On-the-fly PCTL Fast Mean-Field Approximated Model-Checking for Self-organising Coordination[☆]

D. Latella[—a,*], M. Loreti[b,c,*], M. Massink[a,*]

[a] *Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy*
[b] *Dip. di Statistica, Informatica, Applicazioni 'G. Parenti', Università di Firenze, Italy*
[c] *IMT Advanced Studies Lucca, Italy*

Abstract

Typical self-organising collective systems consist of a large number of interacting objects that coordinate their activities in a decentralised and often implicit way. Design of such systems is challenging and requires suitable, scalable analysis tools to check properties of proposed system designs before they are put into operation. We present a novel scalable, on-the-fly approximated model-checking procedure to verify bounded PCTL properties of selected individuals in the context of very large systems of independent interacting objects. The proposed procedure combines on-the-fly model-checking techniques with deterministic mean-field approximation in discrete time. The asymptotic correctness of the procedure is proven and a prototype implementation of the model-checker is presented. The potential of the verification approach is illustrated by its application on self-organising collective systems and an overview of remaining open issues and future extensions is provided.

Keywords: Probabilistic Model-Checking, On-the-fly Model-Checking, Mean-Field Approximation, Discrete Time Markov Chains, Self-organisation

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 [1, 2]. These desirable features of self-organisation, among others, have recently attracted the interest of researchers in the field of engineering (see e.g. [3] and more recently e.g. [4, 5, 1, 6, 7]). There are numerous examples of man-made systems, existing or planned, that incorporate some forms of self-organisation, for the better or the worse. For instance, we can consider the development of fully decentralised smart electric grids [8, 9], with many local

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

[*] Corresponding author

producers and consumers, or smart transportation systems, that provide real-time information to travellers and adapt to changing requests; examples are public transportation [10] and shared bike systems [11, 12]. Often the humans involved in such systems are not simply 'end-users'; they are intrinsic and autonomous interacting elements of the system, constantly receiving feedback and providing input. This is for instance the case for a shared-bike system, when a human decides where to take a bike and which is the most convenient destination where to leave it; this may depend on the time of the day, personal needs, potential incentives and the actual situation of parking lots.

The pervasive nature of engineered self-organised systems, along with the increasing critical dependence of people on their continuous reliable operation, implies that it is extremely important to develop reliable rigorous design models as well as a priori analysis techniques of such models—covering all relevant aspects of their behaviour, including quantitative and emergent ones—before they are put into operation.

The development of formal system modelling and model analysis techniques is a very active field of research. It covers very diverse approaches, among which process algebra, temporal logics and related model-checking techniques, model-based simulation techniques, and their stochastic extensions to mention a few. In this article we propose and explore a new analysis technique that *combines* on-the-fly model-checking and mean-field approximation techniques [13]. In particular, we present the design, implementation and application of the model-checking procedure. The procedure can be used to verify bounded PCTL (Probabilistic Computation Tree Logic) [14] 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 can be used with huge population sizes, typical of analysis techniques based on mean-field approximation. The asymptotic correctness of the on-the-fly approximated 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 epidemic (also discussed in [15]) and public transportation, in particular bike sharing [11]. Following the approach proposed in [13] 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 vector* at each time step t—i.e. the vector where each element gives the fraction of objects that are in a particular local state at that time—can be approximated by the (deterministic) solution $\boldsymbol{\mu}(t)$ of a *difference equation* which is called the 'mean-field'[2]. This convergence result has been extended in [13] 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 [13] to compute the occupancy measure vector combines well with an *on-the-fly* probabilistic model-checking procedure for the verifi-

[1]These models are also known as SIO-models (System of Independent Objects) [15]. 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 [13], i.e. a deterministic approximation of the occupancy measure of a probabilistic population model.

cation of bounded PCTL formulas addressing properties of selected objects of interest, and a limited form of global system properties. An on-the-fly recursive approach also provides a natural way to address nested path formulas and time-varying truth values of such formulas[3].

The main challenge of our work was the extension of the application of the mean-field approach to probabilistic model-checking, i.e. the rigorous definition of the mean-field semantics of a probabilistic population description language as a labelled Discrete Time Markov Chain (DTMC), and, most of all, its use as a model for the bounded PCTL model-checking problem, in such a way that formula satisfaction is preserved. This was obtained in three steps. First, a formal semantics of the probabilistic population description language has been defined, which maps a *model specification* of a system with *population size N* into a labelled DTMC, say $\mathcal{D}^{(N)}$ (where the superscript (N) makes the population size explicit). This is a relatively easy job, given that the model specification language we consider is very simple; it allows the modeller to specify an object essentially as a DTMC and a system as a parallel synchronous composition, i.e. product, of N objects. The formal, detailed definition of the syntax and semantics of the language is just standard language engineering routine and is left out in the present paper; the interested reader can find the details in [16]. A generic state $\mathbf{C}^{(N)}$ of $\mathcal{D}^{(N)}$ is a *global state (vector)*, i.e. a vector of size N, the j-th component of which is the current state of the j-th element of the product.

The second step is the definition of an abstraction of $\mathcal{D}^{(N)}$, say $\mathcal{HD}^{(N)}$, which helps us focusing on the two aspects of global states which we are interested in, namely the current state of a selected object (conventionally, the *first* object of the product) and the occupancy measure of the global state. This is done by means of a mapping $\mathcal{H}^{(N)}$ which maps every global state $\mathbf{C}^{(N)} = \langle C_1, \ldots, C_N \rangle$ into the pair $\langle C_1, \mathbf{m} \rangle$, where $\mathbf{m}$ is the occupancy measure vector of $\mathbf{C}^{(N)}$. Here the main challenge was to show correctness of the abstraction w.r.t. bounded PCTL satisfaction, namely that the abstraction preserves the truth values of bounded PCTL formulas. This can be seen in two different ways: by means of a direct proof by induction of PCTL formulas, or indirectly, by showing that $\mathcal{H}^{(N)}$ induces a probabilistic bisimulation so that, for all states $\mathbf{C}^{(N)}$ of $\mathcal{D}^{(N)}$, $\mathbf{C}^{(N)}$ and $\mathcal{H}^{(N)}(\mathbf{C}^{(N)})$ are bisimilar and, consequently [17], they satisfy the same set of bounded PCTL formulas.

The last step was to define a third DTMC, $\mathcal{HD}$, as the limit of the series of DTMCs $\mathcal{HD}^{(N)}$, for $N \to \infty$, characterised by the fact that there exists a function $\boldsymbol{\mu}$ such that, for large N, all the occupancy measure vectors reachable at any given time step t in $\mathcal{HD}^{(N)}$ (or equivalently in $\mathcal{D}^{(N)}$) can be deterministically approximated by the *single* value $\boldsymbol{\mu}(t)$, and such that, for each state $\mathbf{C}^{(N)}$ of $\mathcal{D}^{(N)}$ reachable in t steps and bounded PCTL formula Φ, $\mathcal{H}^{(N)}(\mathbf{C}^{(N)})$ satisfies Φ in model $\mathcal{HD}^{(N)}$ if and only if $\langle C_1, \boldsymbol{\mu}(t) \rangle$ satisfies Φ in model $\mathcal{HD}$, where, as usual C_1 is the first element of $\mathbf{C}^{(N)}$. Of course, the above logical correspondence immediately extends to $\mathbf{C}^{(N)}$ w.r.t. model $\mathcal{D}^{(N)}$, due to the correspondence result of the previous step. This last step is fundamental for the real scalability of the model-checking algorithm. In fact it is this very step that can drastically reduce the size of the state space, or of the portion of the state space to be visited, in the case of on-the-fly techniques, and relieves the model-checking procedure from the classical state space exponential blow-up, as far as this is a consequence of the population size. Indeed, for each t, the multiplicity of possible occupancy measure vectors stemming from the non-deterministic nature of objects and their composition is dramatically reduced to just *one* (deterministic) value. This makes it possible to efficiently analyse system models with large population sizes (thousands, millions, or more), still using model-checking technology, i.e. without the use of additional and

[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.

often costly technology, such as simulation, as is the case, for instance, with statistical model-checking (see related discussion in Section 2). Notably, the number of equations which define $\boldsymbol{\mu}$, i.e. the "size" of the model representation, is *constant* w.r.t. the number N of objects, and depends (linearly) only on the number S of states of a single object. This, together with the drastic reduction of the system model state space due to the deterministic approximation of the occupancy measure vector, is the key for the essentially unbounded scalability of the method w.r.t. the system population size. Furthermore, comparisons with simulation results show that the technique produces good results also with smaller population sizes (e.g. hundreds or thousands). Existence of $\mathcal{HD}$ and $\boldsymbol{\mu}(t)$ is guaranteed by the fundamental mean-field convergence result of [13]. On the other hand the logical correspondence between $\mathcal{HD}^{(N)}$ and $\mathcal{HD}$ constitutes the main result of our work. The proof, by induction on Φ, requires some insights on the limit behaviour of $\mathcal{HD}^{(N)}$, and in particular the relationship between the probability transition matrix of $\mathcal{HD}^{(N)}$ and that of $\mathcal{HD}$; furthermore, some non-trivial, *safety* restrictions are required on PCTL formulas, in a similar way as in [18] for the continuous time case.

Our model-checking algorithm is parametric w.r.t. the semantic interpretation of the model specification language. In particular, in this paper, we present two different instantiations of the bounded PCTL algorithm; one based on the standard, *exact probabilistic semantics* of the probabilistic population description language, and the other one on the *mean-field approximation in discrete time* of such a semantics. The considered PCTL formulas can be extended along the lines proposed in [19, 20] with properties that address the overall status of the system. We show some simple instances of that as an example. The algorithm presented in the current paper is actually the bounded PCTL fragment of a more general algorithm for (full, exact) PCTL model-checking [21]. A preliminary version of this work has appeared in [16]. The present paper includes detailed correctness proofs and further details on case studies that illustrate the potential and limitations of the approach for the verification of self-organised coordination. In particular, a bike sharing system is modelled and analysed in which the potential effect of user incentives on the overall performance of the system is studied as a self-organising coordination principle [57]. 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 approximated model-checking and provides related correctness results. Section 6 shows the application of the FlyFast model-checker on a case study from the field of self-organised systems. Finally, Section 7 provides some conclusions and a discussion on possible limitations and open issues which require future work.

2. Related Work

In this section we discuss related work on process algebraic modelling languages with stochastic—based on Continuous Time Markov Chains (CTMC)—or probabilistic—DTMC-based—exact semantics, for which a mean-field / fluid-flow approximated semantics is available as well. We will also briefly discuss some probabilistic and stochastic model-checking proposals as well as results on mean-field / fluid-flow analysis. We confine the discussion only to those proposals which are relevant for setting the context for our work.

In the area of *system modelling languages*, process algebras, and in particular their stochastic extensions, have been designed and extensively and successfully used to formally specify

behavioural aspects of concurrent systems, including non-functional, e.g. performance, features [22, 23]. Examples of their application can be found in diverse areas ranging from communication protocols [24] to control theory [25]. A process algebraic design framework is complementary to temporal logics model-checking techniques. Such techniques, and their stochastic extensions (see for instance [26, 27, 28]), 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 [29, 30, 31]. 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 [32, 33, 34, 35]). These extensions are based on stochastic process algebra, i.e. process calculi with an interleaving, *continuous* time semantics, namely CTMCs. The above mentioned languages exploit mean-field or fluid approximation techniques [37, 38]. Such techniques work by approximating the discrete state space of stochastic or probabilistic population models by a continuous one.

To the best of our knowledge, [36] is the only proposal available in the literature for the synchronous, DTMC-based paradigm and for which mean-field semantics have been defined. Although the language presented in [36] enjoys nice compositional features, its definition is based on rather powerful notions, like action priority, which make it relatively complicated. For the purposes of the present paper we prefer to use a simpler, finite automata based, language.

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, commonly known as the 'fluid-flow' approach, the behaviour can be approximated by a (deterministic) solution of a set of *differential* equations, and in the latter, which we will refer to as the 'mean-field' approach, 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. [37] for approximations in continuous time, and [13] for correctness of the approach in discrete time (the interested reader is referred to [15] 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 epidemic 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 [37]) and in a discrete time setting (see for example [13]). In particular, in the discrete time setting, the average system dynamics can be computed by a deterministic iterative procedure [13]. It is this feature that we extensively exploit in the development of our on-the-fly approximated probabilistic mean-field model-checker.

Mean-field approximations in discrete time have also been used in e.g. Bakshi et al. [39] for analysing properties of *large scale* mobile communication networks. 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 [13], but the proposal does not cover the possibility of analysing individual behaviour in the context of a large population, neither its exploitation in model-checking algorithms.

In Chaintreau et al. [40], 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 [41].

In the context of the use of mean-field convergence in continuous time for grouped PEPA (Performance Evaluation Process Algebra) Stefanek et al. [42, 43] 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.

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 [44, 45].

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. [46]) and in a stochastic setting (see e.g. [27, 28]). 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 [27] for stochastic model-checking, the global model-checking algorithm can be reduced to combinations of existing well-known and optimised algorithms for CTMCs such as transient analysis. 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. [44, 45, 47, 48]). 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. 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 [49] and [50]. 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 (Continuous Stochastic Logic) [51, 27] 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 [40], self-organised collective decision making [52], computer epidemic [38] 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 [53, 54] 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.

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 [19], but no model-checking algorithms were presented. Follow-up work on the above mentioned approach can be found in [20] which relies on earlier results on fluid model-checking by Bortolussi and Hillston [18]. 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; the modelling language is PEPA. This work is perhaps closest related to our work; however their procedure exploits mean-field convergence and fast simulation [37, 55] 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 adopted. Consequently, the underlying model-checking algorithms and related correctness proofs are fundamentally different from those presented in this paper. In particular, the treatment of nested formulas, whose truth value may change over time, turns out to be much more difficult in the interleaving, continuous time, global model-checking approach than in the clock-synchronous, discrete time, on-the-fly one.

To the best of our knowledge, ours is the first proposal and implementation of an *on-the-fly mean-field* approximated model-checker for *discrete time, probabilistic, time-synchronous* models.

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

In this section we recall the definition of the *time bounded* fragment[4] of PCTL [14] for which we present an on-the-fly model-checking algorithm. This algorithm is part of a general algorithm for full PCTL model-checking presented in [21]. In the latter, the treatment of *unbounded* until makes use of a novel technique which exploits an interesting property of transient DTMCs, i.e. one in which all recurrent states are absorbing (see [21] for details). The algorithm is parametric in the sense that it can be used for different languages and semantic interpretations.

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

$s \models_{\mathcal{M}} \mathsf{ap}$	iff	$\mathsf{ap} \in \ell(s)$
$s \models_{\mathcal{M}} \neg\Phi$	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	$\mathbb{P}\{\sigma \in \mathit{Paths}_{\mathcal{M}}(s) \mid \sigma \models_{\mathcal{M}} \varphi\} \bowtie p$
$\sigma \models_{\mathcal{M}} \mathcal{X}\,\Phi$	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.

In this paper we use two instantiations of the bounded PCTL fragment of the algorithm; one is on an exact DTMC semantics of a simple language of object populations (Section 4) and the other is on a mean-field approximation semantics of the same language, for "fast model-checking" (Section 5). For the sake of readability, in this paper we present a high level description of the algorithm, which abstracts from implementation details. The reader interested in a detailed description of the algorithm is referred to [21].The instantiation of the algorithm (fragment) on the *exact* DTMC semantics of the language essentially coincides with the algorithm proposed in [49].

3.1. Time bounded PCTL

We first recall that, given a set $\mathscr{P}$ of atomic propositions, the syntax of PCTL *state formulas* Φ and *path formulas* φ is defined as follows, where $\mathsf{ap} \in \mathscr{P}$, $k \geq 0$ and $\bowtie \in \{\geq, >, \leq, <\}$:

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

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 $\mathit{Paths}_{\mathcal{M}}(s)$ 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 initial probability distribution is δ_{s_0}, the Dirac probability distribution function assigning 1 to s_0 and 0 to any other state $s \in \mathcal{S}$. For any such a DTMC $\mathcal{M}$, and for all $t \in \mathbb{N}$ we let the set $L_{\mathcal{M}}(t) = \{\sigma[t] \mid \sigma \in \mathit{Paths}_{\mathcal{M}}(s_0)\}$ be the set of possible states of $\mathcal{M}$ at time t (given that the state of $\mathcal{M}$ at time 0 is s_0).

We recall the bounded PCTL satisfaction relation in Table 1.

3.2. On-the-fly PCTL Model-Checking Algorithm

In this section we introduce the 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.

```
1  Check(  s : proc,  Φ : formula)=
2  match   Φ
3  with
4  | ap → (lab_eval s ap)
5  | ¬Φ1 → ¬Check(s, Φ1)
6  | Φ1 ∨ Φ2 → Check(s, ϕ1) ∨ Check(s, Φ2)
7  | P⟨relop⟩p(φ) → CheckPath(s, φ)⟨relop⟩p
```

Table 2: Function Check

Our 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 ap 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 $\varphi \in$ 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 $\{s' | \exists p' : (s', p') \in \mathsf{next}(s)\}$. 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, which concerns only the algorithm for the time bounded fraction of PCTL, is easily proved by induction on Φ [49, 21]. In [21] also the more involving proofs for full PCTL can be found.

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

```
1  CheckPath( s : proc , φ : path_formula )=
2  match φ with
3  | X Φ → let p = 0.0 and lst = next(s) in
4        for (s′, p′) ∈ lst do if Check(s′, Φ) then p ← p + p′
5        done;
6        p
7  | Φ1 U^≤k Φ2 → if Check(s, Φ2) then 1.0
8          else if Check(s, ¬Φ1) then 0.0
9          else if k > 0 then
10                 begin
11                     let p = 0.0 and lst = next(s) in
12                     for (s′, p′) ∈ lst do
13                         p ← p + p′ ∗ CheckPath(s′, Φ1 U^≤k−1 Φ2)
14                     done;
15                     p
16                 end
17               else 0.0
```

Table 3: Function CheckPath

In Section 4 we show the instantiation of the algorithm on the exact DTMC semantics of a simple probabilistic process population modelling language. When large populations need to be modelled, a mean-field approximated semantics can be defined for the language. In Section 5 we show how a drastic reduction of the state space can be obtained, by using the same algorithm on such mean-field semantic models. We call the combined use of on-the-fly model-checking and mean-field semantics '*Fast* mean-field approximated model-checking' after 'Fast simulation', introduced in [13].

4. Modelling language

In this section we describe a simple probabilistic population description language. The language is essentially a textual version of the graphical notation used in [13] and can be seen as a simplified process algebraic language. For the sake of notational simplicity here we provide an informal description of the language and refer to [16] for its detailed formal definition.

Intuitively, 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.

An *object specification* Δ is a finite set of *state-defining* equations, one for each *state* of the object. We let $\mathcal{S}$, ranged over by $C, C', C_1, \ldots$ denote the (denumerable, non-empty) set of all states (more precisely, state *names*) which can be used in equations. Each equation defines the transitions from the state to other states of the object; each transition is labelled by the *action* the

[5] In [13] *object* is used instead of *process*. We consider the two terms synonyms here.

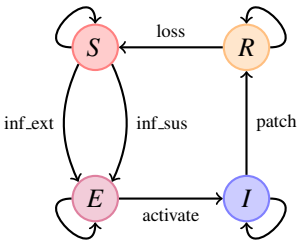


Figure 1: Epidemic model object.

object performs when the transition takes place, where action names $a, a', a_1, \ldots$ are drawn from given (denumerable, non-empty) set $\mathcal{A}$. The general format of a state defining equation is:

$$C := a_1.C_1 + \ldots + a_r.C_r$$

Intuitively, the above notation defines state C of the object at hand and postulates that there are r outgoing transitions from C, with action a_j labelling a transition going from C to C_j. In summary, the r.h.s. of the equation is to be intended as the n-ary extension of the standard process algebraic binary non-deterministic choice operator, the operands of which are action-prefix terms [22, 23]. Obviously, in a given object specification there is exactly one defining equation for each state of the object and, for the sake of simplicity, in this paper we also require that in any equation like the above all actions are distinct. An example of object definition is given below.

Example 1 (An epidemic model [15]). *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 (finite) set of all actions of an object specification Δ is denoted by $\mathcal{A}_\Delta$. Similarly, the (finite) set of its states is denoted by $\mathcal{S}_\Delta$.

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 with value in $[0, 1]$, built from constants $v \in [0, 1]$ and the special operator $\mathsf{frc}\, C$ combined using standard arithmetics operators; for object state C, $\mathsf{frc}\, C$ returns the *fraction* of the objects which are currently in state C, over the total of N objects (i.e. the element of the occupancy measure vector corresponding to state C). Clearly, the use of the frc operator allows action (and, ultimately, transition) probability to depend on the global state, via the occupancy measure vector.

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

```
inf_ext ::  α_e;
inf_sus ::  α_i * (frc I);
activate ::  α_a;
patch ::  α_r;
loss ::  α_s;
```

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

A system is modelled as a population of N instances of an object, so 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* (vector), where $\mathbf{C}_{0[j]} = C_{0_j} \in \mathcal{S}_\Delta$ is the initial state of the j-th instance of the object in the population[6], for $j = 1 \ldots N$. We say that N is the *population size* of the system. For the kind of systems we are interested in, it is natural to assume $N >> S$, where S is the number of states of the single object (specification) $\mathcal{S}_\Delta$. 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. In summary, a system specification can be thought of as process algebraic clock-synchronous parallel composition of N process instantiations.

The probabilistic behaviour of a system can be derived from its specification $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$. A (system) *global state* is an N-tuple $\mathbf{C}^{(N)} \in \mathcal{S}_\Delta^N$. Let $\mathcal{S}_\Delta = \{C_1, \ldots, C_S\}$ and $\mathcal{U}^S = \{\mathbf{m} \in [0, 1]^S \mid \sum_{i=1}^S \mathbf{m}_{[i]} = 1\}$ be the unit simplex of dimension S; we can assume, w.l.o.g. that there is a total ordering on $\mathcal{S}_\Delta$ 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 $\mathbf{M}^{(N)}(\mathbf{C}^{(N)}) = \langle M_1^{(N)}, \ldots, M_S^{(N)} \rangle$ where

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

for $i = 1, \ldots, S$ and the value of $\mathbf{1}_{\{\alpha=\beta\}}$ is 1, if $\alpha = \beta$, and 0 otherwise. So, $M_i^{(N)}$ is the *fraction* of the object instances which are in state C_i, over the total number N, in the current global state $\mathbf{C}^{(N)}$. Object specification Δ uniquely characterises a Labelled Transition System from which an

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

(object) transition matrix can easily be derived which, *given an occupancy measure vector* $\mathbf{m}$ representing the current global state, associates with each pair of states C and C' the probability of jumping from C to C' in one step *in that specific global state*. Such a matrix is thus a function $\mathbf{K}^{(N)} : \mathcal{U}^S \times \mathcal{S}_\Delta \times \mathcal{S}_\Delta \to [0,1]$ and the step probability mentioned above is given by $\mathbf{K}^{(N)}(\mathbf{m})_{C,C'}$.

The behaviour of the system is the result of the parallel-synchronous execution of the N instances of the object. Thus, the probabilistic behaviour of the system is characterised by the DTMC $\mathbf{X}^{(N)}(t)$ with initial probability distribution $\boldsymbol{\delta}_{\mathbf{C}_0}$ and one step probability matrix $\mathbf{P}^{(N)}$ defined by the following product:

$$\mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \Pi_{n=1}^{N} \mathbf{K}^{(N)}(\mathbf{M}^{(N)}(\mathbf{C}))_{\mathbf{C}_{[n]},\mathbf{C}'_{[n]}}. \tag{2}$$

Of course, the 'occupancy measure' view of the evolution in time of stochastic process $\mathbf{X}^{(N)}(t)$ is again a DTMC, namely the *occupancy measure DTMC*, which is defined as expected: $\mathbf{M}^{(N)}(t) = \mathbf{M}^{(N)}(\mathbf{X}^{(N)}(t))$.

PCTL local Model-checking. For the purpose of expressing system properties in PCTL, we partition the set of atomic propositions $\mathscr{P}$ into sets $\mathscr{P}_1$, for expressing properties of the *first*[7] object of a system, and $\mathscr{P}_G$ for global properties. 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 $\mathscr{P}_1$. Recall that, as far as *local* properties are concerned, we are focused on the first object of the system; thus we extend ℓ_1 to global states by simply requiring $\ell_1(\langle C_1, \ldots, C_N \rangle) = \ell_1(C_1)$; this way, we can express *local* properties of the first object in the system, in the *context* of the complete population.

In order to express directly also (a limited class of) properties of the population *global* state, we use set $\mathscr{P}_G$. The system specification is further enriched by associating state labels $\mathbf{g} \in \mathscr{P}_G$ with boolean expressions B built from values $v \in [0,1]$, the already mentioned expression $\mathsf{frc}\, C$, for $C \in \mathcal{S}_\Delta$ and comparison operators $<$ and $>$[8]. Again, note that the use of $\mathsf{frc}\, C$ implies that the truth value of these boolean expressions B may depend on the current occupancy measure vector. An additional state labelling function ℓ_G is then defined which associates each system global state $\mathbf{C}$ with the set $\ell_G(\mathbf{C})$ of those labels $\mathbf{g} \in \mathscr{P}_G$ such that the boolean expression associated with $\mathbf{g}$ evaluates to *true* in state $\mathbf{C}$. 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 $\ell_{\mathcal{D}^{(N)}}(\mathbf{C}) = \ell_1(\mathbf{C}) \cup \ell_G(\mathbf{C})$. We finally let $\mathscr{P}_\Delta$ denote the range of $\ell_{\mathcal{D}^{(N)}}$.

The definition of $Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$ as well as that of the satisfaction relation $\models_{\mathcal{D}^{(N)}}$ are obtained by instantiating those given in Section 3.1 to $\mathcal{D}^{(N)}$. For $\sigma \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, $\sigma[j]_{[n]}$ denotes the n-th local state of global state $\sigma[j]$.

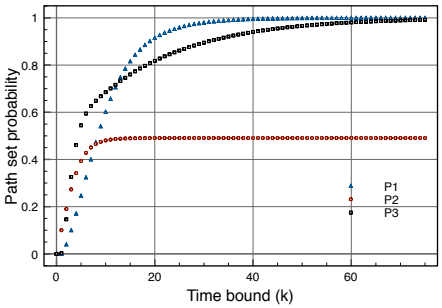
For model-checking a system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$ we instantiate proc with[9] $\mathcal{S}_\Delta^N$ and lab with $\mathscr{P}_\Delta$. Function next is instantiated to the function $\mathsf{next}_{\mathcal{D}^{(N)}}$, where

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

[7] 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.

[8] In [16] a richer language is considered. Here we address a limited version thereof, for the sake of simplicity.

[9] 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.



	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).

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 $\mathbf{C}$. The state-space size grows as S^N, with S being the cardinality of $\mathcal{S}_\Delta$, but only those elements of $\mathbf{P}^{(N)}$ that are necessary for $\mathsf{next}_{\mathcal{D}^{(N)}}$ are actually computed. Of course, in the worst case, i.e. when every state $\mathbf{C}'$ is reachable in one step from $\mathbf{C}$ with some, even small, probability, $\mathsf{next}_{\mathcal{D}^{(N)}}(\mathbf{C})$ returns the complete, S^N-sized, state space. Function $\mathsf{lab\_eval}$ is instantiated with function $\mathsf{lab\_eval}_{\mathcal{D}^{(N)}} : \mathcal{S}_\Delta^N \times \mathscr{P}_\Delta \to \mathbb{B}$ with $\mathsf{lab\_eval}_{\mathcal{D}^{(N)}}(\mathbf{C}, \mathsf{ap}) = \mathsf{ap} \in \ell_{\mathcal{D}^{(N)}}(\mathbf{C})$.

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* $(\mathsf{frc}\, I) < 0.25$*:*

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

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

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 [28] and using exact on-the-fly PCTL model-checking are presented[10], showing that the latter has comparable performance in time. 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 Section 5 we instantiate the procedure with the mean-field, approximated, semantics of the language, leading to a scalable, 'fast', model-checker.

[10] 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://goo.gl/BdWV42`).

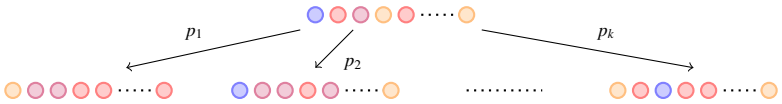


Figure 3: A fragment of $\mathcal{D}^{(N)}(t)$ transition matrix $\mathbf{P}^{(N)}$.

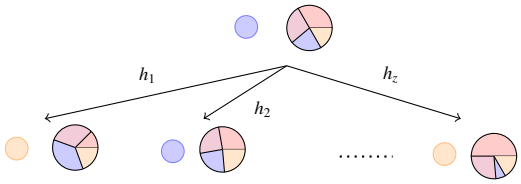


Figure 4: A fragment of $\mathcal{HD}^{(N)}(t)$ transition matrix $\mathbf{H}^{(N)}$.

5. Fast Mean-field On-the-fly Approximated 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 of the behaviour of an individual object, conventionally the first object, within the context of a system modelled by $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$. 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 (1) on page 12, 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. $C = \mathbf{C}^{(N)}_{[1]}$, 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 fundamental convergence result by Le Boudec et al. (Theorem 4.1 of [13]) 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 truly drastic reduction in the maximal size of the state space generated for evaluating the formula. 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 [13]. More specifically, such a combination gives rise to a new state labelled DTMC, $\mathcal{HD}(t)$, which is the limit of $\mathcal{HD}^{(N)}(t)$, when N goes to infinity (see

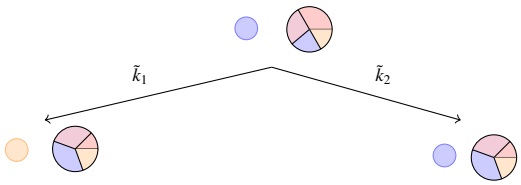


Figure 5: A fragment of $\mathcal{HD}(t)$ transition matrix.

Fig. 5). We show that, 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 approximated model-checking.

5.1. The Abstraction Step

Given a system specification $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$ with initial global 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 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 $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) = \langle \mathbf{C}^{(N)}_{[1]}, \mathbf{M}^{(N)}(\mathbf{C}^{(N)}) \rangle$. Note that $\mathcal{H}^{(N)}$ and $\mathcal{D}^{(N)}(t)$ together characterise a state labelled DTMC, defined as $\mathcal{H}^{(N)}(\mathbf{X}^{(N)}(t))$, which we denote by $\mathcal{HD}^{(N)}(t)$. The labelling function $\ell_{\mathcal{HD}^{(N)}}$ of $\mathcal{HD}^{(N)}$ is defined in the expected way: $\ell_{\mathcal{HD}^{(N)}}(\langle C, \mathbf{m} \rangle) = \ell_1(\langle C, \mathbf{m} \rangle) \cup \ell_G(\langle C, \mathbf{m} \rangle)$ where $\ell_1(\langle C, \mathbf{m} \rangle) = \ell_1(C)$ and $\ell_G(\langle C, \mathbf{m} \rangle)$ is the set of those $\mathbf{g} \in \mathscr{P}_G$ such that the boolean expression associated to $\mathbf{g}$ evaluates to *true* in $\mathbf{m}$. The one-step probability matrix of $\mathcal{HD}^{(N)}(t)$ is:

$$\mathbf{H}^{(N)}_{\langle C,\mathbf{m}\rangle,\langle C',\mathbf{m}'\rangle} = \sum_{\mathbf{C}' : \mathcal{H}^{(N)}(\mathbf{C}') = \langle C', \mathbf{m}' \rangle} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \tag{3}$$

where $\mathbf{C}$ is such that $\mathcal{H}^{(N)}(\mathbf{C}) = \langle C, \mathbf{m} \rangle$. Note that the above definition is a good definition; in fact, if $\mathbf{M}^{(N)}(\mathbf{C}) = \mathbf{M}^{(N)}(\mathbf{C}'')$ and $\mathbf{C}_{[1]} = \mathbf{C}''_{[1]}$, then $\mathbf{C}$ and $\mathbf{C}''$ are just two permutations of the same local states, starting with the same value $\mathbf{C}_{[1]}$, which implies that for all $\mathbf{C}'$ we have $\mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \mathbf{P}^{(N)}_{\mathbf{C}'',\mathbf{C}'}$. Finally, the initial distribution of $\mathcal{HD}^{(N)}(t)$ is $\delta_{\mathcal{H}^{(N)}(\mathbf{C}^{(N)}_0)}$. The size of the state-space is bounded by $S \cdot N^S$.

The definitions of the set of paths for state $\langle C, \mathbf{m} \rangle$ of $\mathcal{HD}^{(N)}(t)$[11], that is $Paths_{\mathcal{HD}^{(N)}}(\langle C, \mathbf{m} \rangle)$, of the set of possible states $\mathcal{HD}^{(N)}$ at step t, $L_{\mathcal{HD}^{(N)}}(t)$, 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 Section 3.1 to the model $\mathcal{HD}^{(N)}(t)$. Furthermore, we let $L_{\mathcal{HD}^{(N)}}(t, C) = \{\langle C', \mathbf{m}' \rangle \in L_{\mathcal{HD}^{(N)}}(t) \mid C' = C\}$ be the set of those states of $\mathcal{HD}^{(N)}$ at time t which have local state C of the first object as first component. Mapping $\mathcal{H}^{(N)}$ is extended 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 $\sigma \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, let $\mathcal{H}^{(N)}(\sigma) = \mathcal{H}^{(N)}(\sigma[0])\mathcal{H}^{(N)}(\sigma[1])\mathcal{H}^{(N)}(\sigma[2]) \cdots$

The following lemma relates the two interpretations of the logic showing that $\mathcal{H}^{(N)}$ preserves bounded PCTL (see Appendix A for the proof).

[11] In the sequel, for the sake of notational simplicity, for generic stochastic process $Y(t)$ we will often drop the (t) from the notation, writing just Y, whenever this does not create confusion.

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* $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$. •

5.2. The Approximation Step

We now move to Fast Simulation and its exploitation in the on-the-fly approximated model-checking procedure. To that purpose, let us first 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 $\mathbf{m}$:

$$\begin{aligned} &\mathbb{P}\{\mathcal{HD}(0) = \langle C, \mathbf{m}\rangle\} = \boldsymbol{\delta}_{\langle C_0,\boldsymbol{\mu}_0\rangle}(\langle C, \mathbf{m}\rangle), \\ &\mathbb{P}\{\mathcal{HD}(t+1) = \langle C', \mathbf{m}'\rangle \,|\, \mathcal{HD}(t) = \langle C, \mathbf{m}\rangle\} = \begin{cases} \mathbf{K}(\mathbf{m})_{C,C'}, & \text{if } \mathbf{m}' = \mathbf{m} \cdot \mathbf{K}(\mathbf{m}) \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

The states of $\mathcal{HD}(t)$ are of the same type as those of $\mathcal{HD}^{(N)}(t)$, with initial state $\langle C_0, \boldsymbol{\mu}_0\rangle$. The right-hand element of each state is deterministic and at step $t > 0$ it is recursively computed from that of (any state at) step $t - 1$, with $\boldsymbol{\mu}_0$ as initial value. Finally, under the assumption that $\mathbf{m}$ is the right-hand element at time t, the probability of jumping, at step t from a state $\langle C, \mathbf{m}\rangle$ to a state with left-hand component C' is given by $\mathbf{K}(\mathbf{m})_{C,C'}$; we stress here the fact that any such next state will have the same right-hand element, i.e. $\mathbf{m} \cdot \mathbf{K}(\mathbf{m})$.

Note that $\mathcal{HD}(t)$ 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 (4). Given our system specification $\langle \Delta, A, \mathbf{C}_0\rangle^{(N)}$, with probability functions and state labels duly defined, we make $\mathcal{HD}(t)$ a labelled DTMC, just by defining $\ell_{\mathcal{HD}}$ exactly as for $\ell_{\mathcal{HD}^{(N)}}$. 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 Section 3.1 to the model $\mathcal{HD}(t)$. Furthermore, we formalise the computation of $\mathbf{m}$ in the states of $\mathcal{HD}(t)$ by defining 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 $\mathbf{m} = \boldsymbol{\mu}(t)$.

As we will see below, with an appropriate choice of $\boldsymbol{\mu}_0 \in \mathcal{U}^S$ and function $\mathbf{K}$, process $\mathcal{HD}(t)$ will play a central role in the on-the-fly approximate model-checking procedure. In fact $\mathcal{HD}(t)$ will be the mean-field semantics of our modelling system language. We first recall the fundamental result stated below, due to Le Boudec et al. [13].

> **Theorem 4.1 of [13]** *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 $\lim_{N\to\infty} \mathbf{M}^{(N)}(t) = \boldsymbol{\mu}(t)$.•*

We first of all observe that, for large N, state labelling $\ell_{\mathcal{HD}^{(N)}}$ of $\mathcal{HD}^{(N)}$ and $\ell_{\mathcal{HD}}$ of $\mathcal{HD}$ for corresponding states coincide, almost surely, as formalised by the following

Remark 1. As direct consequence of Theorem 4.1 of [13] and of the definition of the expressions in the modelling language described in Section 4, letting $\mathscr{E}[\![\cdot]\!]_{\mathbf{m}}$ denote the expression evaluation

function for the language, evaluated on occupancy measure vector $\mathbf{m}$, for any fixed t and for all $\epsilon > 0$, there exists $\bar{N}$ such that, for all $N \geq \bar{N}$, almost surely

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

for all $\langle C, \mathbf{m}\rangle \in L_{\mathcal{HD}^{(N)}}(t)$ and any expression E. As a consequence, boolean expressions built using such expressions E will, almost surely, yield the same boolean value both on $\mathbf{m}$ and on $\boldsymbol{\mu}(t)$ for such t and N. In other words, for N large enough and $\langle C, \mathbf{m}\rangle \in L_{\mathcal{HD}^{(N)}}(t)$, $\ell_{G,\mathcal{HD}^{(N)}}(\langle C, \mathbf{m}\rangle) = \ell_{G,\mathcal{HD}}(\langle C, \boldsymbol{\mu}(t)\rangle)$, and, consequently, $\ell_{\mathcal{HD}^{(N)}}(\langle C, \mathbf{m}\rangle) = \ell_{\mathcal{HD}}(\langle C, \boldsymbol{\mu}(t)\rangle)$. •

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 (4) 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 [13], 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 [13], for any fixed t, almost surely* $\lim_{N\to\infty} \mathcal{HD}^{(N)}(t) = \mathcal{HD}(t)$. •

This convergence result is fundamental for our approach. In fact, it allows us to use process $\mathcal{HD}(t)$ as a model in the model-checking procedure instead of process $\mathcal{HD}^{(N)}(t)$, for large N. The consequence is that our on-the-fly algorithm will work on a model where all the occupancy measure vectors to be considered at any given time step t will collapse to a single vector, namely $\boldsymbol{\mu}(t)$ above, with clear decrease of memory requirements. Furthermore, the cumulative probability of jumping, at time t, from, say, $\langle C, \mathbf{m}\rangle$ to, say, $\langle C', \mathbf{m}'\rangle$, *for any* $\mathbf{m}'$ allowed by the dynamics of $\mathcal{HD}^{(N)}(t)$, for large N can be simply approximated by $\mathbf{K}(\boldsymbol{\mu}(t))_{C,C'}$ above, as formalised by the following

Remark 2. A consequence of Corollary 3 is that, under the assumptions of Theorem 4.1 of [13], 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* enough population sizes N, a formula Φ is satisfied by a state $\mathcal{H}^{(N)}(\mathbf{C}^{(N)})$ in $\mathcal{HD}^{(N)}(t)$ at time $\tilde{t}$ if and only if it is satisfied by the corresponding state in $\langle \mathbf{C}_{[1]}^{(N)}, \boldsymbol{\mu}(\tilde{t})\rangle$ in $\mathcal{HD}(t)$. In particular, 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. A necessary condition for the correspondence theorem is that the formula Φ is *safe*: 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 $\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{M}}(s) \mid \eta \models_{\mathcal{M}} \varphi\} \neq p$.

The theorem, together with Theorem 1 and Lemma 2, establishes the formal relationship between the satisfaction relation on the exact semantics $\mathcal{HD}^{(N)}(t)$ of the language and that on its mean-field approximation $\mathcal{HD}(t)$, thus justifying the fast local model-checking instantiation we will show in the sequel.

Theorem 4. *Under the assumptions of Theorem 4.1 of [13], 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, it holds:* $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$ *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, linking model-checking on the approximated model to that on the original model, namely DTMC $\mathcal{D}(t)$:

Corollary 5. *Under the assumptions of Theorem 4.1 of [13], 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* $\langle \mathbf{C}^{(N)}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi$.•

Remark 3. It is in general not so easy to establish for which population size the approximation is sufficiently good. It would be useful to provide confidence bounds on the basis of error bounds in the computation of the relevant probabilities. This is not straightforward and a topic for future research. See Section 7 for further discussion on the issue.

5.3. Fast Mean-Field On-the-fly PCTL Approximated Model-checking

On-the-fly *fast* PCTL approximated 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 $\mathscr{P}_\Delta$; 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 [13]; lab_eval is instantiated as expected:

$$\mathsf{lab\_eval}_{\mathcal{HD}}(\langle C, \mathbf{m}\rangle, \mathsf{ap}) = \mathsf{ap} \in \ell_{\mathcal{HD}}(\langle C, \mathbf{m}\rangle).$$

Note that, in the worst case, $\mathsf{next}_{\mathcal{HD}}(\langle C, \mathbf{m}\rangle)$ returns S states only, due to the collapse of the occupancy measure vectors to a single one, at the relevant step. It is worth pointing out that, would we have used the instantiation on the abstract, but *not* approximated, semantics $\mathcal{HD}^{(N)}(t)$, the worst case behaviour of $\mathsf{next}_{\mathcal{HD}^{(N)}}(\langle C, \mathbf{m}\rangle)$ would have returned $S \cdot N^S$ states. The instantiation is implemented in FlyFast.

Remark 4. In the hypothesis of the theorem formula safety is required. Formula safety is a requirement typical of fluid-flow/mean-field model-checking algorithms (see [56, 18]) which is expensive to check. The weaker safety check that $\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(s') \mid \eta \models_{\mathcal{HD}} \varphi\}$ is not close to p for all formulas $\mathcal{P}_{\bowtie p}(\varphi)$ and states s' can be performed automatically during the execution of $\mathsf{CheckPath}(s, \Phi)$ (see Table 3). If the approximated probability value is close to p, this is an indication that the actual probability measure might be close to p, so that the result of the model-checking session might be unreliable. •

Example 4 (FlyFast results). *Fig. 6 shows the result of* 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* $\mathit{true}\ \mathcal{U}^{\leq k}\ (!e \wedge !i \wedge \mathcal{P}_{>0.3}(\ \mathit{true}\ \mathcal{U}^{\leq 5}\ i\ ))$ *of property* P3 *on page 14, (for* $k = 3$*), changes for initial time* $t0$ *varying from* 0 *to* 10*.*

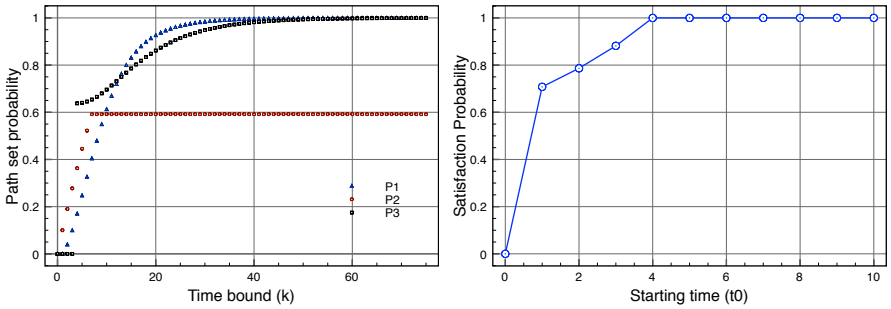


Figure 6: Fast model-checking results.

6. Case Study: A Bike Sharing System

In this section we illustrate the use of the on-the-fly mean-field approximated model-checker FlyFast by its application on a case study, namely a recent model of a bike sharing system inspired by the continuous time stochastic model developed by Fricker and Gast in [58].

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) [11, 62]. 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 varies 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 over time, for example due to people that rely on bikes to go to their work or to 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 [58] 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 FlyFast model-checker.

6.1. A Homogeneous Bike Sharing Scenario

The first model we consider is a homogeneous 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 park a bike. By "homogeneous", we mean that we abstract from the actual spatial distribution of bike stations. Taking limited forms of spatial distribution and space-dependent issues into account in this particular example would not be difficult: one could for example define classes of stations characterised by different probabilistic features (e.g. due to different demands in different areas of a city). It is not difficult to define different "classes" of objects using our modelling language, although we did not show this in the present paper for the sake of simplicity[12]. However, taking full account of detailed spatial issues requires further research (see also the related discussion in Section 7), both from a conceptual point of view and from a scalability point of view. We therefore abstract from treating spatial issues in the current example. Furthermore, our aim here is to show that our analysis approach is feasible and scales to the size of real systems. It provides efficient approximations of expected performance aspects of a large scale system, which is for example very useful for the analysis of different design options in the early stages of system design.

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, and `YK` denotes that the station is full. The states `Yk`, where $1 \leq \mathtt{k} \leq \mathtt{K} - 1$ denote stations containing `k` bikes. A graphical representation of the model is given in Fig. 7.

[12]For an example with different "classes" of objects using our modelling language, the reader is referred to [59], where also the application on the well-known and extensively discussed predator-prey model by Lotka and Volterra [60, 61] is presented and interesting oscillating behaviour is studied in the model-checking framework.

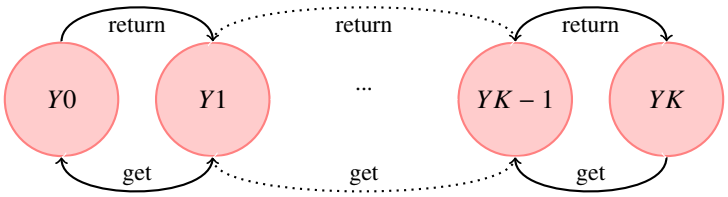


Figure 7: Bike station.

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 $(sN - \sum_{k=1}^{K} kNy_k)$ 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 $C = (sN - \sum_{k=1}^{K} kNy_k)/sN$ which gives $C = (s - \sum_{k=1}^{K} ky_k)/s$, or equivalently $C = 1 - (\sum_{k=1}^{K} ky_k)/s)$. The probability that a bike is returned to a station is then $\alpha_r C$.

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

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, assuming for simplicity that the average duration that a bike is used is 1 time unit. 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 $K = 10$, $s = 5$, $N = 1000$, $\alpha_g = 1/(s+1)$ 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 for mean-field approximation on the left and for the average of 100 random simulation runs for 1000 bike stations on the right. We can observe that the number of empty stations quickly decreases while less than 10% of stations get full in the long run. The mean-field and full simulation results show a very good correspondence. 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. Fig. 9 shows the probability that an empty station gets full when

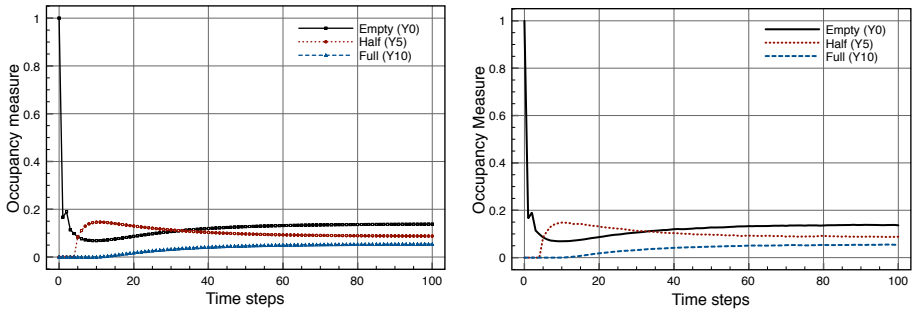


Figure 8: Fraction of empty, half full and full stations for $s = 5$, $req = 1$ (and using $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$). mean-field approximation (left) and full simulation for N=1000 (right)

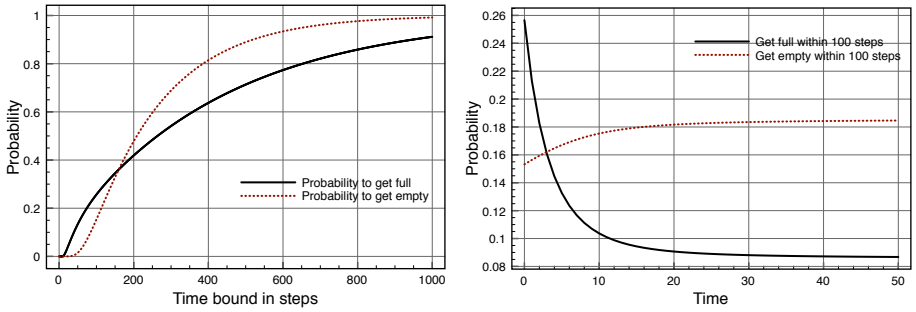


Figure 9: (Left) Probability to get full and empty within time bound t expressed in number of steps: $\mathcal{P}_{=?}(\text{tt } \mathcal{U}^{\leq t}\ \texttt{Y10})$ (for $s = 5$, $req = 1$). (Right) Probability of the individual station to get full (empty resp.) within 100 steps starting from the individual 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 $\alpha_r = s/(s+1)$.

starting from an initial state of the system in which all the stations are empty. This is expressed as $\mathcal{P}_{=?}(\text{tt } \mathcal{U}^{\leq t}\ \texttt{Y10})$. The path formula tt $\mathcal{U}^{\leq t}$ 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 (left). 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 (left).

Fig. 9 (right) 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. The former is expressed by $\mathcal{P}_{=?}(\text{ tt } \mathcal{U}^{\leq 100}\ Y10\ )$. It is assumed that the overall system at time 0 is empty, i.e. all stations other than the individual one are empty. The formula for the latter situation (from full to empty) is similar. 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

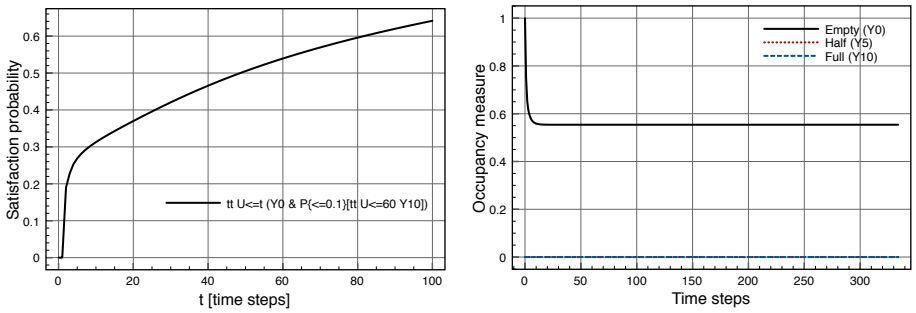


Figure 10: (Left) Satisfaction probability of a nested formula. (Right) Fraction of empty, half full and full stations for $s = 5$, $req = 10$ (with $\alpha_g = 10/(s+10)$ and $\alpha_r = s/(s+10)$).

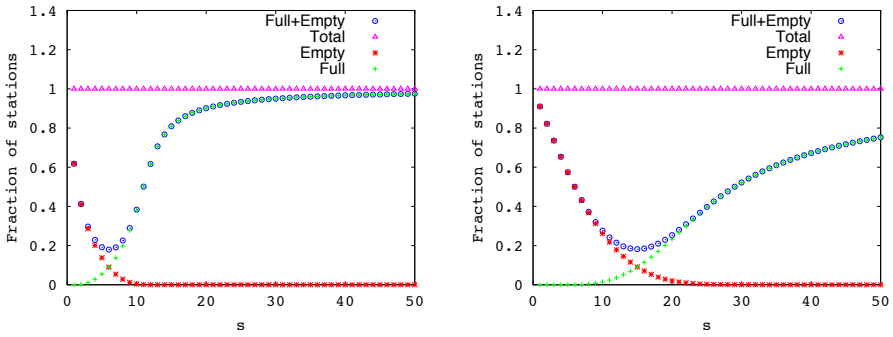


Figure 11: (Left) Fraction of problematic stations for varying values of s where $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$. (Right) Fraction of problematic stations for varying values of s where $\alpha_g = 10/(s+10)$ and $\alpha_r = s/(s+10)$.

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. 10 (left) shows the satisfaction probability of the nested path formula $\text{tt } \mathcal{U}^{\leq t} (Y0 \wedge \mathcal{P}_{\leq 0.1}(\text{ tt } \mathcal{U}^{\leq 60} Y10))$ denoting the probability that the single station reaches a state, within t time steps, in which the probability is less than 0.1 that it passes from being empty to being full within 60 time steps. Note how this satisfaction probability is first zero, and then increases when t increases, illustrating that the probability that an empty station gets full within 60 time steps varies over time.

Fig. 10 (right) 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.

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 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. 11 (left) 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 [58] 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 abstract 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. 11 (right) 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.

6.2. A Bike Sharing Scenario with User Incentives

One of the main problems concerning 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 following a form of self-organising coordination [57]. If enough users react to the incentives such strategies would have an effect of self-adaptation at the global system level. We present an extension of the previous bike sharing model that models a particular strategy of incentives, again taking inspiration from a continuous Markov model presented by Fricker and Gast in [58].

A well-known load balancing strategy is selecting the best among two offered choices (see Mitzenmacher [63]). In the case of bike sharing this strategy consists in the user returning the bike to the station with fewer bikes among two randomly selected ones. In our setting the selected ones are not required to be adjacent.

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:

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

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:

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

Fig. 12 shows the fraction of empty, half full and full stations of the model with incentives for $K = 10$, $N = 1000$, $s = 5$ 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.

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. 13 (left) 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. 13 (right) 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.

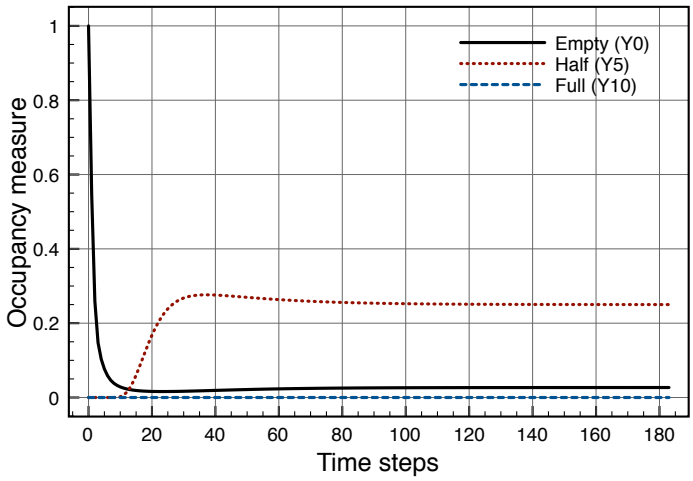


Figure 12: Fraction of empty, half full and full stations for $s = 5$, $req = 1$ (and using $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$).

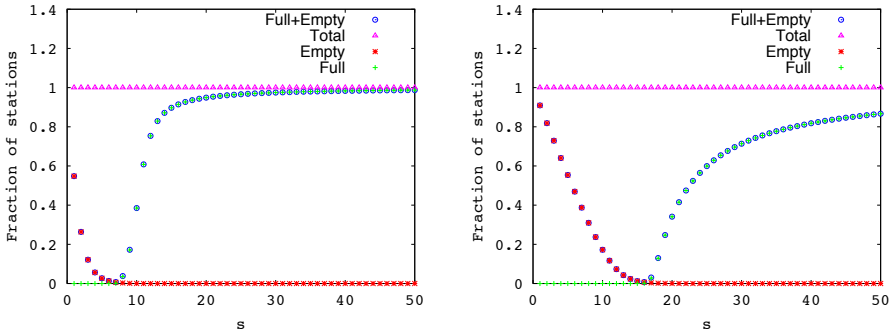


Figure 13: (Left) Fraction of problematic stations for varying values of s where $\alpha_g = 1/(s+1)$ and $\alpha_r = s/(s+1)$. (Right) Fraction of problematic stations for varying values of s where $\alpha_g = 10/(s+10)$ and $\alpha_r = s/(s+10)$.

6.3. Some final considerations

The model-checking results of the bike sharing models suggest that the proposed strategy of user incentives may indeed lead to a considerable reduction of the fraction of problematic stations in the system, at least when an appropriate fleet size is chosen. This form of adaptivity, where users act locally on an incentive, for example by receiving some free rides or other small rewards, leads to a better level of service of the overall system. Stochastic simulation results obtained for a more detailed model in which the users can choose the least occupied station among two *adjacent* ones indicates that also in this setting the strategy leads to a significant improvement, even in the situation in which only 20% of the users follow the incentive [58]. It would also be interesting to see whether similar results as in [58] can be obtained using statistical model-checking instead of stochastic simulation as a complementary analysis technique in case of relatively small bike sharing systems.

The more abstract models used for mean-field model-checking are useful because they provide a first indication that a strategy like user incentives may work well with relatively little modelling and computational effort. Based on such first results more detailed analysis can then be performed, either by developing more detailed models for mean-field model-checking such as introducing spatial aspects in a more explicit way, or introducing time-dependent request rates for different areas of the city. Of course, the more details are introduced, the more costly and time-consuming the analysis may get. Therefore it is useful to first try to select only the most promising configurations using more abstract models first. It is worth remarking that stochastic population models can also be analysed using other model-checking methods, either exact [28] or statistical, simulation based ones [64]. These techniques, however, suffer from state space explosion in different ways, but both of them have a computational complexity which increases with the number N of objects in the system. The dependence is linear for statistical model-checking, and polynomial or exponential for numerical model-checking. The computational complexity of fast mean-field approximated model-checking, instead, does *not* depend on N.

7. Conclusions and Future Work

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. In this paper we have presented a fast PCTL approximated model-checking approach that builds upon *local, on-the-fly* model-checking and *mean-field* approximation. Traditional model-checking has shown to be a very useful technique in the system design phase of concurrent and distributed 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 fact, the complexity of the mean-field approximated model-checking algorithm is independent of the population size N.

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 set of underlying difference equations is 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 relatively 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 still provide 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 selecting and studying in more detail those which seem more promising. An example of such model-development can be found in [34] 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-organising collective systems. Such phenomena are also a reminder of the fact that it is important to analyse collective systems using several complementary approaches. Furthermore, one has to be aware of the fact that mean-field techniques are *approximate* 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 model-checking or forms of approximate model-checking different from mean-field-based ones [53, 64, 54]. These simulation-based techniques may be computationally costly when 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 appeared 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[13].

In this paper we have shown that the combination of on-the-fly model-checking based on high-level modelling-language and mean-field approximation in a discrete time setting is feasible and promising. We used an instantiation of a (fragment of a) parametric probabilistic model-checking algorithm we proposed in [21]. We presented related correctness results, and an example of application to collective systems of a prototype implementation of the FlyFast model-checker.

There are several lines of future work of our interest. First of all, following approaches similar to those presented in [13], we plan to investigate the extension of the model-checking technique to systems with memory. Furthermore, although not shown in the present paper, it is technically easy to deal with multiple populations in our approach, by just using multiple object

[13] See, e.g., `http://www.quanticol.eu/`

classes, i.e. allowing multiple object definitions in a system specification; nevertheless, we would be 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, in much a similar way as done for the continuous time stochastic setting, see for example [31]. Also for population models in the probabilistic discrete time setting there have been some proposals in this direction, e.g. [36], but further research is needed.

Spatial information is often embedded in system modelling languages; the space structures that are typically used are sets of discrete localities with an adjacency relation (see, e.g. [65]) or hierarchical structures as in the Ambient Calculus [66]. The information concerning space is usually coded in object/system states. A similar approach can be taken for our system modelling language and underlying semantics. For instance, extending the language with localities would be pretty easy. On the other hand, we think that space plays a major role in collective adaptive systems and, consequently, it should be a 'first class' component of the modelling language and the underlying framework. For this reason, we are currently investigating different notions of space [67], appropriate (modal) logics for reasoning about space [68, 69, 70], and novel model-checking techniques for the efficient automatic verification of properties of points or regions of space [71, 73, 72], as well as their relationship with specific languages for spatial computing (see, e.g. [74]).

The correctness theorem for on-the-fly mean-field model-checking algorithm presented in the present paper requires formula safety. We did not investigate formula safety in depth in this paper. By relying on a reasonable level of accuracy of the mean-field approach (see below), the user can be automatically warned when the approximated probability for a path formula is equal or very close to the formula probability bound. We plan to address the issue following an approach similar to that developed in [56] for formula robustness.

Our model-checking technique is ultimately based on mean-field approximation of the occupancy measure of the population Markov process. An important question is what guaranteed estimates can be obtained of the deviation of the stochastic process from its deterministic limit and the related impact on the result of the approximated model-checking procedure. The first question is addressed in [75]. Among other things, the proposed method, for the first time, allows one to obtain transient error bounds which do not explode exponentially, as was the case in previously known approaches to compute error bounds. We are interested in investigating on how the approach proposed in [75] can be extended in order to address the second question, in relation to our model-checking procedure as well as formula safety.

Finally, we think more work is necessary for extending the atomic propositions sublanguage. 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 [65] or even distribution patterns, properties that involve several different individuals and perhaps properties that address certain stability aspects of systems [34].

Acknowledgments. The authors thank the anonymous Reviewers for their accurate and constructive comments.

Appendix 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 $\mathcal{H}^{(N)}(\mathbf{C}^{(N)}) \models_{\mathcal{HD}^{(N)}} \Phi$. •

We provide two different proofs for this lemma. The first one exploits probabilistic bisimulation, while the second is a direct proof, carried on by induction on formula Φ.

Proof. We use the notion of *Probabilistic Bisimulation* (see, e.g. [17][14]): an equivalence relation R over the states of a labelled DTMC $\langle \mathcal{M}, \ell \rangle$ is a probabilistic bisimulation if and only if whenever $\langle s_1, s_2 \rangle \in R$ the following two conditions hold, where $\mathbf{P}$ is the one step probability matrix of $\mathcal{M}$:

- $\ell(s_1) = \ell(s_2)$
- $\sum_{s \in C} \mathbf{P}_{s_1,s} = \sum_{s \in C} \mathbf{P}_{s_2,s}$ for each equivalence class C of R,

Two states s_1 and s_2 are *bisimilar* if there exists a bisimulation R such that $\langle s_1, s_2 \rangle \in B$.

Let $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$ be a system specification. Consider the associated state labelled DTMC $\mathcal{D}^{(N)}(t)$ and its abstraction labelled DTMC $\mathcal{HD}^{(N)}(t)$. It is easy to see that each state $\mathbf{C}^{(N)}$ of $\mathcal{D}^{(N)}(t)$ is bisimilar to $\mathcal{H}^{(N)}(\mathbf{C}^{(N)})$ in $\mathcal{HD}^{(N)}(t)$. To that purpose, let us consider the relation defined as follows:

$$R = H \cup H^{-1} \cup K \cup E$$

where

- $H = \{\langle \mathbf{C}, \mathcal{H}^{(N)}(\mathbf{C}) \rangle | \mathbf{C} \text{ is a state of } \mathcal{D}^{(N)}(t)\}$
- $H^{-1} = \{\langle \mathcal{H}^{(N)}(\mathbf{C}), \mathbf{C} \rangle | \mathbf{C} \text{ is a state of } \mathcal{D}^{(N)}(t)\}$
- $K = \{\langle \mathbf{C}_1, \mathbf{C}_2 \rangle | \mathbf{C}_1, \mathbf{C}_2 \text{ are states of } \mathcal{D}^{(N)}(t) \text{ and } \mathcal{H}^{(N)}(\mathbf{C}_1) = \mathcal{H}^{(N)}(\mathbf{C}_2)\}$
- $E = \{\langle \mathcal{H}^{(N)}(\mathbf{C}), \mathcal{H}^{(N)}(\mathbf{C}) \rangle | \mathbf{C} \text{ is a state of } \mathcal{D}^{(N)}(t)\}$

Reflexivity and symmetry of R follow directly from its definition. For transitivity, suppose $s_1 \, R \, s_2$ and $s_2 \, R \, s_3$. Suppose $s_1 = \mathbf{C}_1$ is a state of $\mathcal{D}^{(N)}(t)$. In this case either $s_2 = \mathcal{H}^{(N)}(\mathbf{C}_1)$ or $s_2 = \mathbf{C}_2$ is a state of $\mathcal{D}^{(N)}(t)$ and $\mathcal{H}^{(N)}(\mathbf{C}_1) = \mathcal{H}^{(N)}(\mathbf{C}_2)$; if $s_2 = \mathcal{H}^{(N)}(\mathbf{C}_1)$ and $s_2 \, R \, s_3$ then either $s_3 = \mathbf{C}_1$ or $s_3 = s_2$; in both cases we get $s_1 \, R \, s_3$. If $s_2 = \mathbf{C}_2$—with $\mathcal{H}^{(N)}(\mathbf{C}_1) = \mathcal{H}^{(N)}(\mathbf{C}_2)$—and $s_2 \, R \, s_3$, then either $s_3 = \mathcal{H}^{(N)}(\mathbf{C}_2)$ or $s_3 = \mathbf{C}_3$ is a state of $\mathcal{D}^{(N)}(t)$ and $\mathcal{H}^{(N)}(\mathbf{C}_2) = \mathcal{H}^{(N)}(\mathbf{C}_3)$; in both cases we get $s_1 \, R \, s_3$ because $\mathcal{H}^{(N)}(\mathbf{C}_2) = \mathcal{H}^{(N)}(\mathbf{C}_1)$. All other cases are similar. This proves that R is an equivalence relation. Clearly, if $s \, R \, s'$ then $\ell(s) = \ell(s')$, as it follows directly from the relevant definitions. Let now C be an equivalence class of R and suppose $s_1 \, R \, s_2$. Let us assume, w.l.o.g., that s_1 is a state $\mathbf{C}$ of $\mathcal{D}^{(N)}(t)$ while $s_2 = \mathcal{H}^{(N)}(\mathbf{C})$ (this is the case of our main interest and the proof in the other cases is similar). The elements of C will be either states of $\mathcal{D}^{(N)}(t)$ or

[14] Note that in [17] a representation for DTMCs is used which is slightly different than the standard one. In particular, in [17] an absorbing state s is one such that $\mathbf{P}_{s,s'} = 0$ for all s', i.e. including s, while, typically an absorbing state is one such that $\mathbf{P}_{s,s} = 1$ (and $\mathbf{P}_{s,s'} = 0$ for all $s' \neq s$). All relevant results can be reformulated in either framework, provided the above conventional differences are taken into account.

states of $\mathcal{HD}^{(N)}(t)$; clearly, all transitions from $\mathbf{C}$ to states of $\mathcal{HD}^{(N)}(t)$ have probability 0 as well as all transitions from $\mathcal{H}^{(N)}(\mathbf{C})$ to states of $\mathcal{D}^{(N)}(t)$. Thus, we have to prove that

$$\sum_{\mathbf{C}' \in \mathcal{C}} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \sum_{\langle C', \mathbf{m}' \rangle \in \mathcal{C}} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}), \langle C', \mathbf{m}' \rangle}.$$

The above equality directly follows from the application of the definition of $\mathbf{H}^{(N)}$ (see (3) on page 16):

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

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

Case ap:
$\mathbf{C} \models_{\mathcal{D}^{(N)}} \mathsf{ap}$ if and only if $\mathsf{ap} \in \ell_1(\mathbf{C}_{[1]}) \cup \ell_G(\mathbf{C})$, 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 $\mathsf{ap} \in \ell_1(C_{[1]})$, then $\mathbf{C} \models_{\mathcal{D}^{(N)}} \mathsf{ap}$ if and only if $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \mathsf{ap}$, by definition of $\models_{\mathcal{HD}^{(N)}}$. If $\mathsf{ap} \in \ell_G(\mathbf{C})$, then $\mathbf{C} \models_{\mathcal{D}^{(N)}} \mathsf{ap}$ if and only if $\mathcal{H}^{(N)}(\mathbf{C}) \models_{\mathcal{HD}^{(N)}} \mathsf{ap}$, by definition of $\models_{\mathcal{HD}^{(N)}}$ and of the boolean expressions semantics evaluation function.
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 $\mathcal{P}_{\bowtie p}(\mathcal{X}\,\Phi)$:

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

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

$\sum_{\langle C', \mathbf{m}' \rangle : \langle C', \mathbf{m}' \rangle \models_{\mathcal{HD}^{(N)}} \Phi} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}), \langle C', \mathbf{m}' \rangle}$

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

$\sum_{\langle C', \mathbf{m}' \rangle : \langle C', \mathbf{m}' \rangle \models_{\mathcal{HD}^{(N)}} \Phi} \sum_{\mathbf{C}' : \mathcal{H}^{(N)}(\mathbf{C}') = \langle C', \mathbf{m}' \rangle} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'}$

$=$ {I.H., i.e. $\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi$ iff $\mathbf{C}' \models_{\mathcal{D}^{(N)}} \Phi$}

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

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

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

$=$ {I.H. on k}

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

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

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

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

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

Appendix B: Proof of Theorem 4

Theorem 4. Under the assumptions of Theorem 4.1 of [13], 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 ap:
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

$$p^{(N)}_{\mathbf{H}} = \sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}), \mathcal{H}^{(N)}(\mathbf{C}')} \tag{5}$$

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

$$p(t)_{\mathbf{K}} = \sum_{\mathbf{C}'_{[1]}:\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1) \rangle \models_{\mathcal{HD}} \Phi} \mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]}, \mathbf{C}'_{[1]}}. \tag{6}$$

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 $N \geq \hat{N}$ $\left| p^{(N)}_{\mathbf{H}} - p(t)_{\mathbf{K}} \right| < \epsilon$ i.e. the value $p^{(N)}_{\mathbf{H}}$ of sum (5) approaches the value $p(t)_{\mathbf{K}}$ of sum (6). Finally, safety of $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$, implies that the value $p(t)_{\mathbf{K}}$ of (6) is different from p. If $p(t)_{\mathbf{K}} > p$ then we can choose ϵ small enough that also $p^{(N)}_{\mathbf{H}} > p$ and, similarly, if $p(t)_{\mathbf{K}} < p$, we get also $p^{(N)}_{\mathbf{H}} < p$, 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 ($k = 0$):

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

$=$ {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 $\rho[0]$}

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

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.

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

$=$ {See above}

$$\begin{cases} 1, \text{ if } \langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_2, \\ \\ 0, \text{ if not } \langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi_2 \end{cases}$$

$=$ {Def. of $\mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle)$ and $\eta[0]$}

$\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle) \mid \eta[0] \models_{\mathcal{HD}} \Phi_2\}$

$=$ {Def. of $\eta \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq k}\, \Phi_2$}

$\mathbb{P}\{\eta \in \mathit{Paths}_{\mathcal{HD}}(\langle \mathbf{C}_{[1]}, \boldsymbol{\mu}(t)\rangle) \mid \eta \models_{\mathcal{HD}} \Phi_1 \,\mathcal{U}^{\leq 0}\, \Phi_2\}$

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

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

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,

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

$=$ {See above}

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

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

approaches

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

  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. □

[1] M. Dorigo, T. Stützle, Ant colony optimization, MIT Press, 2004.
[2] T. Seeley, P. Visscher, T. Schlegel, P. Hogan, N. Franks, J. Marshall, Stop signals provide cross inhibition in collective decision making by honey bee swarms, Science 335 (2012) 108 – 111.
[3] J. Buchli, C. Santini, Complexity engineering, harnessing emergent phenomena as opportunities for engineering, reports of the Santa Fe Institute's Complex Systems Summer School 2005 (2005).
[4] R. Frei, G. Di Marzo Serugendo, Concepts in complexity engineering, International Journal of Bio-Inspired Computation 3 (2) (2011) 123–139. doi:10.1504/IJBIC.2011.039911.
[5] M. Mamei, F. Zambonelli, Programming pervasive and mobile computing applications: The tota approach, ACM Trans. Softw. Eng. Methodol. 18 (4).
[6] M. Viroli, J. Beal, K. Usbeck, Operational semantics of PROTO, Sci. Comput. Program. 78 (6) (2013) 633–656.
[7] A. Omicini, Nature-inspired coordination models: Current status and future trends, ISRN Software Engineering 2013, article ID 384903, 13 pages. doi:http://dx.doi.org/10.1155/2013/384903.
[8] Ericsson, Network Society City Index: Triple-bottom-line effects of accelerated ICT maturity in cities worldwide, Tech. rep., Ericsson (2011).
URL http://www.ericsson.com/networkedsociety/media/hosting/City_Index_Report.pdf
[9] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, R. Morris, Smarter cities and their innovation challenges, Computer 44 (2011) 32–39. doi:http://dx.doi.org/10.1109/MC.2011.187.
[10] A. Caragliu, C. Del Bo, P. Nijkamp, Smart cities in Europe, Journal of Urban Technology 18 (2011) 65–82.
URL http://ideas.repec.org/p/dgr/vuarem/2009-48.html
[11] P. De Maio, Bike-sharing: Its history, impacts, models of provision, and future., Journal of Public Transportation 12 (4) (2009) 41–56.
[12] C. Fricker, N. Gast, H. Mohamed, Mean field analysis for inhomogeneous bike sharing systems, International Meeting on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (AofA 2012).
[13] J.-Y. Le Boudec, D. McDonald, J. Mundinger, A generic mean field convergence result for systems of interacting objects, in: QEST07, IEEE Computer Society Press, 2007, pp. 3–18, iSBN 978-0-7695-2883-0.
[14] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal Aspects of Computing 6 (1994) 512–535.
[15] L. Bortolussi, J. Hillston, D. Latella, M. Massink, Continuous approximation of collective system behaviour: A tutorial, Performance Evaluation 70 (5) (2013) 317 – 349. doi:10.1016/j.peva.2013.01.001.
URL http://www.sciencedirect.com/science/article/pii/S0166531613000023
[16] D. Latella, M. Loreti, M. Massink, On-the-fly mean field model-checking, in: M. Abadi, A. Lluch Lafuente (Eds.), Proceedings of the International Conference on Trustworthy Global Computing (TGC 2013), Vol. 8358 of LNCS, Springer, 2014, pp. 297–314, DOI:10.1007/978-3-319-05119-2_17, ISBN 978-3-319-05118-5 (print), 978-3-319-05119-2 (on-line), ISSN 0302-9743 .
[17] C. Baier, J. Katoen, H. Hermanns, V. Wolf, Comparative branching-time semantics for Marckov chains, Information and Computation (200) (2005) 149–214.
[18] L. Bortolussi, J. Hillston, Fluid model checking, in: M. Koutny, I. Ulidowski (Eds.), CONCUR, Vol. 7454 of LNCS, Springer-Verlag, 2012, pp. 333–347.
[19] A. Kolesnichenko, A. Remke, P.-T. de Boer, A logic for model-checking of mean-field models, Technical Report TR-CTIT-12-11, http://doc.utwente.nl/80267/ (2012).
[20] A. Kolesnichenko, A. Remke, P.-T. de Boer, A logic for model-checking of mean-field models, in: DSN13, 2013.
[21] D. Latella, M. Loreti, M. Massink, On-the-fly Probabilistic Model Checking, in: I. Lanese, A. Sokolova (Eds.), Proceedings of the 7th Interaction and Concurrency Experience (ICE 2014), June 6, 2014, Berlin, Germany, EPTCS, ISSN: 2075-2180, http://cgi.cse.unsw.edu.au/ rvg/eptcs/, 2014.
[22] H. Hermanns, U. Herzog, J. Katoen, Process algebra for performance evaluation, Theoretical Computer Science 274 (2002) 43–87.
[23] E. Brinksma, H. Hermanns, Process algebra and Markov chains, in: Lectures on Formal Methods and Performance Analysis, LNCS 2090, Springer, 2001, pp. 181–231.
[24] M. Duflot, M. Kwiatkowska, G. Norman, D. Parker, A formal analysis of Bluetooth device discovery, Int. Journal on Software Tools for Technology Transfer 8 (6) (2006) 621–632.
[25] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, R. Gupta, Using probabilistic model checking for dynamic power management, Formal Aspects of Computing 17 (2005) 160–176.
[26] C. Baier, J.-P. Katoen, H. Hermanns, Approximate Symbolic Model Checking of Continuous-Time Markov Chains, in: J. Baeten, S. Mauw (Eds.), Concur '99, Vol. 1664 of LNCS, Springer-Verlag, 1999, pp. 146–162.
[27] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, Model-Checking Algorithms for Continuous-Time Markov Chains, IEEE Transactions on Software Engineering. IEEE CS 29 (6) (2003) 524–541.
[28] M. Kwiatkowska, G. Norman, D. Parker, Probabilistic Symbolic Model Checking using PRISM: A Hybrid Approach, STTT 6 (2) (2004) 128–142.
[29] J. Hillston, Fluid flow approximation of PEPA models, in: Proceedings of the Second International Conference on

the Quantitative Evaluation of Systems (QEST 2005), IEEE, 2005, pp. 33–43. doi:10.1109/QEST.2005.12.
[30] M. Tribastone, S. Gilmore, J. Hillston, Scalable differential analysis of process algebra models, IEEE Transactions on Software Engineering 38 (2012) 205–219. doi:10.1109/TSE.2010.82.
[31] F. Ciocchetta, J. Hillston, Bio-PEPA: a framework for the modelling and analysis of biological systems, Theoretical Computer Science 410 (2009) 3065–3084.
[32] M. Massink, D. Latella, A. Bracciali, J. Hillston, Modelling non-linear crowd dynamics in Bio-PEPA, in: D. Giannakopoulou, F. Orejas (Eds.), Proceedings of the 14th International Conference on Fundamental Approaches to Software Engineering (FASE 2011), LNCS 6603, Springer, 2011, pp. 96–110.
[33] M. Massink, D. Latella, Fluid Analysis of Foraging Ants, in: M. Sirjani (Ed.), Coordination Models and Languages, Vol. 7274 of LNCS, Springer-Verlag, 2012, pp. 152–165, DOI: 10.1007/978-3-642-30829-1_11, ISSN: 0302-9743, ISBN: 978-3-642-30828-4.
[34] L. Bortolussi, D. Latella, M. Massink, Stochastic Process Algebra and Stability Analysis of Collective Systems, in: R. De Nicola, C. Julien (Eds.), Coordination Models and Languages, Vol. 7890 of LNCS, Springer-Verlag, 2013, pp. 1–15, 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).
[35] M. Massink, M. Brambilla, D. Latella, M. Dorigo, M. Birattari, On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics, Swarm Intelligence. Springer 7 (2-3) (2013) 201–228, dOI 10.1007/s11721-013-0079-6; ISSN 1935-3820 (on line), ISSN 1935-3812 (print).
URL http://link.springer.com/article/10.1007%2Fs11721-013-0079-6
[36] C. McCaig, R. Norman, C. Shankland, From individuals to populations: A mean field semantics for process algebra, Theor. Comput. Sci. 412 (17) (2011) 1557–1580.
[37] R. Darling, J. Norris, Differential equation approximations for Markov chains, Probability Surveys 5 (2008) 37–79. doi:10.1214/07-PS121.
[38] J. T. Bradley, S. T. Gilmore, J. Hillston, Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models, J. Comput. Syst. Sci. 74 (6) (2008) 1013–1032.
[39] R. Bakhshi, J. Endrullis, S. Endrullis, W. Fokkink, B. Haverkort, Automating the mean-field method for large dynamic gossip networks, in: QEST 2010, IEEE Computer Society, 2010, pp. 241–250.
[40] A. Chaintreau, J.-Y. Le Boudec, N. Ristanovic, The age of gossip: spatial mean field regime, in: J. R. Douceur, A. G. Greenberg, T. Bonald, J. Nieh (Eds.), SIGMETRICS/Performance, ACM, 2009, pp. 109–120.
[41] M. Benaïm, J. Y. Le Boudec, A class of mean field interaction models for computer and communication systems, Performance Evaluation 65 (11-12) (2008) 823–838.
[42] A. Stefanek, R. A. Hayden, J. T. Bradley, A new tool for the performance analysis of massively parallel computer systems, in: QAPL 2010. EPTCS, vol. 28, 2010, pp. 159–181.
[43] R. Hayden, Scalable performance analysis of massively parallel stochastic systems, phD Thesis, http://aesop.doc.ic.ac.uk/pubs/hayden-thesis/ (2011).
[44] C. Courcoubetis, M. Vardi, P. Wolper, M. Yannakakis, Memory-efficient algorithms for the verification of temporal properties, Form. Methods Syst. Des. 1 (2-3) (1992) 275–288.
[45] G. Bhat, R. Cleaveland, O. Grumberg, Efficient on-the-fly model checking for CTL*, in: LICS, IEEE Computer Society, 1995, pp. 388–397.
[46] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Trans. Program. Lang. Syst. 8 (2) (1986) 244–263.
[47] G. J. Holzmann, The SPIN Model Checker - primer and reference manual, Addison-Wesley, 2004.
[48] S. Gnesi, F. Mazzanti, An abstract, on the fly framework for the verification of service-oriented systems, in: M. Wirsing, M. M. Hölzl (Eds.), Results of the SENSORIA Project, Vol. 6582 of LNCS, Springer, 2011, pp. 390–407.
[49] G. Della Penna, B. Intrigila, I. Melatti, E. Tronci, M. V. Zilli, Bounded probabilistic model checking with the mur*alpha* verifier, in: A. J. Hu, A. K. Martin (Eds.), FMCAD 2004, Vol. 3312 of LNCS, Springer, 2004, pp. 214–229.
[50] E. M. Hahn, H. Hermanns, B. Wachter, L. Zhang, INFAMY: An infinite-state markov model checker, in: CAV09, LNCS, vol. 5643, Springer, 2009, pp. 641–64.
[51] A. Aziz, K. Sanwal, V. Singhal, R. Brayton, Model checking Continuous Time Markov Chains, ACM Transactions on Computational Logic 1 (1) (2000) 162–170.
[52] M. A. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, M. Dorigo, Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making, Swarm Intelligence 5 (3–4) (2011) 305–327.
[53] T. Hérault, R. Lassaigne, F. Magniette, S. Peyronnet, Approximate probabilistic model checking, in: VMCAI04. LNCS, vol. 2937, Springer, 2004, pp. 73–84.
[54] G. Guirado, T. Hérault, R. Lassaigne, S. Peyronnet, Distribution, approximation and probabilistic model checking, in: PDMC 2005. LNCS, vol. 135, Springer, 2006, pp. 19–30.

[55] N. Gast, B. Gaujal, A mean field model of work stealing in large-scale systems, in: V. Misra, P. Barford, M. S. Squillante (Eds.), SIGMETRICS, ACM, 2010, pp. 13–24.
[56] L. Bortolussi, J. Hillston, Fluid approximation of ctmc with deterministic delays, in: QEST, IEEE Computer Society, 2012, pp. 53–62.
[57] M. Viroli, M. Casadei, A. Omicini, A Framework for Modelling and Implementing Self-Organising Coordination, in: SAC, ACM, 2009, pp. 1353–1360.
[58] C. Fricker, N. Gast, Incentives and Redistribution in Bike-sharing Systems with Stations of Finite Capacity, EURO J. Transp. Logist., 2014.
[59] D. Latella, M. Loreti, M. Massink, On-the-fly PCTL Fast Mean-Field Model-Checking for Self-organising Coordination - Preliminary Version, Technical Report TR-QC-01-2013, QUANTICOL (2013).
[60] A. J. Lotka, Elements of Mathematical Biology, Williams and Wilkins Company, 1924.
[61] V. Volterra, Fluctuations in the abundance of a species considered mathematically, Nature 118 (1926) 558 – 560.
[62] Wikipedia, Bicycle sharing system — wikipedia, the free encyclopedia, (2013).
URL `http://en.wikipedia.org/w/index.php?title=Bicycle_sharing_system&oldid=573165089`
[63] M. Mitzenmacher, The power of two choices in randomized load balancing., IEEE Trans. Parallel Distrib. Syst. 12 (10) (2001) 1094–1104.
[64] H. Younes, M. Kwiatkowska, G. Norman, D. Parker, Numerical vs. statistical probabilistic model checking: An empirical study, in: K. Jensen, A. Podelski (Eds.), Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04), Vol. 2988 of LNCS, Springer, 2004, pp. 46–60.
[65] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, M. Massink, Model Checking Mobile Stochastic Logic, Theoretical Computer Science. Elsevier. 382 (1) (2007) 42–70, http://dx.doi.org/10.1016/j.tcs.2007.05.008; DOI 10.1016/j.tcs.2007.05.008.
[66] L. Cardelli, A. Gordon, Mobile ambients, in: M. Nivat (Ed.), FoSSaCS'98, Vol. 1378 of LNCS, Springer-Verlag, 1998, pp. 140–145.
[67] V. Galpin, L. Bortolussi, V. Ciancia, A. Clark, R. De Nicola, C. Feng, S. Gilmore, N. Gast, J. Hillston, A. Lluch-Lafuente, M. Loreti, M. Massink, L. Nenzi, D. Reijsbergen, V. Senni, F. Tiezzi, M. Tribastone, M. Tschaikowski, A preliminary investigation of capturing spatial information for CAS, Technical Report Deliverable D2.1, QUANTICOL (2014).
[68] M. Aiello, I. Pratt-Hartmann, J. van Benthem, (editors), Handbook of Spatial Logics, Springer, 2007.
[69] M. Massink, L. Bortolussi, V. Ciancia, J. Hillston, A. Lluch-Lafuente, D. Latella, M. Loreti, D. Reijsbergen, A. Vandin, Foundations of scalable verification for stochastic logics, Technical Report Deliverable D3.1, QUANTICOL (2014).
[70] V. Ciancia, D. Latella, M. Massink, Logics of Space and Time, Technical Report TR-QC-1-2014, QUANTICOL (2014).
[71] V. Ciancia, D. Latella, M. Loreti, M. Massink, Specifying and Verifying Properties of Space, in: 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Vol. 8705 of LNCS, Springer, 2014, pp. 222–235.
[72] L. Bortolussi, L. Nenzi, Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic, in: VALUETOOLS 2014, ACM SIGMETRICS. To Appear.
[73] V. Ciancia, S. Gilmore, D. Latella, M. Loreti, M. Massink, Data Verification for Collective Adaptive Systems: Spatial Model-checking of Vehicle Location Data, in: 2nd FoCAS Workshop on Fundamentals of Collective Systems, 2014 IEEE Eight International Conference on Self-Adaptive and Self-Organizing Systems Workshops, IEEE, 2014, pp. 32–37, DOI 10.1109/SASOW.2014.16.
[74] J. Beal, S. Dulman, K. Usbeck, M. Viroli, N. Correll, Organizing the Aggregate: Languages for Spatial Computing, in: Formal and Practical Aspects of Domain-Specific Languages: Recent Developments, IGI Global, 2013, pp. 436–501, DOI: 10.4018/978-1-4666-2092-6.ch016.
[75] L. Bortolussi, R. A. Hayden, Bounds on the deviation of discrete-time Markov chains from their mean-field model, Perform. Eval. 70 (10) (2013) 736–749.