M}^{(N)}(\mathbf{C}^{(N)}) \rangle$. Note that $\mathcal{H}^{(N)}$ and $\mathcal{D}^{(N)}(t)$ together define a state labelled DTMC, denoted $\mathcal{HD}^{(N)}(t)$, and defined as $\mathcal{H}^{(N)}(\mathbf{X}^{(N)}(t))$, with $\ell_1(\langle c, \mathbf{m} \rangle) = \ell_1(c)$, $\ell_g(\langle c, \mathbf{m} \rangle) = \{a \in \mathscr{P}_g \mid \mathscr{E}[\![\mathrm{bexp}_a]\!]_{\mathbf{m}} = \mathrm{tt}\}$, and $\ell_{\mathcal{HD}^{(N)}}(\langle c, \mathbf{m} \rangle) = \ell_1(\langle c, \mathbf{m} \rangle) \cup \ell_g(\langle c, \mathbf{m} \rangle)$, where $\mathscr{P}_1, \mathscr{P}_g$ and bexp_a are defined in a similar way as in Sect. 4. The one-step matrix of $\mathcal{HD}^{(N)}(t)$ is:

$$\mathbf{H}^{(N)}_{\langle c,\mathbf{m}\rangle,\langle c',\mathbf{m}'\rangle} = \sum_{\mathbf{C}' : \mathcal{H}^{(N)}(\mathbf{C}') = \langle c',\mathbf{m}'\rangle} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \qquad (6)$$

where $\mathbf{C}$ is such that $\mathcal{H}^{(N)}(\mathbf{C}) = \langle c, \mathbf{m} \rangle$.[12] The definitions of paths for state $\langle c, \mathbf{m} \rangle$ of $\mathcal{HD}^{(N)}$, $\mathit{Paths}_{\mathcal{HD}^{(N)}}(\langle c, \mathbf{m} \rangle)$, of $L_{\mathcal{HD}^{(N)}}(t)$ and of the satisfaction relation $\models_{\mathcal{HD}^{(N)}}$ of PCTL formulas against $\mathcal{HD}^{(N)}(t)$, are obtained by instantiating the relevant definitions of Sect. 3.1 to the model $\mathcal{HD}^{(N)}(t)$. Furthermore, we let $L_{\mathcal{HD}^{(N)}}(t, c) = \{\langle c', \mathbf{m}' \rangle \in L_{\mathcal{HD}^{(N)}}(t) \mid c' = c\}$.

We extend mapping $\mathcal{H}^{(N)}$ to sets and paths in the obvious way: for set X of states, let $\mathcal{H}^{(N)}(X) = \{\mathcal{H}^{(N)}(x) \mid x \in X\}$, and for $\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, let $\mathcal{H}^{(N)}(\sigma)$ be the path

$$\mathcal{H}^{(N)}(\sigma[0])\mathcal{H}^{(N)}(\sigma[1])\mathcal{H}^{(N)}(\sigma[2]) \cdots$$

Note that, by definition of $\mathcal{H}^{(N)}$ and $\mathcal{HD}^{(N)}(t)$, if $\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, then $\mathcal{H}^{(N)}(\sigma) \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C}^{(N)}$
and if $\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\langle c, \mathbf{m} \rangle)$, then there exist $\mathbf{C}^{(N)}$, with $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) = \langle c, \mathbf{m} \rangle$, and $\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$ s.t. $\mathcal{H}^{(N)}(\sigma) = \rho$. Also, it is easy to see that $\mathcal{H}^{(N)}(L_{\mathcal{D}^{(N)}}(t)) = L_{\mathcal{HD}^{(N)}}(t)$ and $\mathbf{C} \in L_{\mathcal{D}^{(N)}}(t)$ iff $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$. The following lemma relates the two interpretations of the logic (see Appendix A for the proof).

Lemma 2. *For all $N > 0$, states $\mathbf{C}^{(N)}$ and formulas Φ the following holds:*
$\mathbf{C}^{(N)} \models_{\mathcal{D}^{(N)}} \Phi$ *iff* $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$. •

[12] With a similar argument as for definition (5), noting that $\mathbf{M}^{(N)}(\mathbf{C}) = \mathbf{M}^{(N)}(\mathbf{C}'')$ and $\mathbf{C}_{[1]} = \mathbf{C}''_{[1]}$, it can be easily seen that also definition (6) is a good definition.

5.2 The Approximation Step

We now consider the stochastic process $\mathcal{HD}(t)$ defined below, for $c_0, c, c' \in \mathcal{S}_\Delta$, $\boldsymbol{\mu}_0, \mathbf{m}, \mathbf{m}' \in \mathcal{U}^S$ and function $\mathbf{K}(\mathbf{m})_{c,c'}$, continuous in $\mathbf{m}$:

$$\begin{array}{l} \mathbb{P}\{\mathcal{HD}(0) = \langle c, \mathbf{m}\rangle\} = \boldsymbol{\delta}_{\langle c_0, \boldsymbol{\mu}_0\rangle}(\langle c, \mathbf{m}\rangle), \\ \mathbb{P}\{\mathcal{HD}(t+1) = \langle c', \mathbf{m}'\rangle \,|\, \mathcal{HD}(t) = \langle c, \mathbf{m}\rangle\} = \begin{cases} \mathbf{K}(\mathbf{m})_{c,c'}, & \text{if } \mathbf{m}' = \mathbf{m}\cdot\mathbf{K}(\mathbf{m}) \\ 0, & \text{otherwise.} \end{cases} \end{array} \tag{7}$$

The definition of the labeling function $\ell_{\mathcal{HD}}$ is the same as that of $\ell_{\mathcal{HD}^{(N)}}$. Note that $\mathcal{HD}$ is a DTMC with initial state $\langle c_0, \boldsymbol{\mu}_0\rangle$; memoryless-ness as well as time homogeneity directly follow from the definition of the process (7). The definitions of paths for state $\langle c, \mathbf{m}\rangle$ of $\mathcal{HD}$, $Paths_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle)$, of $L_{\mathcal{HD}}(t)$ and of the satisfaction relation $\models_{\mathcal{HD}}$ of PCTL formulas against $\mathcal{HD}(t)$ are obtained by instantiating the relevant definitions of Sect. 3.1 to the model $\mathcal{HD}(t)$. Furthermore, define function $\boldsymbol{\mu}(t)$ as follows: $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$ and $\boldsymbol{\mu}(t+1) = \boldsymbol{\mu}(t)\cdot\mathbf{K}(\boldsymbol{\mu}(t))$; then, for $t \geq 0$ and for $\langle c, \mathbf{m}\rangle \in L_{\mathcal{HD}}(t)$ we have $\mathbf{m} = \boldsymbol{\mu}(t)$.

In the following we use the fundamental result stated below, due to Le Boudec et al. [47]. We recall that, for each N, the occupancy measure process $\mathbf{M}^{(N)}(t)$ is the stochastic process $\mathbf{M}^{(N)}(t) = \mathbf{M}^{(N)}(\mathbf{X}^{(N)}(t))$, with initial distribution $\boldsymbol{\delta}_{\mathbf{M}^{(N)}(\mathbf{C}_0^{(N)})}$ where $\mathbf{C}_0^{(N)}$ is the initial state vector of the given system specification.

Theorem 4.1 of [47] *Assume that for all $c, c' \in \mathcal{S}_\Delta$, there exists function $\mathbf{K}(\mathbf{m})_{c,c'}$, continuous in $\mathbf{m}$, such that, for $N \to \infty$, $\mathbf{K}^{(N)}(\mathbf{m})_{c,c'}$ converges uniformly in $\mathbf{m}$ to $\mathbf{K}(\mathbf{m})_{c,c'}$. Assume, furthermore, that there exists $\boldsymbol{\mu}_0 \in \mathcal{U}^S$ such that $\mathbf{M}^{(N)}(\mathbf{C}_0^{(N)})$ converges almost surely to $\boldsymbol{\mu}_0$. Define function $\boldsymbol{\mu}(t)$ of t as follows: $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$ and $\boldsymbol{\mu}(t+1) = \boldsymbol{\mu}(t)\cdot\mathbf{K}(\boldsymbol{\mu}(t))$. Then, for any fixed t, almost surely $\lim_{N\to\infty}\mathbf{M}^{(N)}(t) = \boldsymbol{\mu}(t)$.* •

Remark 1. *We observe that, as direct consequence of Theorem 4.1 of [47] and of the restrictions on the definition of BExp, for any fixed t and for all $\epsilon > 0$, there exists $\bar{N}$ such that, for all $N \geq \bar{N}$, almost surely*

$$|\ \mathscr{E}[\![bexp]\!]_{\mathbf{m}} - \mathscr{E}[\![bexp]\!]_{\boldsymbol{\mu}(t)}\ | < \epsilon$$

for all $\langle c, \mathbf{m}\rangle \in L_{\mathcal{HD}^{(N)}}(t)$ and $bexp \in BExp$. In other words, for N large enough and $\langle c, \mathbf{m}\rangle \in L_{\mathcal{HD}^{(N)}}(t)$, $\ell_g(\langle c, \mathbf{m}\rangle) = \ell_g(\langle c, \boldsymbol{\mu}(t)\rangle)$, and, consequently, $\ell(\langle c, \mathbf{m}\rangle) = \ell(\langle c, \boldsymbol{\mu}(t)\rangle)$. •

In the rest of the paper we will focus on sequences $\left(\langle \Delta, A, \mathbf{C}_0\rangle^{(N)}\right)_{N\geq N_0}$ of system specifications, for some $N_0 > 0$. In particular, we will consider only sequences $\left(\mathcal{HD}^{(N)}(t)\right)_{N\geq N_0}$ such that for all $N \geq N_0$, $\mathbf{C}_{0}{}_{[1]}^{(N)} = \mathbf{C}_{0}{}_{[1]}^{(N_0)}$; in other words, we want that the population size increases with N, while the (initial state of the) first object of the system is left unchanged.

Let us now go back to process $\mathcal{HD}(t)$, where, in equation (7) we use function $\mathbf{K}(\mathbf{m})_{c,c'}$ of the hypothesis of the theorem recalled above; similarly, for the initial distribution we use $\boldsymbol{\delta}_{\langle \mathbf{C}_{0}{}_{[1]}^{(N)}, \boldsymbol{\mu}(0)\rangle}$.

The following is a corollary of Theorem 4.1 and Theorem 5.1 (Fast simulation) presented in [47], when considering sequences $\left(\mathcal{HD}^{(N)}(t)\right)_{N\geq N_0}$ as above (see also Remark 1):

Corollary 3. *Under the assumptions of Theorem 4.1 of [47], for any fixed t, almost surely*

$$\lim_{N\to\infty} \mathcal{HD}^{(N)}(t) = \mathcal{HD}(t).$$

•

Remark 2. *A consequence of Corollary 3 is that, under the assumptions of Theorem 4.1 of [47], for any fixed t, almost surely, for N to ∞, we have that, for all $\langle c, \mathbf{m}\rangle \in L_{\mathcal{HD}^{(N)}}(t,c)$ and $c' \in \mathcal{S}_\Delta$, $\sum_{\langle c',\mathbf{m}'\rangle : L_{\mathcal{HD}^{(N)}}(t+1,c')} \mathbf{H}^{(N)}_{\langle c,\mathbf{m}\rangle,\langle c',\mathbf{m}'\rangle}$ approaches $\mathbf{K}(\boldsymbol{\mu}(t))_{c,c'}$, i.e. for all $\epsilon > 0$ there exists N_0 s.t. for all $N \geq N_0$*

$$\left| \left(\sum_{\langle c',\mathbf{m}'\rangle : L_{\mathcal{HD}^{(N)}}(t+1,c')} \mathbf{H}^{(N)}_{\langle c,\mathbf{m}\rangle,\langle c',\mathbf{m}'\rangle} \right) - \mathbf{K}(\boldsymbol{\mu}(t))_{c,c'} \right| < \epsilon$$

•

In the sequel we state the main theorem of the present paper. For *large* population sizes N, the probability of the set of paths of $\mathcal{HD}^{(N)}(t)$ satisfying a formula φ, *approaches* the probability of the set of paths of $\mathcal{HD}(t)$ that satisfy φ. In order to guarantee that a formula Φ of form $\mathcal{P}_{\bowtie p}(\varphi)$ holds of a state s of $\mathcal{HD}(t)$ iff it holds of the corresponding state of $\mathcal{HD}^{(N)}(t)$, it is sufficient that Φ is *safe* with w.r.t. $\mathcal{HD}(t)$: a formula Φ is *safe* for a model $\mathcal{M}$ iff for all sub-formulas Φ' of Φ and states s of $\mathcal{M}$, if Φ' is of the form $\mathcal{P}_{\bowtie p}(\varphi)$ then $\mathbb{P}\{\eta \in Paths_{\mathcal{M}}(s) \mid \eta \models_{\mathcal{M}} \varphi\} \neq p$.

The theorem, together with Theorem 1 and Lemma 2, establishes the formal relationship between the satisfaction relation on the exact semantics of the language and that on its mean-field approximation, thus justifying the fast local model-checking instantiation we will show in the sequel.

Theorem 4. *Under the assumptions of Theorem 4.1 of [47], for all safe formulas Φ, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \in L_{\mathcal{HD}^{(N)}}(t)$, almost surely, for N large enough, $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$ iff $\langle \mathbf{C}^{(N)}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi$.* •

Proof. The proof is carried out by induction on Φ (see Appendix B for details). □

Finally, using Lemma 2 we get the following

Corollary 5. *Under the assumptions of Theorem 4.1 of [47], for all safe formulas Φ, for any fixed t and $\mathbf{C}^{(N)} \in L_{\mathcal{D}^{(N)}}(t)$, almost surely, for N large enough $\mathbf{C}^{(N)} \models_{\mathcal{D}^{(N)}} \Phi$ iff $\langle \mathbf{C}^{(N)}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi$.* •

5.3 Fast On-the-Fly PCTL Model-checking

On-the-fly *fast* PCTL model-checking on the limit DTMC $\mathcal{HD}(t)$ is obtained by instantiating proc with $\mathcal{S}_\Delta \times \mathcal{U}^S$ and lab with $\mathscr{P}_1 \cup \mathscr{P}_g$; next is instantiated with $\mathsf{next}_{\mathcal{HD}}$ defined as follows:

$$\mathsf{next}_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle) = [(\langle c', \mathbf{m} \cdot \mathbf{K}(\mathbf{m})\rangle, p') \mid \mathbf{K}(\mathbf{m})_{c,c'} = p' > 0],$$

with $\mathbf{K}(\mathbf{m})_{c,c'}$ as in Theorem 4.1 of [47]; $\mathsf{lab\_eval}$ is instantiated as expected: $\mathsf{lab\_eval}_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle, a) = a \in \ell_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle)$. The instantiation is implemented in $\mathsf{FlyFast}$.

Remark 3. *Although in the hypothesis of the theorem we require formulae safety, for all practical purposes, it is actually sufficient to require that*

$$\mathbb{P}\{\eta \in Paths_{\mathcal{HD}}(s') \mid \eta \models_{\mathcal{HD}} \varphi\} \neq p$$

for all formulas $\mathcal{P}_{\bowtie p}(\varphi)$ and states s' such that $\mathsf{CheckPath}(s', \varphi)$ is computed during the execution of $\mathsf{Check}(s, \Phi)$(see Table 2). This (weaker) safety check is readily added to the algorithm. •

Example 4 ($\mathsf{FlyFast}$ results)**.** *Fig. 6 shows the result of* $\mathsf{FlyFast}$ *on the model of Ex. 1 for the first object of a large population of objects, each initially in state S. In Fig. 6 (left) the same properties are considered as in Ex. 3. The analysis takes less than a second and is* insensitive *to the total population size. Fig. 6 (right) shows how the probability measure of the set of paths satisfying formula true $\mathcal{U}^{\leq k}$ (!e∧!i $\wedge\, \mathcal{P}_{>0.3}($ true $\mathcal{U}^{\leq 5}$ i)) of property* P3 *on page 12, (for $k = 3$), changes for initial time $t0$ varying from 0 to 10.*

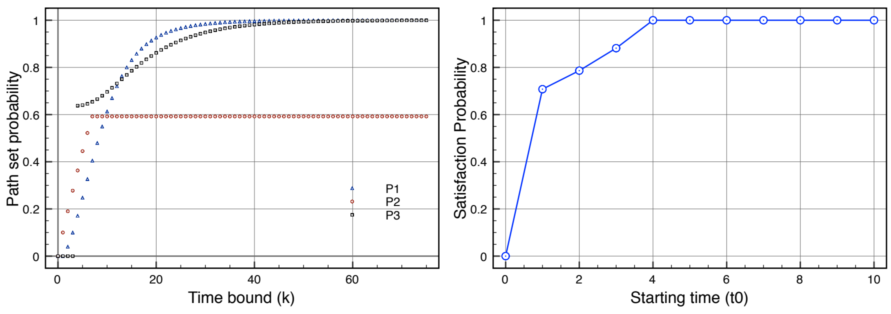


Figure 6: Fast model-checking results.

6 Case Studies

In this section we illustrate the use of the on-the-fly mean field model-checker FlyFast by its application on two selected case studies. One is the well-known and extensively discussed predator-prey model by Lotka and Volterra [48, 63], whereas the other is a more recent model of a bike sharing system inspired by the continuous time stochastic model developed by Fricker and Gast in [29].

6.1 Shared Bikes

Bike sharing systems are a recent, and increasingly popular, form of public transport in urban areas. The idea is that bikes are made available in a number of stations that are placed in various areas of a city. Users that plan to use a bike for a short trip can pick up a bike at a suitable station and return it later on to any other station close to their planned destination. The idea of making bikes available publicly was first experimented in Amsterdam in 1965, where bikes painted white could be used by the citizens to move around the city. Unfortunately, many of them got stolen quickly, and it was not until the introduction of appropriate incentives to make people return the bikes to specific stations that the idea was launched again, such as in Copenhagen where bike sharing was introduced in 1995. Since then over 500 cities in the world have followed including very large cities, such as Paris (20,000 bikes and 1,500 stations) in 2007 and Wuhan and Hangzhou in China (with respectively 90,000 bikes and 60,000 bikes and stations every 100 meters) [21, 64]. Bike sharing systems are of interest not only for providing a useful service, but also contributing to the reduction of greenhouse gases and to stimulate people to physical activity.

One of the major issues in bike sharing systems is the availability and distribution of resources, both in terms of available bikes at the stations and in terms of available empty parking places in the stations, where to park the bikes after using them. Moreover, the demand for these resources vary from station to station with time during the day and with the particular day of the week, due to different needs of users. These needs change for example due to people that rely on the bikes to go to their work or go back home. Therefore the dynamic allocation of resources is managed by the operator of the system, e.g. by moving bikes from full stations to empty ones by means of trucks. One of the ideas to reduce the number of bikes that need to be moved by the operator is to provide incentives to users e.g. to return bikes in less full stations. Fricker and Gast [29] compare the effect of incentives and redistribution mechanisms on the overall performance of the system using mean field approximation techniques in continuous time. In this section we analyse discrete time models inspired by their approach and analyse them using the on-the-fly mean field model-checking techniques introduced in the previous sections.

6.1.1 An Idealised Scenario

The first model we consider is a homogeneous bike sharing model consisting of N stations and a fleet size of $s \cdot N$ bikes, where s is the average number of bikes per station. Each station has K slots to

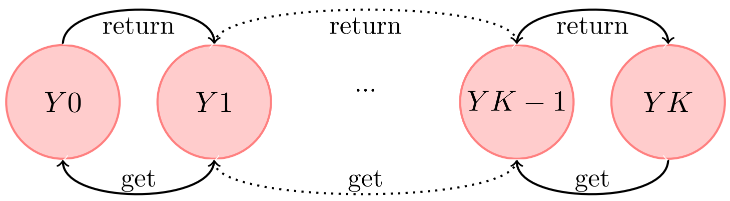


Figure 7: Bike station.

park a bike. We model a station as a discrete time Markov model with $K+1$ states ranging from 0 to K. A station in state i contains i bikes. In every discrete time step, bikes can be taken or returned to the station. This happens with probability α_g ('get') and α_r ('return') respectively. We assume that the duration of the time-steps is small enough so that we need to consider only the return or retrieval of *one bike* at a time in a particular station. In other words, we assume that the probability that two or more bikes are taken or returned in a single step from a single station is negligible and can be abstracted from. Multiple bikes may be returned or taken at *different* stations in the same time step.

The probability that a bike is taken in a station containing k bikes, $0 \leq k \leq K$, at a particular time step depends on k and on the probability α_g. If $k = 0$ then clearly no bike can be taken and in the model the user leaves the system unsatisfied. Similarly, the probability that a bike is returned to a station with k bikes, $0 \leq k \leq K$, at a particular time step depends on the probability α_r and the number of bikes in circulation. Also in this case, if the station has exactly K bikes, the user cannot return the bike and searches for another station. The number of bikes in circulation can be expressed as the difference between the total fleet size and the total of all bikes that are parked in the stations.

The above scenario can be formalised by defining the behaviour of individual stations with capacity K in the following way:

```
Y0 := return.Y1
Y1 := return.Y2 + get.Y0
Y2 := return.Y3 + get.Y1
...
YK-1 := return.YK + get.YK-2
YK := get.YK-1
```

where `Y0` denotes that the station is empty, so no bikes can be taken from it, and `YK` denotes that the station is full, so no bikes can be returned to it. The states `Yk`, where $1 \leq$ `k` $\leq$ `K`-1 denote stations containing `k` bikes. A graphical representation of the model is given in Fig. 7.

The probability that a station changes state is given by the probability of `get` and `return` actions. As described before, the probability that a bike is taken from the station (during a time-step) is α_g. The probability that a bike is returned at the station (during a time-step) depends also on the number of bikes in transit. This number can be expressed as $(sN - \sum_{k=1}^{N} kNy_k)$ where y_k is the fraction of bike parking places that contain exactly k bikes in the time-step under consideration. The fraction of the fleet that is in circulation (C), i.e. non-parked, is then $C = (sN - \sum_{k=1}^{N} kNy_k)/sN$ which gives $C = (s - (\sum_{k=1}^{N} ky_k))/s$ (or equivalently $C = (1 - (\sum_{k=1}^{N} ky_k)/s)$). The probability that a bike is returned to a station is then $\alpha_r C$.

`get ::` α_g;
`return ::` $\alpha_r * ((s - ((\mathsf{frc}\,\mathtt{Y1}) + 2*(\mathsf{frc}\,\mathtt{Y2}) + \ldots + \mathtt{K}*(\mathsf{frc}\,\mathtt{YK})))/s)$;

Following the formal semantics provided in Sect. 4, it is not difficult to obtain the definition of the probability matrix $\mathbf{K}$ and the underlying difference equations of the model.

$$\mathbf{y}(n+1) = \mathbf{y}(n) \cdot \mathbf{K}(\mathbf{y}(n))$$

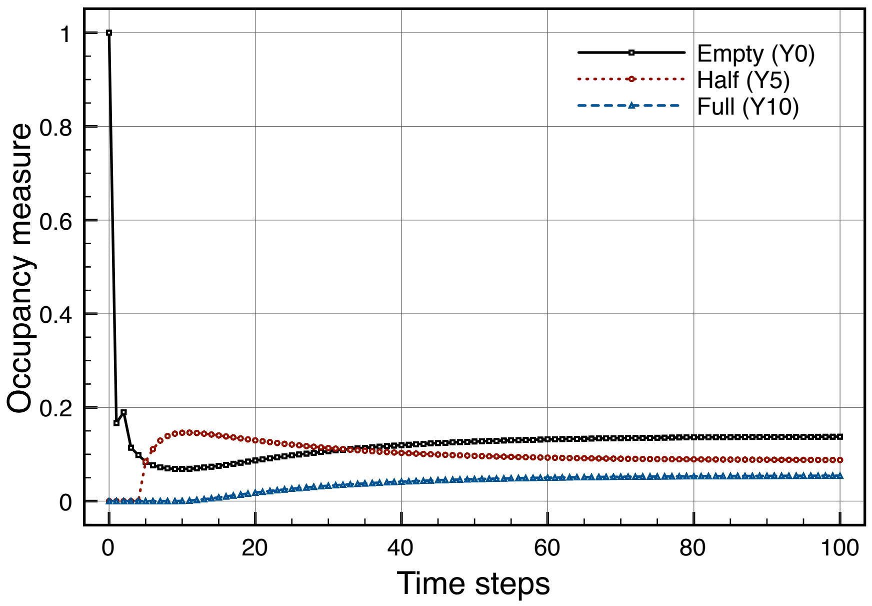


Figure 8: Fraction of empty, half full and full stations for $s = 5$, $req = 1$ (and using $\alpha_g = 1/(s + 1)$ and $\alpha_r = s/(s + 1)$).

where $\mathbf{y}$ denotes the vector of occupancy measures $\mathbf{y}_{[i]}$ of the fraction of stations with i bikes, and the non-zero elements of $\mathbf{K}$, for $1 \leq i \leq K$ are defined as follows, where $C = (s - (\sum_{k=1}^{N} k\mathbf{y}_{[k]}(n)))/s$:

$$\begin{array}{rcl}\mathbf{K}_{i,i-1}(\mathbf{y}(n)) & = & \alpha_g \\ \mathbf{K}_{i-1,i}(\mathbf{y}(n)) & = & \alpha_r C \\ \mathbf{K}_{i,i}(\mathbf{y}(n)) & = & (1 - \alpha_g 1_{i>0} - \alpha_r C \; 1_{i<K})\end{array}$$

The definition of the probabilities α_g and α_r deserve some further attention. Recall that one assumption we made is that there is at most one event (either a request or return of a bike) per time-step per station. This implies that $\alpha_g + \alpha_r \leq 1$. Let u denote the *unit of time* on which the model parameters are based; for simplicity we assume $u = 1$. So, depending on the rate at which bikes are requested in the system, we need to find the appropriate duration of single time-steps so that the assumptions are full-filled. This can be obtained in the following way. First note that the maximum number of bikes returned in a time unit is equal to the fleet size that is $N \cdot s$. Since we have N stations, the average number of bikes returned to a single station in one time unit is s. Letting req be the number of bikes taken from a single station in one time unit, the total number of events at a single station in one time unit is then $s + req$. To see on average at most one event per *time step* we need to have at least $s + req$ time-steps per time unit. The probability to have a request in such a time step is then $\alpha_g = req/(req + s)$ and the probability of a bike being returned is $\alpha_r = s/(req + s)$. Instantiating the above model with $K = 10$, $s = 5$, $\alpha_g = 1/(s + 1)$ and $\alpha_r = s/(s + 1)$ and assuming that all parking places are empty initially, several interesting aspects of the system can be studied. First of all, we can get a quick first impression of the overall behaviour of the system by analysing the occupancy measure over time of the problematic ones that are empty or full and of the half full ones. The result is shown in Fig. 8. We can observe that the number of empty stations quickly decreases while less than 10% of stations get full in the long run. As explained before, the duration of the time step is $1/(s + req)$ that is $\frac{1}{6}$ time units.

We now turn to the analysis of the behaviour of an individual station in the context of the overall system. In what follows we will use $\mathcal{P}_{=?}(\varphi)$ to denote the probability mass of all the paths that satisfy path formula φ. In Fig. 9 we study the probability that an empty station gets full starting from an

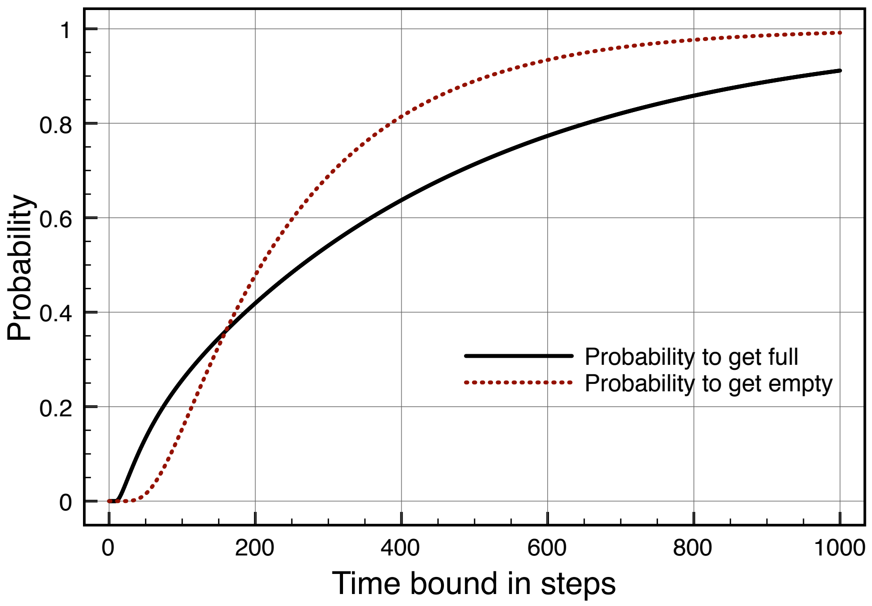


Figure 9: Probability to get full and empty within time bound t expressed in number of steps: $\mathcal{P}_{=?}(\mathrm{tt}\ \mathcal{U}^{\leq t}\ \mathtt{Y10})$ (for $s = 5$, $req = 1$).

initial state of the system in which all the stations are empty. This probability is made available by the model-checker FlyFast by entering the notation $\mathcal{P}_{=?}(\mathrm{tt}\ \mathcal{U}^{\leq t}\ \mathtt{Y10})$. The path formula tt $\mathcal{U}^{\leq t}$ $\mathtt{Y10}$ is satisfied by all the paths in which the single station gets full within t time steps. We let the time bound t vary from 0 to 1000. The bound t is shown on the x-axes of Fig. 9. For example, in the figure we can observe that the probability that the station gets full within 200 time steps is just above 0.4. A similar analysis can be performed starting from an initial state in which all the stations are empty, except the individual station selected for model-checking, which we assume to be full. We can then analyse the probability that this individual station gets empty within t time steps. The result is shown by the dashed curve in Fig. 9.

Fig. 10 shows the probability of an individual station to get full (when being empty initially) and to get empty (when being full initially) within 100 time steps and for initial times ranging from 0 to 50. It is assumed that the overall system at time 0 is empty, i.e. all stations other than the individual one are empty. Fig. 10 shows the probability measure of the set of paths satisfying $\mathcal{P}_{=?}(\ \mathrm{tt}\ \mathcal{U}^{\leq 100}\ Y10\ )$ for the individual station to get full within 100 time steps starting from initial times ranging from 0 to 50. It is interesting to see that for time step 0 the probability of an individual full station to get empty is lower than that of an empty station to get full. This situation is reversed when the same properties are checked for example starting from the overall system in time step 20. This shows the dependence of the properties (or probabilities) on the time at which they are checked, i.e. the time-inhomogeneity of the stochastic model of the individual station.

Fig. 11 shows the global behaviour of a model with a ten times higher request rate compared to the previous model. Since the rate is higher, we also need to rescale the duration of the time steps to $1/(5 + 10)$ of a time unit. It is easy to see that in this model the fraction of empty stations is much higher compared to the model with request rate 1. The fraction of full stations is almost zero. Fig. 12 shows the probability of the individual station to get full or empty, starting from the same initial conditions as in the previous model. As expected, the probability to get empty quickly increases with a higher time bound, whereas the probability to get full increases only slowly.

Note that the behaviour of the model stabilises rather quickly in general. The information on the fractions of stations that are full or empty in such stable situations can provide useful information on

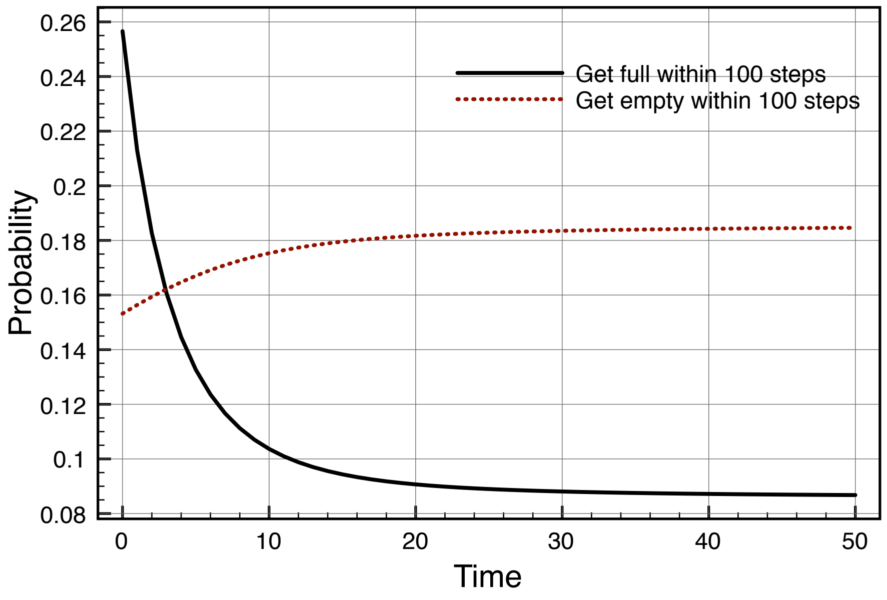


Figure 10: Probability of the individual station to get full (empty resp.) within 100 steps starting from the individual station in state Y0 (Y10 resp.) and the overall system empty initially, for initial times ranging from time step 0 to 50 , with $s = 5$ and $\alpha_g = 1/(s + 1)$ and $\alpha_r = s/(s + 1)$.

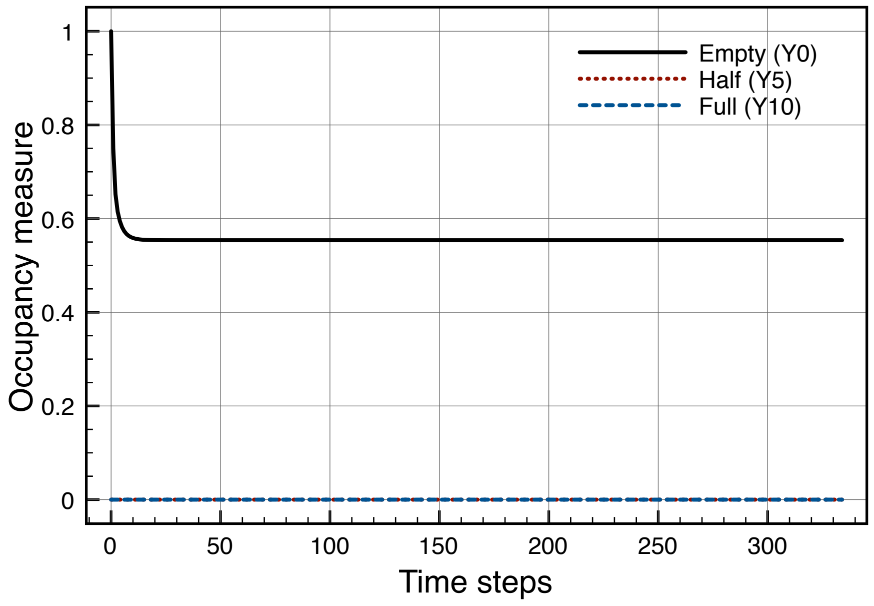


Figure 11: Fraction of empty, half full and full stations for $s = 5$, $req = 10$ (with $\alpha_g = 10/(s + 10)$ and $\alpha_r = s/(s + 10)$).

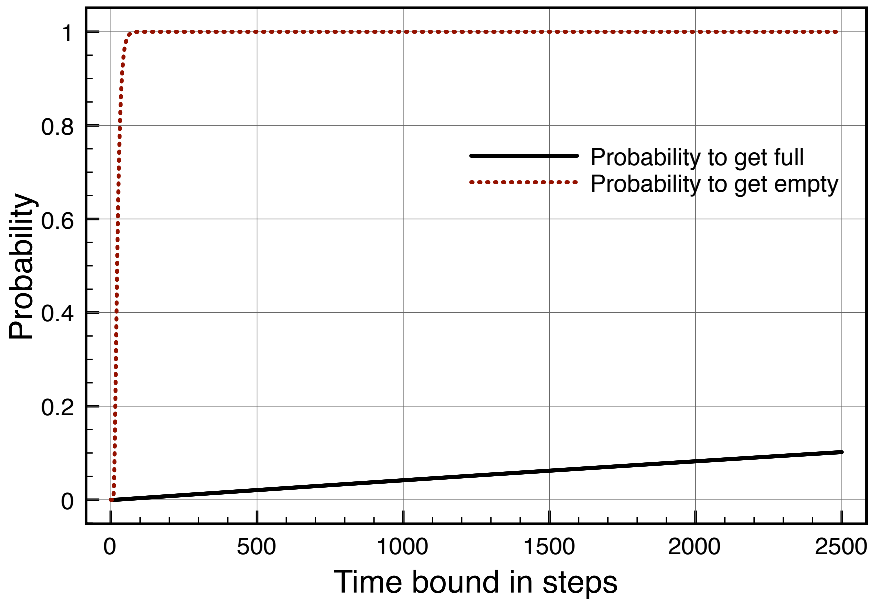


Figure 12: Probability to get full within time bound t expressed in number of steps: $\mathcal{P}_{=?}(\text{tt}\ \mathcal{U}^{\leq t}\ \texttt{Y10})$ (with $s = 5$, $req = 10$).

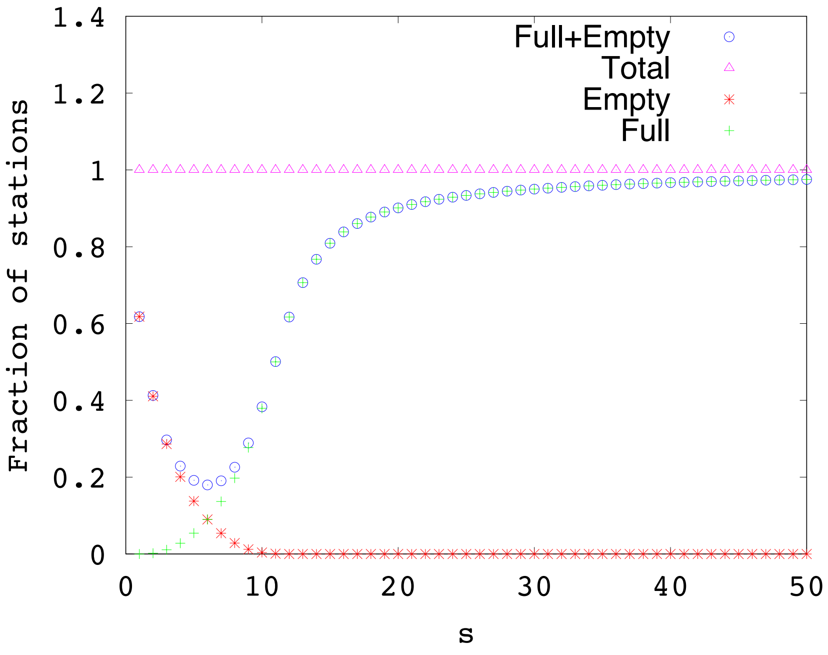


Figure 13: Fraction of problematic stations for varying values of s where $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$.

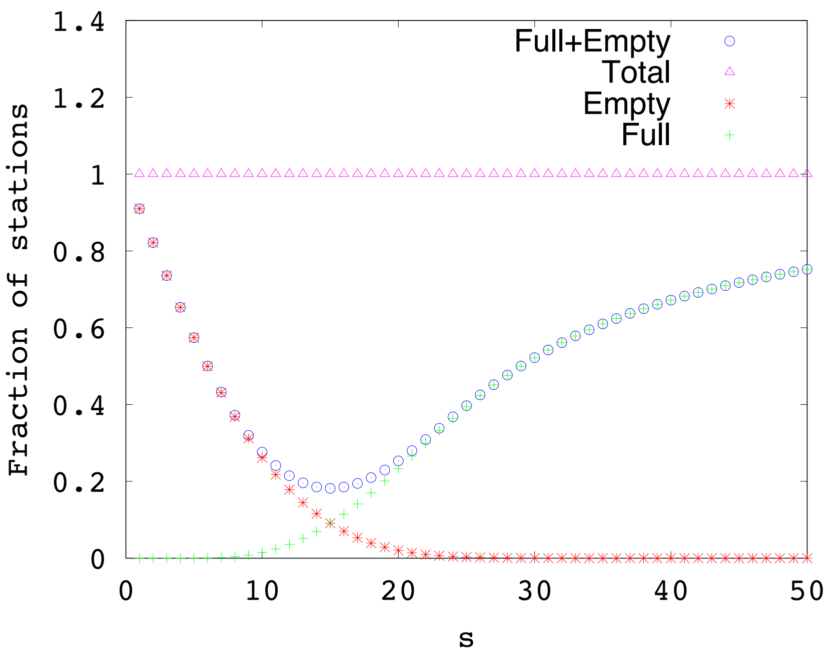


Figure 14: Fraction of problematic stations for varying values of s where $\alpha_g = 10/(s+10)$ and $\alpha_r = s/(s+10)$.

what can be expected of the average quality of service of the system. In particular, it is interesting to know how this depends on the average number of bikes per station s given the number of parking slots in each station. Fig. 13 shows the fraction of problematic stations at time step 100 (curve shown by circles), which is a point in time in which the system has reached approximately stable behaviour for values of s ranging from 1 to 50. Note that for an average number of bikes per station smaller than 5 or 6 the fraction of problematic stations is decreasing, reaching a minimum at around $s = 6$ and then increasing again, due to an increasing fraction of full stations for higher values of s. A similar pattern was observed for the continuous time Markov model analysed by Fricker and Gast [29] which inspired our discrete probabilistic model. So, given the characteristics of a system in terms of probabilities that bikes are requested and returned per time step, the fleet size and the size of stations, the model provides an (idealised and simplified) view on the expected fraction of problematic stations, and thus an indication of expected user satisfaction. The results can be obtained by a transient analysis of the system for different values of s that provides the occupancy measure of the variable of interest at time step 100.

Fig. 14 shows the results assuming a ten times higher request rate compared to the return rate. The higher request rate causes more stations to be empty, and the optimum (smallest fraction of problematic stations) is reached for somewhat higher values of s.

6.1.2 A Scenario with Incentives

One of the main problems with bike sharing systems is the need to redistribute bikes from low request/high return stations and high request/low return ones. This is usually done by special trucks that can carry several bikes at a time. However, if re-distribution is not performed efficiently, this may increase costs of running the system and also produce pollution. One of the ideas that has been proposed is to give shared bike users incentives to return bikes to less full stations or to take them from the fuller ones if there is a choice. Such a choice could be offered via smart interfaces that provide real-time information on the situation of stations. We present an extension of the previous

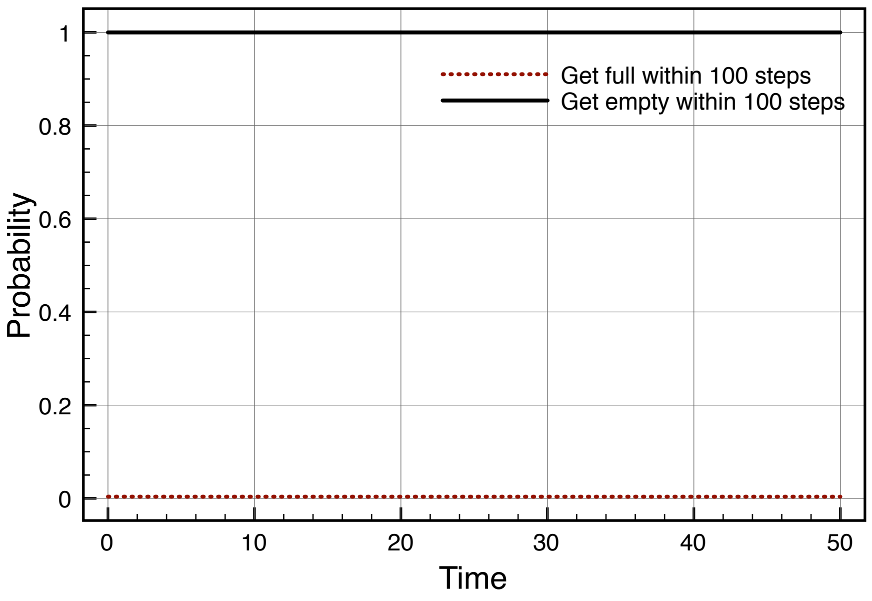


Figure 15: Probability of the individual station to get full (empty resp.) within 100 steps starting from the individual station in state Y0 (Y10 reps.) and the overall system empty initially, for initial times ranging from 0 to 50, with $s = 5$ and $\alpha_g = 10/(s + 10)$ and $\alpha_r = s/(s + 10)$.

bike sharing model that models a particular strategy of incentives, again taking inspiration from a continuous Markov model presented by Fricker and Gast in [29].

A well-known load balancing strategy is selecting the best among two offered choices (see Mitzenmacher [54]). In the case of bike sharing, in an idealised and simplified setting, this strategy consists in the user returning the bike to the station with fewer bikes among two randomly selected ones. This strategy is clearly a first idealised approximation, but may serve to get a first impression of the potential of the strategy in general terms. If the strategy seems promising, then other, more detailed and realistic analyses can be performed on specific, promising configurations. This is also the method followed in [29], where an analytic study is followed by a detailed simulation of a more realistic setting. It is an advantage to first investigate strategies of incentives on more abstract models because these can be analysed much quicker and provide information to select only the most promising configurations for a more detailed and more costly or time consuming analysis.

An advantage of the methodology proposed in this paper is the use of a high-level modelling language that facilitates the modelling and analysis of self-organising systems in several ways. First of all, the modelling takes place at a higher level of abstraction, so the focus of attention can be in terms of the model. The possibly large set of relatively complicated underlying difference equations are derived automatically. This is an advantage when several variants of a model need to be studied where each is slightly different from the other. Manipulating large sets of difference equations manually may easily lead to errors. Furthermore, although analytic solutions in a continuous time setting may be preferred for their generality, often real-world situations show many forms of irregularity or asymmetry, which makes the mathematics that need to be applied particularly hard and tedious, if closed form solutions can be found at all. Numeric approximations may provide still informative results in those cases, and are relatively easier and faster to obtain. This is an important consideration in particular when one is interested in first getting an impression of the behaviour of a self-organised model, in particular of its possible emergent aspects. This means to a certain extent having the option to experiment with a model or ideas for strategies before defining promising and more finalised ones.

Let us now turn to the model and include the above described ideas for user incentives. We mainly need to adjust the probability functions modelling the return of a bike. The probability that a bike is returned to a station with $k-1$ bikes still depends on α_r and on the proportion of bikes in transit in the system, but also on the fact that among two randomly selected stations, the one with the smallest number of bikes is exactly a station with $k-1$ bikes. This is the case when first a station with $k-1$ bikes is selected, and then one with more than $k-1$ bikes, or the other way around, and in the case that two stations with $k-1$ bikes are selected. More formally, from a system point of view at the macro-level this probability is:

$$\mathsf{frc}\,(Y_{k-1}) * (2 * (\mathsf{frc}\,(Y_k) + ... + \mathsf{frc}\,(Y_K)) + \mathsf{frc}\,(Y_{k-1}))$$

At the level of a single station, and in particular if we want to consider models in which the average number of bikes per station, s, exceeds the number of parking slots per station, K, we need to handle this probability with some care because the maximum number of bikes that can be parked in that case is less than the fleet size $s \cdot N$. In fact, the probability that a particular station with $k-1$ bikes will be selected to park a bike is $(\mathsf{frc}\,(Y_k) + ... + \mathsf{frc}\,(Y_K)) + 0.5 * \mathsf{frc}\,(Y_{k-1})$, and the fact that this may happen to the first or the second station that is randomly selected by the user, which explains the factor 2, can be reflected in the probability α_r with which bikes are returned. In particular, the maximum rate of an event is now $2s + req$, and the return probability $\alpha_r = 2s/(2s + req)$, whereas $\alpha_g = req/(2s + req)$. This leads to the following probability function for returning a bike to a station with $k-1$ bikes:

$$\begin{aligned}\texttt{returnAtk-1 ::}\quad & \alpha_r * ((s - ((\mathsf{frc\,Y1}) + 2 * (\mathsf{frc\,Y2}) + \ldots + \mathsf{K} * (\mathsf{frc\,YK})))/s)* \\ & ((\mathsf{frc\,Yk} + ... + \mathsf{frc\,YK}) + 0.5 * (\mathsf{frc\,Y(k-1)})));\end{aligned}$$

Note furthermore that in this model the probability function to return a bike depends on the number of bikes the station contains. The full specification is therefore:

```
Y0 := returnAt0.Y1
Y1 := returnAt1.Y2 + get.Y0
Y2 := returnAt3.Y3 + get.Y1
...
YK-1 := returnAtK-1.YK + get.YK-2
YK := get.YK-1
```

and the probability functions:

$$\begin{aligned}\texttt{get ::}\quad & \alpha_g; \\ \texttt{returnAt0 ::}\quad & \alpha_r * ((s - ((\mathsf{frc\,Y1}) + 2 * (\mathsf{frc\,Y2}) + \ldots + \mathsf{K} * (\mathsf{frc\,YK})))/s)* \\ & ((\mathsf{frc\,Y1} + ... + \mathsf{frc\,YK}) + 0.5 * (\mathsf{frc\,Y0})); \\ \texttt{returnAt1 ::}\quad & \alpha_r * ((s - ((\mathsf{frc\,Y1}) + 2 * (\mathsf{frc\,Y2}) + \ldots + \mathsf{K} * (\mathsf{frc\,YK})))/s)* \\ & ((\mathsf{frc\,Y2} + ... + \mathsf{frc\,YK}) + 0.5 * (\mathsf{frc\,Y1})); \\ \ldots & \\ \texttt{returnAtK-1 ::}\quad & \alpha_r * ((s - ((\mathsf{frc\,Y1}) + 2 * (\mathsf{frc\,Y2}) + \ldots + \mathsf{K} * (\mathsf{frc\,YK})))/s)* \\ & ((\mathsf{frc\,YK}) + 0.5 * (\mathsf{frc\,YK-1}));\end{aligned}$$

Fig. 16 shows the fraction of empty, half full and full stations of the model with incentives for $s = 5$ and $req = 1$ for the first 180 time steps. Compared to the model without incentives it can be observed that the number of problematic stations, those empty or full, is much reduced, and that there are more stations that are half full instead. This indicates that there is indeed a better distribution of bikes over the stations. Note that for this model we have $2 * s + req$ time steps per time unit instead of $s + req$. This explains why in the figure we show 180 time steps instead of 100 to cover the same duration in terms of time units. This scaling applies to all the results for the model with incentives.

Fig. 17 shows the probability that an empty station gets full (resp. empty) starting from an initial state of the system in which all the stations are empty in the model with incentives. The probability

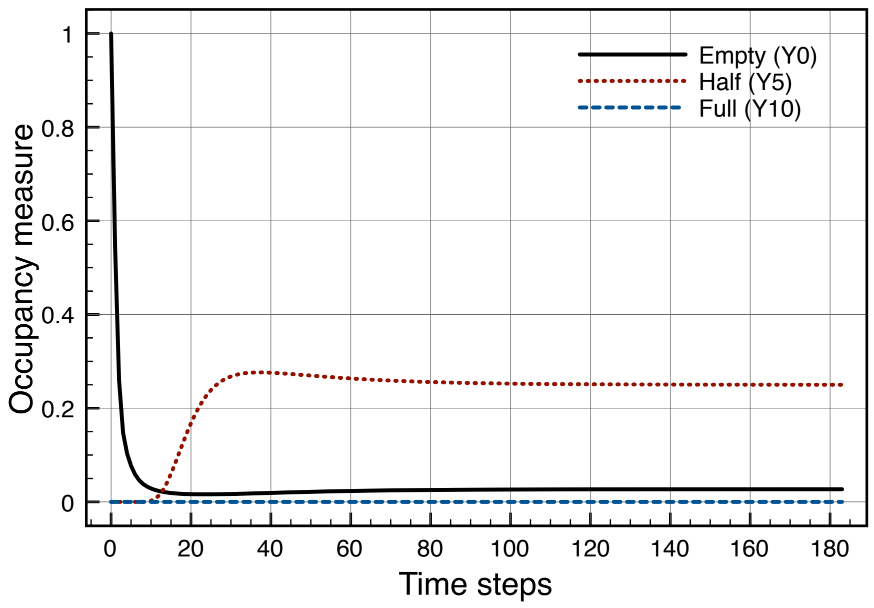


Figure 16: Fraction of empty, half full and full stations for $s = 5$, $req = 1$ (and using $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$).

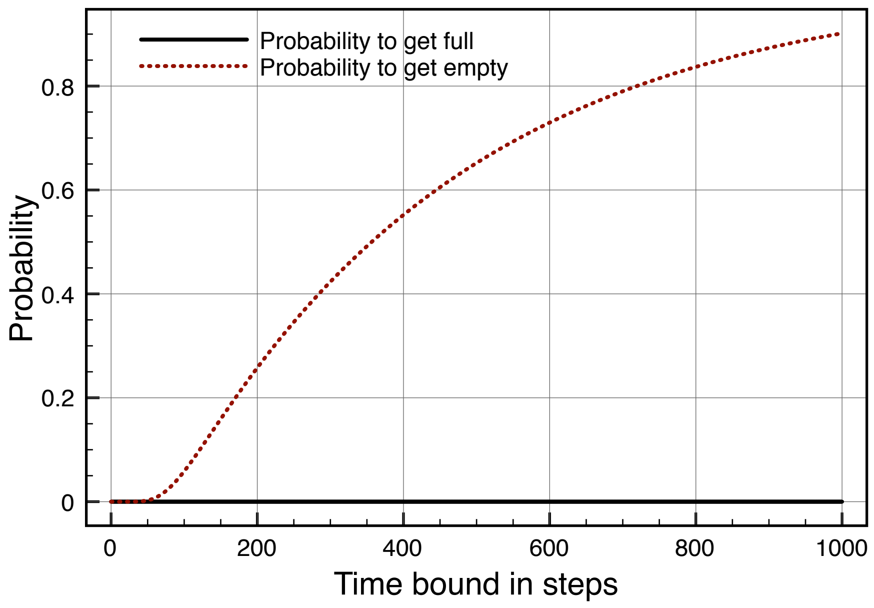


Figure 17: Probability to get full and empty within time bound t expressed in number of steps: $\mathcal{P}_{=?}(\text{tt}\ \mathcal{U}^{\leq t}\ \texttt{Y10})$ (for $s = 5$, $req = 1$).

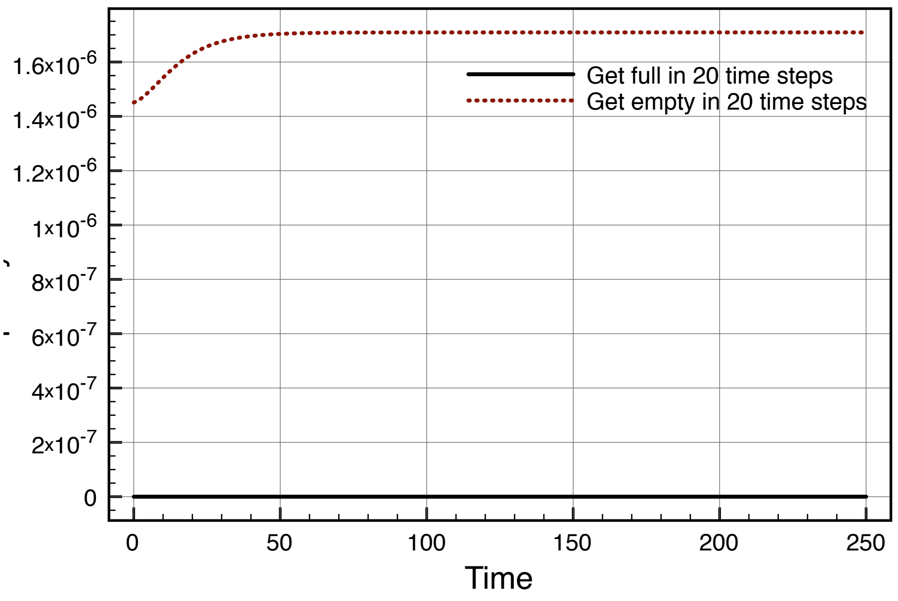


Figure 18: Probability of the individual station to get full (empty resp.) within 20 steps starting from the individual station in state Y0 (Y10 resp.) and the overall system empty initially, for initial times ranging from time step 0 to 250 , with $s = 5$ and $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$.

that a station gets empty is still considerable, but the probability that the station gets full dropped to zero. The related formula for the station getting full is $\mathcal{P}_{=?}($ tt $\mathcal{U}^{\leq t}$ $Y10$), the other is similar.

Fig. 18 shows the probability of an individual station to get full (when being empty initially) and to get empty (when being full initially) within 20 time steps and for initial times ranging from 0 to 250 for the model with incentives. Although the probability to get empty is non-zero, this probability is nevertheless very small. The formula for the station to get full is $\mathcal{P}_{=?}($ tt $\mathcal{U}^{\leq 20}$ $Y10$), the one for getting empty is similar.

As before, we can analyse the fraction of problematic stations in the presence of user incentives for s varying from 1 to 50. In Fig. 19 the results show that there is indeed a significant improvement, in the sense that for values of s around 8 we obtain a situation in which there are nearly no problematic stations, whereas in the model without incentives we had a minimum of 20% of problematic stations. Fig. 20 shows similar results for a model with a ten times higher request rate compared to the return rate. In fact, the minimum is now obtained for a larger fleet size sN.

Similar results are obtained for a model with incentives and a ten times higher request rate compared to the return rate. The results are shown in Fig 20.

6.1.3 Some final considerations

Although the analysis results of these models suggest that the proposed strategy of user incentives leads to a considerable reduction of the fraction of problematic stations in the system, at least when an appropriate fleet size is chosen, the results are based on an idealised scenario. In particular, it has been assumed that users return their bike to the least occupied station out of two randomly chosen ones. Nothing was required about the position of these two stations, and in principle they could be in two different parts of the city, which is not a very realistic scenario. However, such results provide a first indication that the strategy may work, and that further, more detailed analysis is worth the effort. In fact, stochastic simulation of a more realistic model in which the users can choose the least occupied station among two *adjacent* ones shows that also in this more realistic setting the strategy

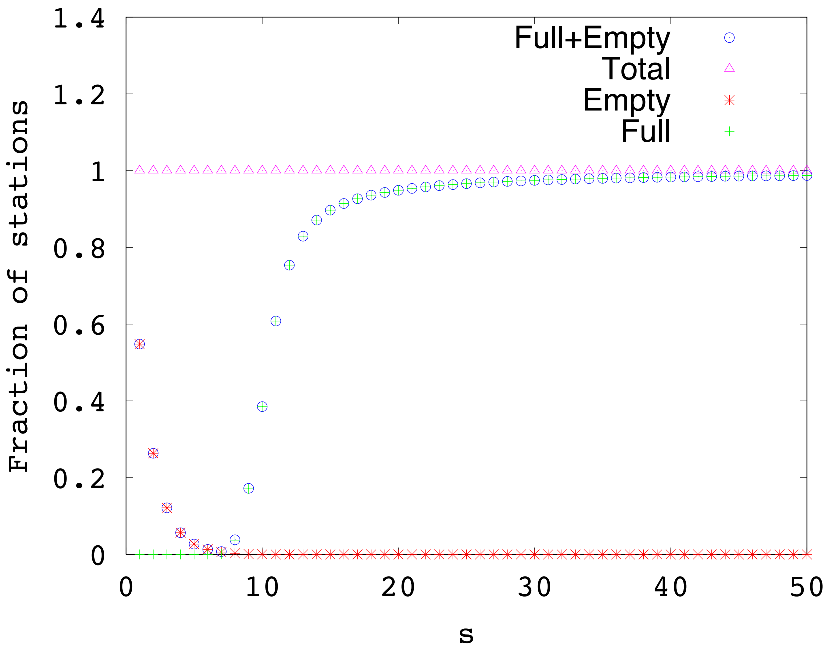


Figure 19: Fraction of problematic stations for varying values of s where $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$.

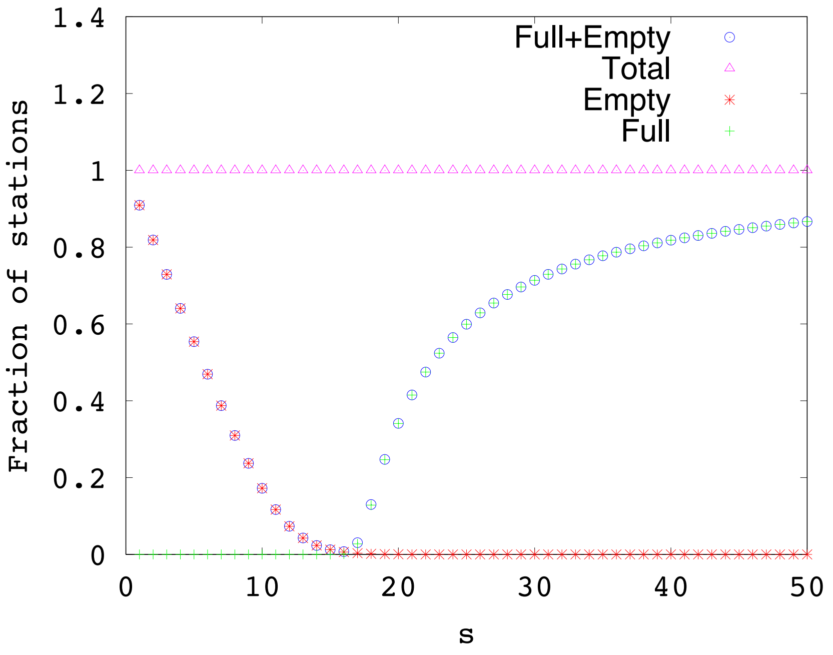


Figure 20: Fraction of problematic stations for varying values of s where $\alpha_g = 10/(s+10)$ and $\alpha_r = s/(s+10)$.

leads to a significant improvement, even in the situation in which only 20% of the users follow the incentive [29]. It would be interesting to see whether similar results can be obtained using statistical model-checking instead of stochastic simulation as a complementary analysis technique.

6.2 Predator-prey model of Lotka-Volterra

The next example we consider is a widely studied model for ecological competition, first independently investigated by the biophysicist Alfred Lotka and the Italian mathematician and physicist Vito Volterra in the twenties of the 19th century [48, 63]. Since then, the model has been studied extensively by numerous other scientists and some of its elements are still at the basis of many population models that have been developed in the course of time, both in continuous time, e.g. [33, 20] and references therein, and in discrete time settings, e.g. [24]. In its simplest form the model can be interpreted as a simplified and idealised description of two species in an ecosystem, often indicated as predator and prey, or foxes and rabbits for a concrete example.

In the variant we consider here we assume that each element of the two species can be in one of two states; its is either alive, or it is somehow 'dormant' waiting to get born again. We do this because the language we use does not provide explicit constructs for the dynamic creation of objects and is implicitly assuming that the total population size of all species remains constant. If we choose the size of the 'dormant' part of the population of each species large enough, this should not have any effect on the part of the population that is alive, since there are always enough dormant rabbits and foxes to get born.

As in the original version, we assume that the model depends on four parameters:

- The net probability 'a' of an increase in the size of the rabbit population which is the difference between the natural birth and death probabilities.
- The probability 'b' of rabbits that die because they are eaten by foxes
- The probability 'e' of extra foxes being born and surviving because they eat rabbits (efficiency).
- The net probability 'c' of the natural decrease in the population of foxes. Since the life of a fox depends on the availability of rabbits, there is a natural tendency of foxes to die when there a few or no rabbits

A model in terms of difference equations of the populations of foxes and rabbits can then be given by:

$$\begin{aligned}
RD(n+1) &= RD(n) + b.h.RL(n).FL(n) - a.h.RL(n)\\
RL(n+1) &= RL(n) + a.h.RL(n) - b.h.RL(n).FL(n)\\
FD(n+1) &= FD(n) - e.h.RL(n).FL(n) + c.h.FL(n)\\
FL(n+1) &= FL(n) + e.h.RL(n).FL(n) - c.h.FL(n)
\end{aligned}$$

where RD and RL are the fractions of 'dormant' and 'alive' rabbits, respectively, and FD and FL the fractions of 'dormant' and 'alive' foxes, respectively. The factor h is a rescaling factor for the duration of steps and $0 < h < 1$. The smaller the value of h the smaller the relative probabilities of the different events and the more accurate the results, but at the cost of an increase of the number of steps per time-unit in the model-checking procedure, which takes more time [13]. For the model in this section we chose $h = 0.125$.

In terms of the language we introduced in Sect. 4, we can present the model as:

[13]Note that when this discrete model is interpreted as an approximation of the well-known continuous time model, i.e. in terms of differential equations, this approximation is not perfect, in the sense that the solution of the differential equations would give a perfect oscillating behaviour, whereas the solution of the difference equations will result in a small error in each step. This error has a cumulative effect resulting in oscillations with ever higher peaks, as can easily be observed in the results. A better approximation of the continuous model could be reached by using a more sophisticated integration method instead of the Euler method that is used implicitly in this case study. In this paper we focus on discrete models, leaving the relation to and approximation of continuous models for further work.

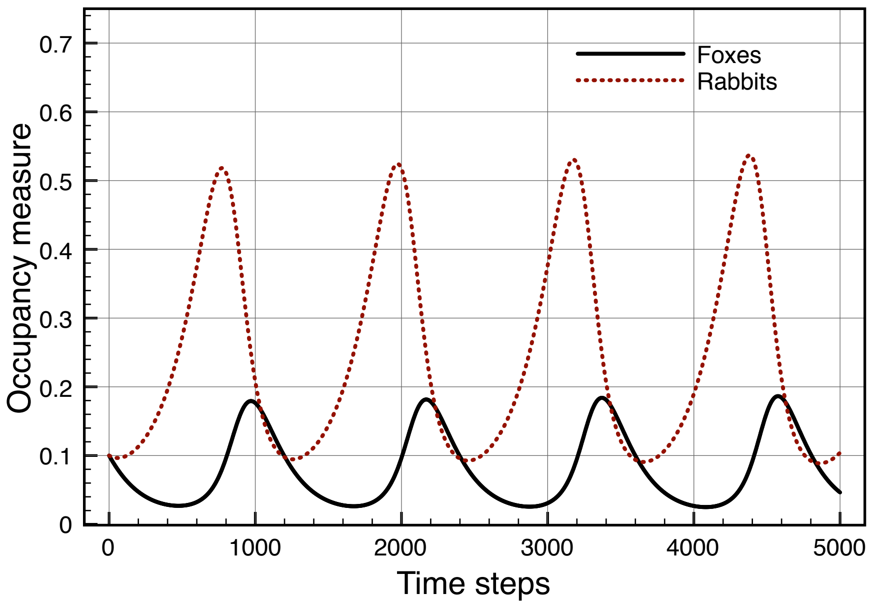


Figure 21: Fraction of rabbit and fox populations.

```
RD := rborn.RL
RL := rdies.RD

FD := fborn.FL
FL := fdies.FD
```

Where the probability functions can be defined as:

rborn :: $\alpha_a * h * (\textsf{frc}\, RL)/(\textsf{frc}\, RD)$;
rdies :: $\alpha_b * h * (\textsf{frc}\, FL)$;

fborn :: $\alpha_e * h * (\textsf{frc}\, RL) * ((\textsf{frc}\, FL)/(\textsf{frc}\, FD))$;
fdies :: $\alpha_c * h$;

These probability functions perhaps need some explanation. The probability that a rabbit gets born in a particular time step depends on the net probability of rabbits getting born, α_a, and on the number of rabbits that are alive. The probability that a rabbit dies depends on the probability α_b and on the number of alive foxes. The probability that a fox is born depends on the efficiency probability α_e and the number of alive rabbits and foxes. Finally, the probability that a fox dies depends on the probability α_c.

As is well known, the global behaviour of the (idealised) model shows oscillations in the populations of rabbits and foxes for certain values of the model parameters. In fact, the model has very interesting behaviour and is therefore widely studied, but in this paper we focus mainly on the illustration of the application of fast mean field model checking of an individual rabbit or fox in the context of the overall oscillating behaviour. For example, for the following values of the four parameters, $a = 0.04$, $b = 0.5$, $e = 0.2$ and $c = 0.05$, and for the initial population sizes of rabbits and foxes of $RD = 5000$, $RL = 1000$, $FD = 3000$ and $FL = 1000$, we obtain the results for the occupancy measure varying over time shown in Fig. 21.

The results in Fig. 21 have been obtained by a numerical analysis of the difference equations that are associated to the model and that are implicitly provided by the FlyFast model checker.

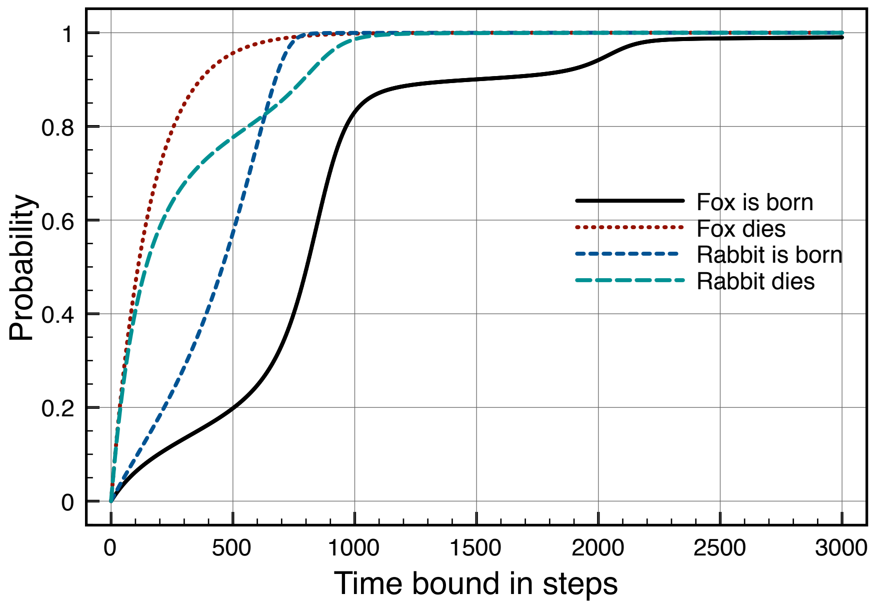


Figure 22: Probability that a fox (rabbit) gets born or dies within time bound t ranging from 0 to 3000 time steps.

In the predator-prey model one could furthermore be interested to know what is the probability that a rabbit survives for a certain amount of time, and how this probability changes over time with the oscillation of the population of foxes. Fig. 22 shows the probability that a fox gets born or dies within time bound t ranging from 0 to 3000 time steps. It also shows the results for a rabbit getting born or dying. The formula for the probability that a rabbit gets born within t time steps is $\mathcal{P}_{=?}(\ RD\ \mathcal{U}^{\leq t}\ \ RL\ )$. The other formulas are similar. Fig. 22 shows that both foxes and rabbits eventually get born and die when given enough time and starting from the initial state of the overall system. The curves also reflect the oscillations in the populations over time and consequently the change in probability to get born or die.

Fig. 23 shows the time-dependent probability of a fox and a rabbit to get born or die in the next 10 time steps, starting from initial times ranging from 0 to 5000. The probability that a particular rabbit dies is obtained by evaluating the property $\mathcal{P}_{=?}(\ RL\ \mathcal{U}^{\leq 10}\ \ RD\ )$, for different initial times. The other formulas are similar. In this oscillating system the time-dependence of these probabilities can be observed very well. The probabilities of a rabbit getting born (dies resp.) within 10 time units and a fox getting born follow closely the oscillations in the respective population sizes. The probability that a fox dies in this model is constant. The amplitude of the oscillations is slowly increasing. This is likely due to the accumulation of small errors in the computation due to the constant step size used in the computations. In fact it is well-known that a mean-field approximation may become less accurate on the longer run. See e.g. [20] for a study of this aspect of the Lotka-Volterra model in the continuous time setting. In future work we plan to address these aspects in more detail.

Finally, Fig. 24 shows the time dependent probability of reaching a state, within 100 time steps, in which the probability of an individual rabbit to die within 10 time steps is higher than 0.2. This probability is shown for different initial times ranging from 0 to 5000. This is a typical example of a 'nested' formula involving two occurrences of the until operator. The formula is:

$$\mathcal{P}_{=?}(\ \mathrm{tt}\ \mathcal{U}^{\leq 100}\ \ (\ RL\ \wedge\ \mathcal{P}_{>0.2}(\ RL\ \mathcal{U}^{\leq 10}\ \ RD\ )))$$

The figure shows that there are indeed relatively short periods in which such states can be reached

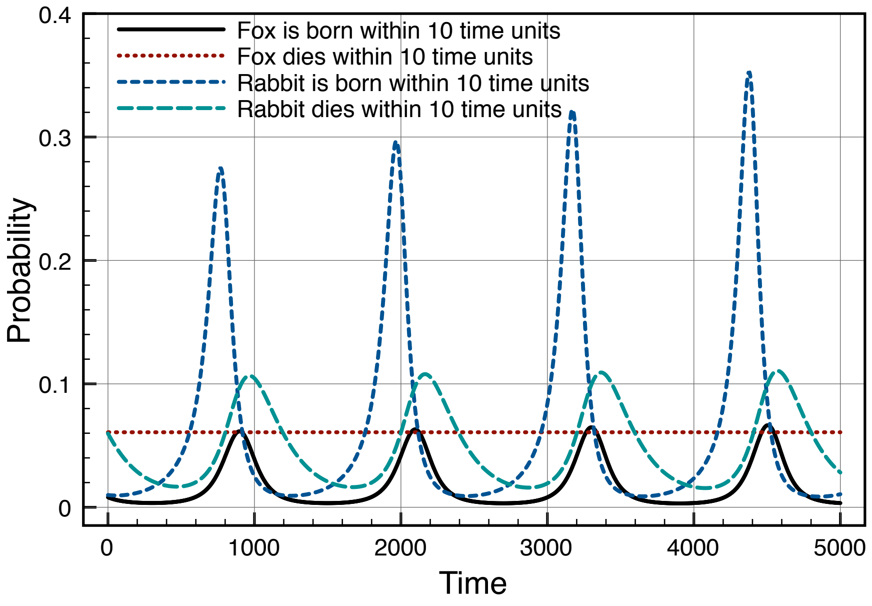


Figure 23: Time dependent probability to get born or die in the next 10 time steps for different initial times from 0 to 5000.

within 100 time steps. Nested formulas are relatively easy to handle due to the iterative and recursive way in which the FlyFast model-checker works.

7 Discussion

Collective, self-organising systems are of great interest for certain types of applications due to their potential robustness and self-adaptivity. On the other hand, their behaviour may be very complex and difficult to analyse and predict, in particular when also aspects such as heterogeneity of populations, different time-scales, uncertainty and spatial inhomogeneity are taken into consideration. The model-checking technique that we introduced is a first step towards addressing the analysis of the many complex aspects of collective self-organised systems. Traditional model-checking has shown to be a very useful technique in the system design phase of concurrent and distributed systems. It provides designers with a high-level modelling and verification tool to analyse relatively quickly the often unpredictable behaviour of concurrent systems. The exploitation of mean-field approximation techniques, as adopted in this paper, is one of the ways to overcome scalability problems that often occur in traditional model-checking due to the combinatorial explosion of the state space when there are very many interacting components in the system.

In this paper we have shown that the foundations of the combination of on-the-fly model-checking and mean field approximation in a discrete time setting is feasible and promising. There are several issues that we have not addressed in the current paper but that we are interested in developing in future work. Among these we briefly outline the following issues:

Heterogeneity Most collective self-organised systems involve several different populations and interactions between them. We have seen a small first example of this in the predator-prey model in the previous section. Although we have not addressed this issue formally, it is not difficult to extend the introduced model-checking framework to address multiple populations. However, from a modelling point of view, it would be desirable to develop a suitable compositional process algebraic probabilistic language to model the behaviour of individuals and in particular

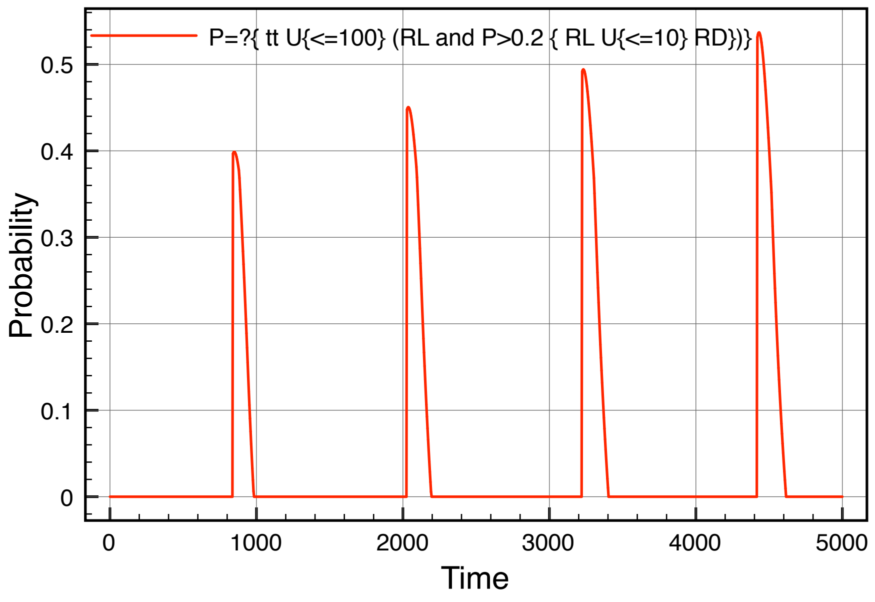


Figure 24: Time dependent probability of reaching a state, within 100 time steps, in which the probability of an individualal rabbit to die within 10 time steps is less than 0.2 for different initial times ranging from 0 to 5000.

their interaction. Such process algebras have been successfully proposed in the continuous time stochastic setting, see for example [16]. Also for population models in the probabilistic discrete time setting there have been some proposals in this direction, e.g. [53], but further research is needed. In particular, we would be interested in a version that is compatible with the stochastic process algebras used in the continuous time setting and that can be related to those in a formal way, for example by means of uniformisation techniques following the ideas outlined in [9].

Compositionality The importance of compositionality of the modelling language, also in the context of self-organised systems, should not be underestimated. A compositional high-level language provides the designer with a relatively simple way to investigate different variants of a model or extensions of a model. The global behaviour of such systems can often also be described as the solution of a set of difference equations, in fact the same equations that are implicitly exploited in the model-checking algorithm. However, developing extensions or variations of the models at the level of difference equations, which are not compositional, is error-prone and often difficult. It is therefore convenient to derive the underlaying difference equations automatically from a high-level compositional specification either implicitly as in model-checking, or explicitly for inspection or other forms of analysis.

Asymmetry in models A common approach in formal modelling is to start from idealised and simplified models that represent the essence of an idea or strategy and then to enrich or extend such models in a step-wise fashion with more realistic aspects. Often such aspects lead to models with some form of irregularity such as heterogeneity of populations or inhomogeneous spatial features. An example of such model-development can be found in [7] where a model of crowd movement between several more or less attractive city squares was studied using fluid flow analysis in a continuous time setting. The study shows that even though the simplest regular and symmetric models can still be solved analytically, leading to useful general results, this is no longer feasible for asymmetric models, where a numeric approach is more adequate. The study also shows that even relatively minor changes to the model, both in terms of real extensions and in terms of different parameter values or initial values, can lead to very different system behaviour, for example in terms of its stability properties, as is typical for this kind of self-organised collective systems. Such phenomena are also a reminder of the fact that it is important to analyse collective systems using several complementary approaches.

Complementary analysis techniques The mean-field approximation on which the FlyFast model-checker is based is implicitly assuming that the populations involved are large enough and sufficiently 'well-mixed' such that the average behaviour of the system provides a good approximation of its real behaviour. However, real systems may be such that one cannot always safely abstract completely from stochastic variations in its behaviour. Such variations could either be included explicitly into the model, for example by using models based on stochastic differential equations, or the analysis could be complemented by for example techniques based on stochastic simulation such as statistical or approximate model-checking [38, 65, 34]. The latter techniques may be computationally costly when the models of the individuals are relatively large and many simulation traces are needed to reach a sufficient level of accuracy. Therefore it would be convenient to use these techniques only to investigate those variants of a model that resulted particularly promising when analysed with the more efficient mean-field based methods. To perform such a step-wise analysis and design methodology different analysis tools and an appropriate framework and tool-support would be needed. Ideally, such a framework should also guide designers on which steps to take and in which order, depending on the type of model at hand and the properties and level of abstraction of interest.

Properties Collective self-organising systems have a range of important properties that one would like to analyse. The PCTL based property language used in this paper to express properties of an individual in the context of the overall system behaviour is only covering some of the properties of interest. There are several possibilities to extend this set. As a first step we have added simple global system properties to PCTL, but there are many other extensions possible. Among those are properties involving rewards and costs, properties that address spatial location [22] or even distribution patterns, properties that involve several different individuals and perhaps properties that address certain stability aspects of systems [7].

Error estimates A further important issue are the conditions under which deterministic approximation of collective models provide reliable results. As is well-known, mean field approximation gives an approximation of the average behaviour of the system and makes a number of important assumptions under which the method provides reliable results. In particular possible variance and higher moments of the system are not considered, and it is assumed that the behaviour of an individual, or a small group of individuals, cannot affect the overall system behaviour in any sudden and relevant way. Furthermore, if the solutions in discrete time are interpreted as some form of approximation of solutions in the continuous setting, further error estimates need to be taken into consideration.

Case studies Finally, the development of suitable case studies is essential for several reasons. First of all they play an important role in the validation of the techniques and to illustrate the potential and the limitations of the techniques. Furthermore, they may serve as a source of inspiration for the development of additional verification techniques and as examples of modelling and verification in this relatively new field of application. Finally, they play an important role for educational purposes and to reach out to professional designers of man-made collective self-organising systems in the foreseeable future.

8 Conclusions and Future Work

In this paper we have presented a fast PCTL model-checking approach that builds upon *local, on-the-fly* model-checking and *mean-field* approximation, allowing for scalable analysis of selected objects in the context of collective self-organising systems. The model-checking algorithm is parametric w.r.t. the specific semantic model of interest. We presented related correctness results, and a number of examples of application to collective self-organising systems of a prototype implementation of the FlyFast model-checker. We also briefly discussed the complexity of the model-checking algorithm.

There are several lines of future work of our interest. First of all, the results obtained in this paper can be trivially extended in order to consider multiple selected objects. Following approaches similar to those presented in [47], we plan to extend the model-checking technique to heterogeneous systems and systems with memory. Furthermore, we are interested in developing a probabilistic process algebraic population modelling language that addresses explicit process interaction via synchronisation and that thus supports a compositional modelling approach. We are also interested in extensions that address spatial distribution of objects as well as more expressive logics, combining local and global properties, including unbounded modalities, and languages (e.g. [53, 43]) and to study the exact relation between mean field convergence results for continuous interleaving models and discrete, time-synchronous ones.

9 Acknowledgments

This research has been partially funded by the EU projects ASCENS (nr. 257414) and QUANTICOL (nr. 600708), and the IT MIUR project CINA.

References

[1] Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model checking Continuous Time Markov Chains. ACM Transactions on Computational Logic 1(1), 162–170 (2000)

[2] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-Checking Algorithms for Continuous-Time Markov Chains. IEEE Transactions on Software Engineering. IEEE CS 29(6), 524–541 (2003)

[3] Baier, C., Katoen, J.P., Hermanns, H.: Approximate Symbolic Model Checking of Continuous-Time Markov Chains. In: Baeten, J., Mauw, S. (eds.) Concur '99. LNCS, vol. 1664, pp. 146–162. Springer-Verlag (1999)

[4] Bakhshi, R., Endrullis, J., Endrullis, S., Fokkink, W., Haverkort, B.: Automating the mean-field method for large dynamic gossip networks. In: QEST 2010. pp. 241–250. IEEE Computer Society (2010)

[5] Benaïm, M., Le Boudec, J.Y.: A class of mean field interaction models for computer and communication systems. Performance Evaluation 65(11-12), 823–838 (2008)

[6] Bhat, G., Cleaveland, R., Grumberg, O.: Efficient on-the-fly model checking for CTL*. In: LICS. pp. 388–397. IEEE Computer Society (1995)

[7] Bortolussi, L., Latella, D., Massink, M.: Stochastic Process Algebra and Stability Analysis of Collective Systems. In: De Nicola, R., Julien, C. (eds.) Coordination Models and Languages. LNCS, vol. 7890, pp. 1–15. Springer-Verlag (2013), DOI: 10.1007/978-3-642-38493-6_1, ISSN: 0302-9743, ISBN: 978-3-642-38492-9 (print), 978-3-642-38493-6 (on line)

[8] Bortolussi, L., Hillston, J.: Fluid model checking. In: Koutny, M., Ulidowski, I. (eds.) CONCUR. LNCS, vol. 7454, pp. 333–347. Springer-Verlag (2012)

[9] Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective system behaviour: A tutorial. Performance Evaluation 70(5), 317 – 349 (2013), `http://www.sciencedirect.com/science/article/pii/S0166531613000023`

[10] Bradley, J.T., Gilmore, S.T., Hillston, J.: Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. J. Comput. Syst. Sci. 74(6), 1013–1032 (2008)

[11] Brinksma, E., Hermanns, H.: Process algebra and Markov chains. In: Lectures on Formal Methods and Performance Analysis. pp. 181–231. LNCS 2090, Springer (2001)

[12] Buchli, J., Santini, C.: Complexity engineering, harnessing emergent phenomena as opportunities for engineering (2005), reports of the Santa Fe Institute's Complex Systems Summer School 2005

[13] Camazine, S., Deneubourg, J., Franks, N., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ (2002)

[14] Caragliu, A., Del Bo, C., Nijkamp, P.: Smart cities in Europe. Journal of Urban Technology 18, 65–82 (2011), `http://ideas.repec.org/p/dgr/vuarem/2009-48.html`

[15] Chaintreau, A., Le Boudec, J.Y., Ristanovic, N.: The age of gossip: spatial mean field regime. In: Douceur, J.R., Greenberg, A.G., Bonald, T., Nieh, J. (eds.) SIGMETRICS/Performance. pp. 109–120. ACM (2009)

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

[17] Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8(2), 244–263 (1986)

[18] Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory-efficient algorithms for the verification of temporal properties. Form. Methods Syst. Des. 1(2-3), 275–288 (1992)

[19] Darling, R., Norris, J.: Differential equation approximations for Markov chains. Probability Surveys 5, 37–79 (2008)

[20] Dayar, T., Mikeev, L., Wolf, V.: On the numerical analysis of stochastic lotka-volterra models. In: IMCSIT. pp. 289–296 (2010)

[21] De Maio, P.: Bike-sharing: Its history, impacts, models of provision, and future. Journal of Public Transportation 12(4), 41–56 (2009)

[22] De Nicola, R., Katoen, J.P., Latella, D., Loreti, M., Massink, M.: Model checking mobile stochastic logic. Theor. Comput. Sci. 382(1), 42–70 (2007)

[23] Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., Zilli, M.V.: Bounded probabilistic model checking with the mur*alpha* verifier. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 214–229. Springer (2004)

[24] Din, Q.: Dynamics of a discrete lotka-volterra model. Advances in Difference Equations 95, 1–13 (2013)

[25] Dorigo, M., Stützle, T.: Ant colony optimization. MIT Press (2004)

[26] Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A formal analysis of Bluetooth device discovery. Int. Journal on Software Tools for Technology Transfer 8(6), 621–632 (2006)

[27] Ericsson: Network Society City Index: Triple-bottom-line effects of accelerated ICT maturity in cities worldwide. Tech. rep., Ericsson (2011), `http://www.ericsson.com/networkedsociety/media/hosting/City_Index_Report.pdf`

[28] Frei, R., Di Marzo Serugendo, G.: Concepts in complexity engineering. International Journal of Bio-Inspired Computation 3(2), 123–139 (2011)

[29] Fricker, C., Gast, N.: Incentives and redistribution in bike-sharing systems (2013), `http://arxiv.org/abs/1201.1178`, online; accessed 17-September-2013]

[30] Fricker, C., Gast, N., Mohamed, H.: Mean field analysis for inhomogeneous bike sharing systems. International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA 2012) (2012)

[31] Gast, N., Gaujal, B.: A mean field model of work stealing in large-scale systems. In: Misra, V., Barford, P., Squillante, M.S. (eds.) SIGMETRICS. pp. 13–24. ACM (2010)

[32] Gnesi, S., Mazzanti, F.: An abstract, on the fly framework for the verification of service-oriented systems. In: Wirsing, M., Hölzl, M.M. (eds.) Results of the SENSORIA Project, LNCS, vol. 6582, pp. 390–407. Springer (2011)

[33] Goel, N.S., Maitra, S.C., Montroll, E.W.: On the volterra and other nonlinear models of interacting populations. Rev. Mod. Phys. 43, 231–276 (Apr 1971), `http://link.aps.org/doi/10.1103/RevModPhys.43.231`

[34] Guirado, G., Hérault, T., Lassaigne, R., Peyronnet, S.: Distribution, approximation and probabilistic model checking. In: PDMC 2005. LNCS, vol. 135. pp. 19–30. Springer (2006)

[35] Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: INFAMY: An infinite-state markov model checker. In: CAV09, LNCS, vol. 5643. pp. 641–64. Springer (2009)

[36] Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994)

[37] Hayden, R.: Scalable performance analysis of massively parallel stochastic systems (2011), phD Thesis, `http://aesop.doc.ic.ac.uk/pubs/hayden-thesis/`

[38] Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: VMCAI04. LNCS, vol. 2937. pp. 73–84. Springer (2004)

[39] Hermanns, H., Herzog, U., Katoen, J.: Process algebra for performance evaluation. Theoretical Computer Science 274, 43–87 (2002)

[40] Hillston, J.: Fluid flow approximation of PEPA models. In: Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST 2005). pp. 33–43. IEEE (2005)

[41] Holzmann, G.J.: The SPIN Model Checker - primer and reference manual. Addison-Wesley (2004)

[42] Kolesnichenko, A., Remke, A., de Boer, P.T.: A logic for model-checking of mean-field models. Technical Report TR-CTIT-12-11, http://doc.utwente.nl/80267/ (2012)

[43] Kolesnichenko, A., Remke, A., de Boer, P.T.: A logic for model-checking of mean-field models. In: DSN13 (2013)

[44] Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic Symbolic Model Checking using PRISM: A Hybrid Approach. STTT 6(2), 128–142 (2004)

[45] Latella, D., Loreti, M., Massink, M.: On-the-fly mean field model-checking. In: Proceedings of the International Conference on Trustworthy Global Computing (TGC 2013). Springer (2013), to appear as LNCS volume.

[46] Latella, D., Loreti, M., Massink, M.: On-the-fly Probabilistic Model-Checking - Full Version. Technical report (2013), `http://rap.dsi.unifi.it/$\sim$loreti/OFMFMC/`

[47] Le Boudec, J.Y., McDonald, D., Mundinger, J.: A generic mean field convergence result for systems of interacting objects. In: QEST07. pp. 3–18. IEEE Computer Society Press (2007), iSBN 978-0-7695-2883-0

[48] Lotka, A.J.: Elements of Mathematical Biology. Williams and Wilkins Company (1924)

[49] Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications: The tota approach. ACM Trans. Softw. Eng. Methodol. 18(4) (2009)

[50] Massink, M., Brambilla, M., Latella, D., Dorigo, M., Birattari, M.: On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics. Swarm Intelligence. Springer 7(2-3), 201–228 (2013), `http://link.springer.com/article/10.1007%2Fs11721-013-0079-6`, dOI 10.1007/s11721-013-0079-6; ISSN 1935-3820 (on line), ISSN 1935-3812 (print)

[51] Massink, M., Latella, D.: Fluid Analysis of Foraging Ants. In: Sirjani, M. (ed.) Coordination Models and Languages. LNCS, vol. 7274, pp. 152–165. Springer-Verlag (2012), DOI: 10.1007/978-3-642-30829-1_11, ISSN: 0302-9743, ISBN: 978-3-642-30828-4

[52] Massink, M., Latella, D., Bracciali, A., Hillston, J.: Modelling non-linear crowd dynamics in Bio-PEPA. In: Giannakopoulou, D., Orejas, F. (eds.) Proceedings of the 14th International Conference on Fundamental Approaches to Software Engineering (FASE 2011). pp. 96–110. LNCS 6603, Springer (2011)

[53] McCaig, C., Norman, R., Shankland, C.: From individuals to populations: A mean field semantics for process algebra. Theor. Comput. Sci. 412(17), 1557–1580 (2011)

[54] Mitzenmacher, M.: The power of two choices in randomized load balancing. IEEE Trans. Parallel Distrib. Syst. 12(10), 1094–1104 (2001)

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

[56] Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., Morris, R.: Smarter cities and their innovation challenges. Computer 44, 32–39 (2011)

[57] Norman, G., Parker, D., Kwiatkowska, M., Shukla, S., Gupta, R.: Using probabilistic model checking for dynamic power management. Formal Aspects of Computing 17, 160–176 (2005)

[58] Omicini, A.: Nature-inspired coordination models: Current status and future trends. ISRN Software Engineering 2013 (2013), article ID 384903, 13 pages.

[59] Seeley, T., Visscher, P., Schlegel, T., Hogan, P., Franks, N., Marshall, J.: Stop signals provide cross inhibition in collective decision making by honey bee swarms. Science 335, 108 – 111 (2012)

[60] Stefanek, A., Hayden, R.A., Bradley, J.T.: A new tool for the performance analysis of massively parallel computer systems. In: QAPL 2010. EPTCS, vol. 28. pp. 159–181 (2010)

[61] Tribastone, M., Gilmore, S., Hillston, J.: Scalable differential analysis of process algebra models. IEEE Transactions on Software Engineering 38, 205–219 (2012)

[62] Viroli, M., Beal, J., Usbeck, K.: Operational semantics of PROTO. Sci. Comput. Program. 78(6), 633–656 (2013)

[63] Volterra, V.: Fluctuations in the abundance of a species considered mathematically. Nature 118, 558 – 560 (1926)

[64] Wikipedia: Bicycle sharing system — wikipedia, the free encyclopedia (2013), `http://en.wikipedia.org/w/index.php?title=Bicycle_sharing_system&oldid=573165089`, [Online; accessed 17-September-2013]

[65] Younes, H., Kwiatkowska, M., Norman, G., Parker, D.: Numerical vs. statistical probabilistic model checking: An empirical study. In: Jensen, K., Podelski, A. (eds.) Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04). LNCS, vol. 2988, pp. 46–60. Springer (2004)

A Proof of Lemma 2

Lemma 2. For all $N > 0$, states $\mathbf{C}^{(N)}$ and formulas Φ the following holds:
$\mathbf{C}^{(N)} \models_{\mathcal{D}^{(N)}} \Phi$ iff $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$. •

Proof. By induction on Φ; in the proof we write $\mathbf{C}$ instead of $\mathbf{C}^{(N)}$ for the sake of readability.

Case a:
$\mathbf{C} \models_{\mathcal{D}^{(N)}} a$ if and only if $a \in \ell_1(C_{[1]}) \cup \ell_g(\mathbf{C})$, by definition of $\models_{\mathcal{D}^{(N)}}$.
$\mathcal{H}^{(N)}(\mathbf{C}) = \langle \mathbf{C}_{[1]}, \mathbf{M}^{(N)}(\mathbf{C})\rangle$, by definition of $\mathcal{H}^{(N)}$.
Clearly, if $a \in \ell_1(C_{[1]})$, then $\mathbf{C} \models_{\mathcal{D}^{(N)}} a$ if and only if $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} a$, by definition of $\models_{\mathcal{HD}^{(N)}}$. If $a \in \ell_g(\mathbf{C})$, then $\mathbf{C} \models_{\mathcal{D}^{(N)}} a$ if and only if $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} a$, by definition of $\models_{\mathcal{HD}^{(N)}}$ and of $\mathscr{E}[\![\mathrm{bexp}_a]\!]$.

Case $\neg\Phi$:
$\mathbf{C} \models_{\mathcal{D}^{(N)}} \neg\Phi$ if and only if not $\mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi$, by definition of $\models_{\mathcal{D}^{(N)}}$. Thus, not $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi$, by the induction hypothesis, and $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \neg\Phi$, by definition of $\models_{\mathcal{H}^{(N)}}$.

Case $\Phi_1 \vee \Phi_2$:
$\mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_1 \vee \Phi_2$ if and only if $\mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_1$ or $\mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_2$, by definition of $\models_{\mathcal{D}^{(N)}}$. Thus, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ or $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$, by the induction hypothesis, and $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \vee \Phi_2$, by definition of $\models_{\mathcal{HD}^{(N)}}$.

Case $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$:

$$
\begin{array}{ll}
 & \mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \mathsf{next}\,\Phi\} \\
= & \{\text{Def. } \rho \models_{\mathcal{HD}^{(N)}} \mathcal{X}\ \Phi\} \\
 & \sum_{\langle c',\mathbf{m}'\rangle : \langle c',\mathbf{m}'\rangle \models_{\mathcal{HD}^{(N)}} \Phi} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\langle c',\mathbf{m}'\rangle} \\
= & \{\text{Def. } \mathbf{H}^{(N)}\} \\
 & \sum_{\langle c',\mathbf{m}'\rangle : \langle c',\mathbf{m}'\rangle \models_{\mathcal{HD}^{(N)}} \Phi} \sum_{\mathbf{C}' : \mathcal{H}^{(N)}(\mathbf{C}') = \langle c',\mathbf{m}'\rangle} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \\
= & \{\text{I.H., i.e. } \mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi \text{ iff } \mathbf{C}' \models_{\mathcal{D}^{(N)}} \Phi\}
\end{array}
$$

$\sum_{\mathbf{C}':\mathbf{C}'\models_{\mathcal{D}^{(N)}}\Phi} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'}$

$=$ $\{$Def. $\sigma \models_{\mathcal{D}^{(N)}} \mathcal{X}\,\Phi\}$

$\mathbb{P}\{\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}) \mid \sigma \models_{\mathcal{D}^{(N)}} \mathsf{next}\,\Phi\}$

Case $\mathcal{P}_{\bowtie p}(\Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2)$:
We prove the assert by (nested) induction on k.

Base case $(k = 0)$:

$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq 0}\, \Phi_2\}$

$=$ $\{$Def. of $\rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$

$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho[0] \models_{\mathcal{HD}^{(N)}} \Phi_2\}$

$=$ $\{$Def. of $\mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C}))$ and $\rho[0]\}$

$$\begin{cases} 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2, \\ 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \end{cases}$$

$=$ $\{$I. H. on logic formulas$\}$

$$\begin{cases} 1, \text{ if } \mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_2, \\ 0, \text{ if not } \mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_2 \end{cases}$$

$=$ $\{$Def. of $\mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C})$ and $\sigma[0]\}$

$\mathbb{P}\{\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}) \mid \sigma[0] \models_{\mathcal{D}^{(N)}} \Phi_2\}$

$=$ $\{$Def. of $\sigma \models_{\mathcal{D}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$

$\mathbb{P}\{\sigma \in \mathit{Paths}_{\mathcal{D}^{(N)}}(\mathbf{C}) \mid \sigma \models_{\mathcal{D}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq 0}\, \Phi_2\}$

Induction step:

$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k+1}\, \Phi_2\}$

$=$ $\{$Def. $\rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k+1}\, \Phi_2\}$

$$\begin{cases} 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \text{ and not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2, \\ \sum_{\rho[1]:\rho[1]\models_{\mathcal{HD}^{(N)}}\Phi_1} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\rho[1]} \cdot \mathbb{P}\{\rho' \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\rho[1]) \mid \rho' \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\} \end{cases}$$

$=$ $\{$Def. $\mathbf{H}^{(N)}\}$

$$\begin{cases} 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \text{ and not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2, \\ 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ \sum_{\rho[1]:\rho[1]\models_{\mathcal{HD}^{(N)}}\Phi_1} \\ \quad \sum_{\mathbf{C}':\mathcal{H}^{(N)}(\mathbf{C}')=\rho[1]} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \cdot \mathbb{P}\{\rho' \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C}')) \mid \rho' \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\} \end{cases}$$

$$
\begin{aligned}
= \quad & \{\text{I.H. on } k\} \\
& \begin{cases}
0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \text{ and not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2, \\
1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\
\sum_{\rho[1]:\rho[1]\models_{\mathcal{HD}^{(N)}}\Phi_1} \\
\quad \sum_{\mathbf{C}':\mathcal{H}^{(N)}(\mathbf{C}')=\rho[1]} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \cdot \mathbb{P}\{\sigma' \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}') \mid \sigma' \models_{\mathcal{D}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\}
\end{cases} \\
= \quad & \{\text{I.H. on logic formulas}\} \\
& \begin{cases}
0, \text{ if not } \mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_1 \text{ and not } \mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_2 \\
1, \text{ if } \mathbf{C} \models_{\mathcal{D}^{(N)}} \Phi_2 \\
\sum_{\mathbf{C}':\mathbf{C}'\models_{\mathcal{D}^{(N)}}\Phi_1} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \cdot \mathbb{P}\{\sigma' \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}') \mid \sigma' \models_{\mathcal{D}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\}
\end{cases} \\
= \quad & \{\text{Def. } \sigma \models_{\mathcal{D}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k+1} \, \Phi_2\} \\
& \mathbb{P}\{\sigma \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}) \mid \sigma \models_{\mathcal{D}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k+1} \, \Phi_2\}
\end{aligned}
$$

□

□

B Proof of Theorem 4

Theorem 4. Under the assumptions of Theorem 4.1 of [47], for all safe formulas Φ, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \in L_{\mathcal{HD}^{(N)}}(t)$, almost surely, for N large enough, $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$ iff $\langle \mathbf{C}^{(N)}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi$. •

Proof. The proof is carried out by induction on Φ; in the proof we write $\mathbf{C}$ instead of $\mathbf{C}^{(N)}$ for the sake of readability.

Case a:
The assert follows directly from the definitions of $\left(\mathcal{HD}^{(N)}(t)\right)_{N \geq N_0}$, $\models_{\mathcal{HD}^{(N)}}$, and $\models_{\mathcal{HD}}$ (see also Remark 1).

Case $\neg\Phi$:
The I. H. ensures that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., there exists $\bar{N}$ s.t. for all $N \geq \bar{N}$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi$. But $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi$ is logically equivalent to

$$\text{not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi \text{ iff not } \langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi.$$

Thus, by definition of $\models_{\mathcal{HD}^{(N)}}$ and $\models_{\mathcal{HD}}$, we get that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., there exists $\bar{N}$ s.t. for all $N \geq \bar{N}$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \neg\Phi$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \neg\Phi$.

Case $\Phi_1 \vee \Phi_2$:
The I. H. ensures that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., there exists $\bar{N}_1$ s.t. for all $N \geq \bar{N}_1$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_1$, and a.s., there exists $\bar{N}_2$ s.t. for all $N \geq \bar{N}_2$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_2$.
Let us now suppose $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \vee \Phi_2$ holds, i.e. $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ holds or $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ holds, by definition of $\models_{\mathcal{HD}^{(N)}}$. Suppose $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ holds and, by the I.H., we know that, a.s. there exists $\bar{N}_1$ s.t. for all $N \geq \bar{N}_1$ $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_1$ holds as well. But then, we

get that, for all such N, also $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_1 \vee \Phi_2$ holds, by definition of $\models_{\mathcal{HD}}$. If, instead $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ holds, we get the same result, using $\bar{N}_2$ instead of $\bar{N}_1$. Thus, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s. there exists $N \geq \max\{\bar{N}_1, \bar{N}_2\}$ such that if $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \vee \Phi_2$ holds, then $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_1 \vee \Phi_2$ holds.
The proof for the reverse implication is similar.

Case $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$:
By definition of $\models_{\mathcal{HD}^{(N)}}$ and $\models_{\mathcal{HD}}$, we have to show that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., for N large enough,

$$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \mathcal{X}\ \Phi\} \bowtie p$$

iff

$$\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \mid \eta \models_{\mathcal{HD}} \mathcal{X}\ \Phi\} \bowtie p.$$

Below, we actually prove that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., for N large enough, the probabilities of the two sets of paths are approaching each other, which, together with formula safety, implies the assert.
$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \mathcal{X}\ \Phi\}$ is defined as

$$p_{\mathbf{H}}^{(N)} = \sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}), \mathcal{H}^{(N)}(\mathbf{C}')} \tag{8}$$

and $\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \mid \eta \models_{\mathcal{HD}} \mathcal{X}\ \Phi\}$ is defined as

$$p(t)_{\mathbf{K}} = \sum_{\mathbf{C}'_{[1]}:\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1) \rangle \models_{\mathcal{HD}} \Phi} \mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]}, \mathbf{C}'_{[1]}}. \tag{9}$$

The I.H. ensures that, a.s., for $N \geq \bar{N}_{\mathbf{C}'}$, $\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi$ if and only if $\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1) \rangle \models_{\mathcal{HD}} \Phi$, with $\mathcal{H}^{(N)}(\mathbf{C}') \in L_{\mathcal{HD}^{(N)}}(t+1)$. In particular, it holds that, for any specific value $\bar{c}$ of $\mathbf{C}'_{[1]}$ above and $\mathcal{H}^{(N)}(\mathbf{C}') \in L_{\mathcal{HD}^{(N)}}(t+1, \bar{c})$, $\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi$ if and only if $\langle \bar{c}, \boldsymbol{\mu}(t+1) \rangle \models_{\mathcal{HD}} \Phi$, that is: either *all* elements of $L_{\mathcal{HD}^{(N)}}(t+1, \bar{c})$ satisfy Φ or *none* of them does it. Furthermore, for such $\bar{c}$, by Corollary 3, for all $\epsilon_{\bar{c}} > 0$ there exists $N_{\bar{c}}$ s.t. for all $N \geq N_{\bar{c}}$

$$\left| \left(\sum_{\langle \bar{c}, \overline{\mathbf{m}} \rangle : L_{\mathcal{HD}^{(N)}}(t+1, \bar{c})} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}), \langle \bar{c}, \overline{\mathbf{m}} \rangle} \right) - \mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]}, \bar{c}} \right| < \epsilon_{\bar{c}}$$

(see Remark 2). So, for any $\epsilon > 0$ there exists an $\hat{N}$ larger than any of such $\bar{N}_{\mathbf{C}'}$ and $N_{\bar{c}}$, such that for all $N \geq \hat{N}$ $\left| p_{\mathbf{H}}^{(N)} - p(t)_{\mathbf{K}} \right| < \epsilon$ i.e. the value $p_{\mathbf{H}}^{(N)}$ of sum (8) approaches the value $p(t)_{\mathbf{K}}$ of sum (9). Finally, safety of $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$, implies that the value $p(t)_{\mathbf{K}}$ of (9) is different from p. If $p(t)_{\mathbf{K}} > p$ then we can choose ϵ small enough that also $p_{\mathbf{H}}^{(N)} > p$ and, similarly, if $p(t)_{\mathbf{K}} < p$, we get also $p_{\mathbf{H}}^{(N)} < p$, which proves the assert.

Case $\mathcal{P}_{\bowtie p}(\Phi_1 \mathcal{U}^{\leq k} \Phi_2)$:
By definition of $\models_{\mathcal{HD}^{(N)}}$ and $\models_{\mathcal{HD}}$, we have to show that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., for N large enough,

$$\mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \mathcal{U}^{\leq k} \Phi_2\} \bowtie p$$

iff

$$\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \mid \eta \models_{\mathcal{HD}} \Phi_1 \mathcal{U}^{\leq k} \Phi_2\} \bowtie p.$$

Below, we actually prove that, for any fixed t and $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, a.s., for N large enough, the probabilities of the two sets of paths are approaching each other, which implies the assert. We proceed by induction on k, using also the induction hypothesis on the structure of the formulas, when necessary.

Base case ($k = 0$):

$$
\begin{array}{ll}
& \mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq 0} \, \Phi_2\} \\
= & \{\text{Def. of } \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\} \\
& \mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho[0] \models_{\mathcal{HD}^{(N)}} \Phi_2\} \\
= & \{\text{Def. of } \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \text{ and } \rho[0]\} \\
& \begin{cases} 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2, \\ 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \end{cases}
\end{array}
$$

By the I.H. on Φ_2, with $\mathcal{H}^{(N)}(\mathbf{C}) \in L_{\mathcal{HD}^{(N)}}(t)$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_2$, i.e., a.s.

$$
\begin{array}{ll}
& \mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho[0] \models_{\mathcal{HD}^{(N)}} \Phi_2\} \\
= & \{\text{See above}\} \\
& \begin{cases} 1, \text{ if } \langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_2, \\ 0, \text{ if not } \langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle \models_{\mathcal{HD}} \Phi_2 \end{cases} \\
= & \{\text{Def. of } \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \text{ and } \eta[0]\} \\
& \mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \mid \eta[0] \models_{\mathcal{HD}} \Phi_2\} \\
= & \{\text{Def. of } \eta \models_{\mathcal{HD}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\} \\
& \mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t) \rangle) \mid \eta \models_{\mathcal{HD}} \Phi_1 \, \mathcal{U}^{\leq 0} \, \Phi_2\}
\end{array}
$$

Induction step:

$$
\begin{array}{ll}
& \mathbb{P}\{\rho \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k+1} \, \Phi_2\} \\
= & \{\text{Def. } \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k+1} \, \Phi_2\} \\
& \begin{cases} 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \text{ and not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ \sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi_1} \\ \qquad \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\mathcal{H}^{(N)}(\mathbf{C}')} \cdot \mathbb{P}\{\rho' \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C}')) \mid \rho' \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\}, \\ \qquad \text{otherwise.} \end{cases}
\end{array}
$$

By the I.H. on k, noting that we are concerned only with those $\mathcal{H}^{(N)}(\mathbf{C}')$ belonging to $L_{\mathcal{HD}^{(N)}}(t+1)$, a.s., there is $\bar{N}$ s.t. for all $N \geq \bar{N}$,

$\mathbb{P}\{\rho' \in \mathit{Paths}_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C}')) \mid \rho' \models_{\mathcal{HD}^{(N)}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\}$ approaches
$\mathbb{P}\{\eta' \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1) \rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \, \mathcal{U}^{\leq k} \, \Phi_2\}$

Thus,

$\mathbb{P}\{\rho \in Paths_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \Phi_1 \,\mathcal{U}^{\leq k+1}\, \Phi_2\}$

$=$ {See above}

$$\begin{cases} 0, \text{ if not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1 \text{ and not } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ 1, \text{ if } \mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2 \\ \sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi_1} \\ \qquad \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\mathcal{H}^{(N)}(\mathbf{C}')} \cdot \mathbb{P}\{\eta' \in Paths_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}, \\ \qquad \text{otherwise.} \end{cases}$$

The I.H. ensures that, a.s., there exist N_1, N_2 and a set of values $N_{\mathbf{C}'}$, for $\mathbf{C}'$ as in the sum above, s.t.

- for all $N \geq N_1$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_1$
- for all $N \geq N_2$, $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_2$
- for all $N \geq N_{\mathbf{C}'}$, $\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi_1$ iff $\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle \models_{\mathcal{HD}} \Phi$.

Furthermore, by Corollary 3, using similar arguments as those used for the case $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$, we get that a.s. there exists $\hat{N}$ such that, for $N \geq \hat{N}$,
$\sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi_1}$
$\mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\mathcal{H}^{(N)}(\mathbf{C}')} \cdot \mathbb{P}\{\eta' \in Paths_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$

approaches

$\sum_{\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle \models_{\mathcal{HD}} \Phi_1}$
$\mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]},\mathbf{C}'_{[1]}} \cdot \mathbb{P}\{\eta' \in Paths_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$. Thus, a.s. for $N \geq \max\{N_1, N_2, \bar{N}_{\mathbf{C}'}, \hat{N}\}$, with $\hat{N}, \bar{N}_{\mathbf{C}'} \geq N\mathbf{C}'$ for $\mathbf{C}'$ as above the following holds:

- not $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_1$ and not $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ iff
  not $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_1$ and not $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_2$
- $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \Phi_2$ iff $\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_2$
- $\sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi_1}$
  $\mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\mathcal{H}^{(N)}(\mathbf{C}')} \cdot \mathbb{P}\{\eta' \in Paths_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$

  approaches

  $\sum_{\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle \models_{\mathcal{HD}} \Phi_1}$
  $\mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]},\mathbf{C}'_{[1]}} \cdot \mathbb{P}\{\eta' \in Paths_{\mathcal{HD}}(\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle) \mid \eta' \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2\}$

and by safety of $\mathcal{P}_{\bowtie p}(\Phi_1 \,\mathcal{U}^{\leq k+1}\, \Phi_2)$ we get the assert. $\square$ $\square$