On-the-fly Fast Mean-Field Model-Checking

Diego Latella[1(✉)], Michele Loreti[2], and Mieke Massink[1]

[1] Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy
Diego.Latella@isti.cnr.it
[2] Università di Firenze, Firenze, Italy

Abstract. A novel, scalable, on-the-fly model-checking procedure is presented 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 shown and some results of the application of a prototype implementation of the FlyFast model-checker are presented.

Keywords: Probabilistic model-checking · On-the-fly model-checking · Mean-field approximation · Discrete time Markov chains

1 Introduction

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. Despite the success of model-checking procedures, their scalability have always been a concern due to the potential combinatorial explosion of the state space that needs to be searched.

The main contribution of this paper is a novel model-checking procedure, based on an original combination of local, on-the-fly model-checking techniques and mean field approximation in discrete time [26]. The procedure can be used to verify bounded PCTL [18] properties of selected individuals in the context of systems consisting of a large number of similar but independent interacting objects. It is scalable in the sense that it is insensitive to the size of the population the system consists of. The asymptotic correctness of the model-checking procedure is proven and a prototype implementation of the model-checker, FlyFast, is applied to a bench-mark example from computer epidemics that was also studied extensively in [7], to which we refer for a detailed discussion. To the best of our knowledge, this is the first implementation of an *on-the-fly mean field* model-checker for *discrete time, probabilistic, time-synchronous* models.

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

M. Abadi and A. Lluch Lafuente (Eds.): TGC 2013, LNCS 8358, pp. 297–314, 2014.
DOI: 10.1007/978-3-319-05119-2_17,

Following the approach in [26] we consider a model for interacting objects, where the evolution of each object is given by a finite state discrete time Markov chain. The transition matrix of each object may depend on the distribution of states of all objects in the system. Each object can be in one of its local states at any point in time and all objects proceed in discrete time and in a clock-synchronous fashion. When the number of objects is large, the overall behaviour of the system in terms of its 'occupancy measure', i.e. the fraction of objects that are in a particular local state at a particular time, can be approximated by the (deterministic) solution of a *difference equation* which is called the 'mean field'[1]. This convergence result has been extended in [26] 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 [26], to compute the average overall behaviour of the system and that of individual objects in the context of the overall system, combines well with an *on-the-fly* probabilistic model-checking procedure for the verification of bounded PCTL formulas addressing selected objects of interest[2]. An on-the-fly recursive approach also provides a natural way to address nested path formulae and time-varying truth values of such formulae. The algorithm presented in this paper is parametric w.r.t. the semantic interpretation of the language. In particular we present two different interpretations; one based on the standard, *exact probabilistic semantics* of a simple probabilistic population description language, and the other one on the *mean-field approximation in discrete time* of such a semantics. The latter is the main contribution of the current paper. The considered PCTL formulae can be extended along the lines proposed in [21,22] with properties that address the overall status of the system. We show a simple instance of that.

The models we consider are also known as SIO-models (System of Independent Objects) [7]. These are time-synchronous models in which each object performs a probabilistic step in each discrete time unit, possibly looping 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 Related Work

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 [5,11].

[1] The term 'mean field' has its origin in statistical physics and is sometimes used with slightly different meaning in the literature. Here we intend the meaning as defined in [26].

[2] 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 with time.

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. [10]) and in a stochastic setting (see e.g. [2,23]). The set of states that satisfy a formula is constructed recursively in a *bottom-up* fashion following the syntactic structure of the formula. Depending on the particular formula to verify, usually the underlying model can be reduced to fewer states before the algorithm is applied. Moreover, as is shown e.g. in [2] for stochastic model checking, the model checking algorithm can be reduced to combinations of existing well-known and optimised algorithms for CTMCs such as transient analysis.

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. [5,11, 15,20]). 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 [13] and [17]. The former is a probabilistic model checker for bounded PCTL formulas. The latter uses an on-the-fly approach to detect a maximal relevant search depth in an infinite state space and then uses a *global* model-checking approach to verify bounded CSL [1,2] formulas in a continuous time setting on the selected subset of states. An on-the-fly approach by itself however, does not solve the challenging scalability problems that arise in truly large parallel systems, such as collective adaptive systems, e.g., gossip protocols [9], self-organised collective decision making [28], computer epidemics [8] 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 [16,19] approximate probabilistic model-checking is introduced. This is a form of statistical model-checking that consists in the generation of random executions of an *a priori* established maximal length. On each execution the property of interest is checked and statistics are performed over the outcomes. The number of executions required for a reliable result depends on the maximal error-margin of interest. The approach relies on the analysis of individual execution traces rather than a full state space exploration and is therefore memory-efficient. However, the number of execution traces that may be required to reach a desired accuracy may be large and

therefore time-consuming. The approach works for general models, i.e., not necessarily populations of similar objects, but is not independent of the number of objects involved.

To analyse properties of large scale mobile communication networks mean field approximations in discrete time have also been used e.g., in Bakshi et al. [3]. 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 [26], but not its extension to analyse individual behaviour in the context of a large population, nor its exploitation in model-checking algorithms.

In Chaintreau et al. [9], 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 [4].

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 [21], but no model-checking algorithms were presented. Follow-up work on the above mentioned approach can be found in [22] which relies on earlier results on fluid model checking by Bortolussi and Hillston [6]. In the latter a *global CSL* model-checking procedure is proposed for the verification of properties of a selection of individuals in a population. This work is perhaps the most closely related to our work, however their procedure exploits mean field convergence and fast simulation [12,14] in a *continuous* time setting rather than in a discrete time setting and is based on an *interleaving* model of computation, rather than a clock-synchronous one; furthermore, a *global* model-checking approach, rather than an on-the-fly approach, is followed. The modelling language used in [6] is PEPA. Earlier work by Stefanek et al. [29] on the use of mean field convergence in *continuous time* for grouped PEPA has investigated the quality of the convergence results when the related differential equations are derived directly from the process algebraic model. Potential issues with accuracy were found concerning the parallel composition operator of PEPA that involves a (non-linear) minimum function applied to rates originating from synchronising populations. This could, in some circumstances, give rise to inaccuracies in the approximation. It is however possible to detect such situations.

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

In this section we recall the definition of the *time bounded* fragment of PCTL[3] and we present an on-the-fly model-checking algorithm. The algorithm is parametric in the sense that it can be used for different languages and semantic

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

Table 1. Satisfaction relation for Time Bounded PCTL.

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

interpretations. In this paper we use two instantiations of the algorithm; one is on a DTMC semantics of a simple language of object populations (Sect. 4) and the other is on a mean-field approximation semantics of the same language, for "fast model-checking" (Sect. 5). For the sake of readability, we present only a schema of the algorithm for time *bounded* PCTL, that is the same as that proposed in [13]. The interested reader is referred to [25] where a novel algorithm is defined and implemented for *the full* logic.

3.1 Time Bounded PCTL

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

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

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

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

3.2 On-the-fly PCTL Model-Checking Algorithm

In this section we introduce a local on-the-fly model-checking algorithm for time-bounded PCTL formulae. 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

Table 2. Function Check

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

are generated. For each of them it updates the information about the satisfaction of the formula that is checked. In this way, only that part of the state space is generated that can provide information on the satisfaction of the formula and irrelevant parts are not taken into consideration.

In the case of probabilistic process population languages, for large populations, a mean-field approximated semantics can be defined. In Sect. 5 we show how a drastic reduction of the state space can be obtained, by using the same algorithm on such semantic models. We call such a combined use of on-the-fly model-checking and mean-field semantics "Fast model-checking" after "Fast simulation", introduced in [26].

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

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

The local model-checking algorithm is defined as a function, Check, shown in Table 2. On atomic state-formulae, the function returns the value of lab_eval; when given a non-atomic state-formula, Check calls itself recursively on sub-formulae, in case they are state-formulae, 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.

Table 3. Function CheckPath

```
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
```

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 is easily proved by induction on Φ [25].

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

4 Modelling Language

In this section we define a simple population description language. The language is essentially a textual version of the graphical notation used in [26]. A *system* is defined as a population of N identical interacting processes or objects[4]. 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

[4] In [26] *object* is used instead of *process*. We consider the two terms synonyms here.

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.[5]

Syntax. Let $\mathcal{A}$ be a denumerable non-empty set of *actions*, ranged over by $a, a', a_1, \ldots$ and $\mathcal{S}$ be a denumerable non-empty set of *state constants*, ranged over by $C, C', C_1, \ldots$ An *object specification* Δ is a set $\{D_i\}_{i \in I}$, for finite index set I, where each *state definition* D_i has the form $C_i := \sum_{j \in J_i} a_{ij}.C_{ij}$, with J_i a finite index set, states $C_i, C_{ij} \in \mathcal{S}$, and $a_{ij} \in \mathcal{A}$, for $i \in I$ and $j \in J_i$. Intuitively, the notation $\sum_{j \in J_i} a_{ij}.C_{ij}$ is to be intended as the n-ary extension of the standard process algebraic binary non-deterministic choice operator. We require that $a_{ij} \neq a_{ij'}$, for $j \neq j'$ and that for each state constant C_{ij} occurring in the r.h.s. of a state definition D_i of Δ there is a *unique* $k \in I$ such that C_{ij} is the l.h.s. of D_k.

Example 1 (*An epidemic model* [7]). 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:

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

The set of all actions occurring in object specification Δ is denoted by $\mathcal{A}_\Delta$. Similarly, the set of states is denoted by $\mathcal{S}_\Delta$, ranged over by $c, c', c_1 \cdots$. In Example 1, we have $\mathcal{A}_{\texttt{EM}} = \{\texttt{inf\_ext}, \texttt{inf\_sus}, \texttt{activate}, \texttt{patch}, \texttt{loss}\}$ and $\mathcal{S}_{\texttt{EM}} = \{\texttt{S}, \texttt{E}, \texttt{I}, \texttt{R}\}$. A system is assumed composed of N interacting instances of an object. Interaction among objects is modelled probabilistically, as described below. Each action in $\mathcal{A}_\Delta$ is assigned a probability value, that may depend on the global state of the system. This is achieved by means of a *probability function definition*, that takes the following form: $a::E$, where $a \in \mathcal{A}_\Delta$ and E is an expression, i.e. an element of Exp, defined according to the following grammar:

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

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

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

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

`inf_ext` :: α_e;
`inf_sus` :: $\alpha_i * (\mathsf{frc}\, I)$;
`activate` :: α_a;
`patch` :: α_r;
`loss` :: α_s;

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

A *system specification* is a triple $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$ where Δ is an object specification, A is a set of probability function definitions containing exactly one definition for each $a \in \mathcal{A}_\Delta$, and $\mathbf{C}_0 = \langle c_{0_1}, \ldots, c_{0_N} \rangle$ is the *initial system state*, with $c_{0_n} \in \mathcal{S}_\Delta$, for $n = 1 \ldots N$; we say that N is the *population size*[6]; in the sequel, we will omit the explicit indication of the size N in $\langle \Delta, A, \mathbf{C}_0 \rangle^{(N)}$, and elements thereof or related functions, writing simply $\langle \Delta, A, \mathbf{C}_0 \rangle$, when this cannot cause confusion.

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

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

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

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

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

A probability function definition $a::E$ associates a real value to action a by evaluating E in the current global state, via the interpretation function $\mathscr{E}$. In practice the occupancy measure representation of the state is used in $\mathscr{E}$.

The expressions interpretation function $\mathscr{E} : \mathrm{Exp} \to \mathcal{U}^S \to \mathbb{R}$ is defined as usual:

$\mathscr{E}[\![v]\!]_{\mathbf{m}} = v$
$\mathscr{E}[\![\mathsf{frc}\, C_i]\!]_{\mathbf{m}} = \mathbf{m}_{[i]}$

[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.

$\mathscr{E}[\![\langle \text{uop} \rangle\, E]\!]_{\mathbf{m}} = \langle \text{uop} \rangle\, (\mathscr{E}[\![E]\!]_{\mathbf{m}})$
$\mathscr{E}[\![E_1\, \langle \text{bop} \rangle\, E_2]\!]_{\mathbf{m}} = (\mathscr{E}[\![E_1]\!]_{\mathbf{m}})\, \langle \text{bop} \rangle\, (\mathscr{E}[\![E_2]\!]_{\mathbf{m}})$
$\mathscr{E}[\![(E)]\!]_{\mathbf{m}} = (\mathscr{E}[\![(E)]\!]_{\mathbf{m}})$

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

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

$$\mathbf{K}^{(N)}(\mathbf{m})_{c,c'} = \begin{cases} \sum_{a:c \overset{a}{\rightarrowtail} c'} \pi(\mathbf{m}, a), \text{ if } c \neq c', \\ 1 - \sum_{a \in I(c)} \pi(\mathbf{m}, a), \text{ if } c = c'. \end{cases} \tag{2}$$

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

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

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

$$\mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} = \mathbf{S}^{(N)}(\mathbf{M}^{(N)}(\mathbf{C}))_{\mathbf{C},\mathbf{C}'} \tag{3}$$

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

$$\mathbb{P}\{\mathbf{M}^{(N)}(t+1) = \mathbf{m}' \mid \mathbf{M}^{(N)}(t) = \mathbf{m}\} = \sum_{\mathbf{C}' : \mathbf{M}^{(N)}(\mathbf{C}') = \mathbf{m}'} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \tag{4}$$

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

PCTL local Model-checking. For the purpose of expressing system properties in PCTL, we partition the set of atomic propositions $\mathscr{P}$ into sets $\mathscr{P}_1$and $\mathscr{P}_g$. Given system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$, we extend it with a *state labelling function definition* that associates each state $c \in \mathcal{S}_\Delta$ with a (possibly empty)

finite set $\ell_1(c)$ of propositions from $\mathscr{P}_1$. We extend ℓ_1 to global states with $\ell_1(\langle c_1, \ldots, c_N\rangle) = \ell_1(c_1)$; this way, we can express *local* properties of the first object in the system, in the *context* of the complete population[7]. In order to express also (a limited class of) *global* properties of the population, we use set $\mathscr{P}_g$. The system specification is further enriched by associating labels $a \in \mathscr{P}_g$ with expressions bexp in the class BExp of restricted boolean expressions. We assume a sublanguage of function specifications be given[8] and for function symbol F, $\mathscr{E}[\![F]\!]_{\mathbf{m}} : [0,1]^q \mapsto \mathbb{R}$ continuous in $[0,1]^q$, with $\mathscr{E}[\![F]\!]_{\mathbf{m}} = \mathscr{E}[\![F]\!]_{\mathbf{m}'}$ for all $\mathbf{m}, \mathbf{m}' \in \mathcal{U}^S$; then BExp is the set of expressions of the form $F(E_1, \ldots, E_q)\,\langle \text{relop}\rangle\, r$, where each E_j is of the form $\mathsf{frc}\, C$, $\langle \text{relop}\rangle \in \{>, <\}$, $r \in \mathbb{R}$ and $\mathscr{E}[\![F(E_1, \ldots, E_q)]\!]_{\mathbf{m}} = \mathscr{E}[\![F]\!]_{\mathbf{m}}(\mathscr{E}[\![E_1]\!]_{\mathbf{m}}, \ldots, \mathscr{E}[\![E_q]\!]_{\mathbf{m}})$.

We define the state global labelling function ℓ_g as

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

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

The definition of $Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$ as well as that of the satisfaction relation $\models_{\mathcal{D}^{(N)}}$ are obtained by instantiating those given in Sect. 3.1 to $\mathcal{D}^{(N)}$. For $\sigma \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, $\sigma[j]_{[n]}$ 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 $\mathscr{P}_1 \cup \mathscr{P}_g$. Function next is instantiated to the function $\mathsf{next}_{\mathcal{D}^{(N)}}$, where

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

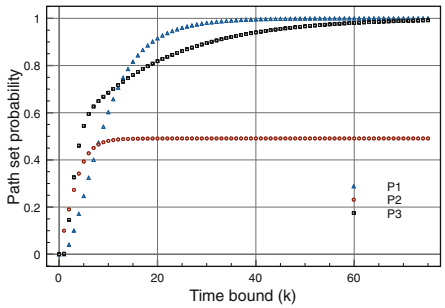
Given a vector $\mathbf{C}$, $\mathsf{next}_{\mathcal{D}^{(N)}}(\mathbf{C})$ computes a list corresponding to the positive elements of the row of matrix $\mathbf{P}^{(N)}$ associated with $\mathbf{C}$. Of course, only those elements of $\mathbf{P}^{(N)}$ that are necessary for $\mathsf{next}_{\mathcal{D}^{(N)}}$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

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

[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 extended 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] The specific features of the sublanguage are not relevant for the purposes of the present paper and we leave their treatment out 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:
$\alpha_e = 0.1$, $\alpha_i = 0.2$, $\alpha_r = 0.2$
$\alpha_a = 0.4$, $\alpha_s = 0.1$

Fig. 1. Exact model-checking results (left) and verification time (right).

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. 1 the result of exact PCTL model-checking of Example 1 is reported. On the left the probability of the set of paths that satisfy the path-formulae used in the three formulae 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 [23] and using exact on-the-fly PCTL model checking are presented[10], showing that the latter has comparable performance. Worst-case complexity of both algorithms are also comparable. The local model-checker has been instantiated with the model defined by the (exact) operational semantics of the language, where each state $\mathbf{C} \in \mathcal{S}_{\Delta}^{N}$ is a *global* system state. In Sect. 5 we instantiate the procedure with the mean-field, approximated, semantics of the language, leading to a scalable, 'fast', model-checker, insensitive to the population size.

5 Fast Mean-Field Model-Checking

Given a system specification $\langle \Delta, A, \mathbf{C}_0^{(N)} \rangle^{(N)}$ with initial state $\mathbf{C}_0$, we want to focus on the behaviour of the first object, starting in the initial state $\mathbf{C}_{0[1]}$,

[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://rap.dsi.unifi.it/~loreti/OFPMC/).

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

$$\mathbf{H}^{(N)}_{\langle c,\mathbf{m}\rangle,\langle c',\mathbf{m}'\rangle} = \sum_{\mathbf{C}' : \mathcal{H}^{(N)}(\mathbf{C}') = \langle c', \mathbf{m}'\rangle} \mathbf{P}^{(N)}_{\mathbf{C},\mathbf{C}'} \tag{5}$$

where $\mathbf{C}$ is such that $\mathcal{H}^{(N)}(\mathbf{C}) = \langle c, \mathbf{m}\rangle$.[11] The definitions of paths for state $\langle c, \mathbf{m}\rangle$ of $\mathcal{HD}^{(N)}$, $Paths_{\mathcal{HD}^{(N)}}(\langle c, \mathbf{m}\rangle)$, of $L_{\mathcal{HD}^{(N)}}(t)$ and of the satisfaction relation $\models_{\mathcal{HD}^{(N)}}$ of PCTL formulas against $\mathcal{HD}^{(N)}(t)$, are obtained by instantiating the relevant definitions of Sect. 3.1 to the model $\mathcal{HD}^{(N)}(t)$. Furthermore, we let $L_{\mathcal{HD}^{(N)}}(t, c) = \{\langle c', \mathbf{m}'\rangle \in \mathrm{L}_{\mathcal{HD}^{(N)}}(t) \mid c' = c\}$.

We extend mapping $\mathcal{H}^{(N)}$ to sets and paths in the obvious way: for set X of states, let $\mathcal{H}^{(N)}(X) = \{\mathcal{H}^{(N)}(x) \mid x \in X\}$, and for $\sigma \in Paths_{\mathcal{D}^{(N)}}(\mathbf{C}^{(N)})$, 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, and can be easily proved by induction on formulae Φ [24].

Lemma 1. *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 now consider the stochastic process $\mathcal{HD}(t)$ defined below, for $c_0, c, c' \in \mathcal{S}_\Delta$, $\boldsymbol{\mu}_0, \mathbf{m}, \mathbf{m}' \in \mathcal{U}^S$ and function $\mathbf{K}(\mathbf{m})_{c,c'}$, continuous in $\mathbf{m}$:

$$\begin{aligned} &\mathbb{P}\{\mathcal{HD}(0) = \langle c, \mathbf{m}\rangle\} = \boldsymbol{\delta}_{\langle c_0,\boldsymbol{\mu}_0\rangle}(\langle c, \mathbf{m}\rangle), \\ &\mathbb{P}\{\mathcal{HD}(t+1) = \langle c', \mathbf{m}'\rangle \,|\, \mathcal{HD}(t) = \langle c, \mathbf{m}\rangle\} = \begin{cases} \mathbf{K}(\mathbf{m})_{c,c'}, & \text{if } \mathbf{m}' = \mathbf{m}\cdot\mathbf{K}(\mathbf{m}) \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

The definition of the labeling function $\ell_{\mathcal{HD}}$ is the same as that of $\ell_{\mathcal{HD}^{(N)}}$. Note that $\mathcal{HD}$ is a DTMC with initial state $\langle c_0, \boldsymbol{\mu}_0\rangle$; memoryless-ness as well as time homogeneity directly follow from the definition of the process (6). The definitions of paths for state $\langle c, \mathbf{m}\rangle$ of $\mathcal{HD}$, $Paths_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle)$, of $L_{\mathcal{HD}}(t)$ and of the satisfaction relation $\models_{\mathcal{HD}}$ of PCTL formulas against $\mathcal{HD}(t)$ are obtained by instantiating the relevant definitions of Sect. 3.1 to the model $\mathcal{HD}(t)$. Furthermore, define function $\boldsymbol{\mu}(t)$ as follows: $\boldsymbol{\mu}(0) = \boldsymbol{\mu}_0$ and $\boldsymbol{\mu}(t+1) = \boldsymbol{\mu}(t)\cdot\mathbf{K}(\boldsymbol{\mu}(t))$; then, for $t \geq 0$ and for $\langle c, \mathbf{m}\rangle \in L_{\mathcal{HD}}(t)$ we have $\mathbf{m} = \boldsymbol{\mu}(t)$.

In the following we use the fundamental result stated below, due to Le Boudec et al. [26].

Theorem 4.1 of [26] *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*

[11] With a similar argument as for definition (4), noting that $\mathbf{M}^{(N)}(\mathbf{C}) = \mathbf{M}^{(N)}(\mathbf{C}'')$ and $\mathbf{C}_{[1]} = \mathbf{C}''_{[1]}$, it can be easily seen that also definition (5) is a good definition.

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

Remark 1. We observe that, as direct consequence of Theorem 4.1 of [26] and of the restrictions on the definition of BExp, for any fixed t and for all $\epsilon > 0$, there exists $\bar{N}$ such that, for all $N \geq \bar{N}$, almost surely

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

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

In the rest of the paper we will focus on sequences $\left(\langle \Delta, A, \mathbf{C}_0\rangle^{(N)}\right)_{N\geq N_0}$ of system specifications, for some $N_0 > 0$. In particular, we will consider only sequences $\left(\mathcal{HD}^{(N)}(t)\right)_{N\geq N_0}$ such that for all $N \geq N_0$, $\mathbf{C}_{0[1]}^{(N)} = \mathbf{C}_{0[1]}^{(N_0)}$; in other words we want the population size increase 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 Eq. (6) 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 [26], when considering sequences $\left(\mathcal{HD}^{(N)}(t)\right)_{N\geq N_0}$ as above (see also Remark 1):

Corollary 1. *Under the assumptions of Theorem 4.1 of [26], for any fixed* t, *almost surely,* $\lim_{N\to\infty} \mathcal{HD}^{(N)}(t) = \mathcal{HD}(t)$. •

Remark 2. A consequence of Corollary 1 is that, under the assumptions of Theorem 4.1 of [26], 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, that relies on the notion of *formulae safety*, with w.r.t. $\mathcal{HD}(t)$: a formula Φ is *safe* for a model $\mathcal{M}$ iff for all sub-formulae Φ' of Φ and states s of $\mathcal{M}$, if Φ' is of the form $\mathcal{P}_{\bowtie p}(\varphi)$ then $\mathbb{P}\{\eta \in Paths_{\mathcal{M}}(s) \mid \eta \models_{\mathcal{M}} \varphi\} \neq p$.

The theorem, together with Theorem 1 and Lemma 1, establishes the formal relationship between the satisfaction relation on the exact semantics of the

language and that on its mean-field approximation, thus justifying the fast local model-checking instantiation we will show in the sequel.

Theorem 2. *Under the assumptions of Theorem 4.1 of [26], 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.

For brevity, we show only the case for $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$; for the complete proof we refer to [24]. 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,

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

iff

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

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 implies the assert.

$\mathbb{P}\{\rho \in Paths_{\mathcal{HD}^{(N)}}(\mathcal{H}^{(N)}(\mathbf{C})) \mid \rho \models_{\mathcal{HD}^{(N)}} \mathcal{X}\ \Phi\}$ is defined as

$$p^{(N)}_{\mathbf{H}} = \sum_{\mathcal{H}^{(N)}(\mathbf{C}'):\mathcal{H}^{(N)}(\mathbf{C}') \models_{\mathcal{HD}^{(N)}} \Phi} \mathbf{H}^{(N)}_{\mathcal{H}^{(N)}(\mathbf{C}),\mathcal{H}^{(N)}(\mathbf{C}')} \tag{7}$$

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

$$p(t)_{\mathbf{K}} = \sum_{\mathbf{C}'_{[1]}:\langle \mathbf{C}'_{[1]}, \boldsymbol{\mu}(t+1)\rangle \models_{\mathcal{HD}} \Phi} \mathbf{K}(\boldsymbol{\mu}(t))_{\mathbf{C}_{[1]},\mathbf{C}'_{[1]}}. \tag{8}$$

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 1, 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 (7)

approaches the value $p(t)_{\mathbf{K}}$ of sum (8). Finally, safety of $\mathcal{P}_{\bowtie p}(\mathcal{X}\ \Phi)$, implies that the value $p(t)_{\mathbf{K}}$ of (8) 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. □

Finally, using Lemma 1 we get the following

Corollary 2. *Under the assumptions of Theorem 4.1 of [26], 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 $\langle \mathbf{C}^{(N)}_{[1]}, \boldsymbol{\mu}(t)\rangle \models_{\mathcal{HD}} \Phi$.* •

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

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

with $\mathbf{K}(\mathbf{m})_{c,c'}$ as in Theorem 4.1 of [26]; lab_eval is instantiated as expected: $\mathsf{lab\_eval}_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle, a) = a \in \ell_{\mathcal{HD}}(\langle c, \mathbf{m}\rangle)$. The instantiation is implemented in FlyFast.

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

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

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

Example 4 (FlyFast *results).* Figure 2 shows the result of FlyFast on the model of Example 1 for the first object of a large population of objects, each initially in state S. In Fig. 2 (left) the same properties are considered as in Example 3. The analysis takes less than a second and is *insensitive* to the total population size. Fig. 2 (right) shows how the probability measure of the set of paths satisfying formula $true\ \mathcal{U}^{\leq k}\ (!e \wedge !i \wedge \mathcal{P}_{>0.3}(\ true\ \mathcal{U}^{\leq 5}\ i\ ))$ of property *P3* on page 12, (for $k = 3$), changes for initial time $t0$ varying from 0 to 10.

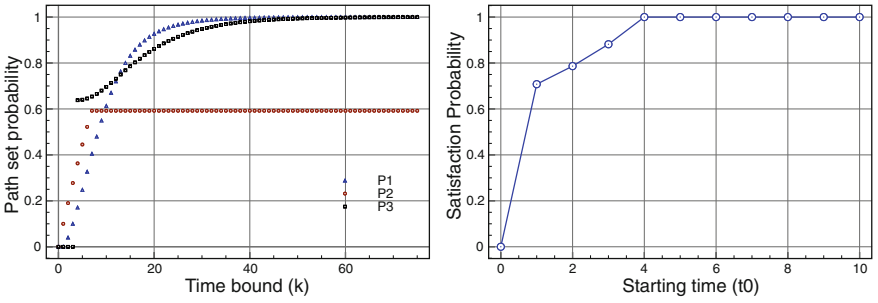


Fig. 2. Fast model-checking results.

6 Conclusions and Future Work

In this paper we have presented a fast PCTL model-checking approach that builds upon *local, on-the-fly* model-checking and *mean-field* approximation, allowing for scalable analysis of selected objects in the context of very large systems. The model-checking algorithm is parametric w.r.t. the specific semantic model of interest. We presented related correctness results, an example of application of a prototype implementation and briefly discussed complexity of the algorithm. The results can be trivially extended in order to consider multiple selected objects. Following approaches similar to those presented in [26], we plan to extend our work to heterogeneous systems and systems with memory. We are interested in extensions that address spatial distribution of objects as well as more expressive logics, combining local and global properties, and languages (e.g. [22,27]) and to study the exact relation between mean field convergence results for continuous interleaving models and discrete, time-synchronous ones.

References

1. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model checking continuous time Markov chains. ACM Trans. Comput. Logic **1**(1), 162–170 (2000)
2. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE Trans. Softw. Eng. **29**(6), 524–541 (2003). IEEE CS
3. Bakhshi, R., Endrullis, J., Endrullis, S., Fokkink, W., Haverkort, B.: Automating the mean-field method for large dynamic gossip networks. In: QEST 2010, pp. 241–250. IEEE Computer Society (2010)
4. Benaïm, M., Le Boudec, J.Y.: A class of mean field interaction models for computer and communication systems. Perform. Eval. **65**(11–12), 823–838 (2008)
5. Bhat, G., Cleaveland, R., Grumberg, O.: Efficient on-the-fly model checking for CTL*. In: LICS, pp. 388–397. IEEE Computer Society (1995)
6. Bortolussi, L., Hillston, J.: Fluid model checking. In: Koutny, M., Ulidowski, I. (eds.) CONCUR. LNCS, vol. 7454, pp. 333–347. Springer, Heidelberg (2012)
7. Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective system behaviour: a tutorial. Perform. Eval. **70**(5), 317–349 (2013). http://www.sciencedirect.com/science/article/pii/S0166531613000023
8. Bradley, J.T., Gilmore, S.T., Hillston, J.: Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. J. Comput. Syst. Sci. **74**(6), 1013–1032 (2008)
9. Chaintreau, A., Le Boudec, J.Y., Ristanovic, N.: The age of gossip: spatial mean field regime. In: Douceur, J.R., Greenberg, A.G., Bonald, T., Nieh, J. (eds.) SIGMETRICS/Performance, pp. 109–120. ACM, Seattle (2009)
10. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8**(2), 244–263 (1986)
11. Courcoubetis, C., Vardi, M., Wolper, P., Yannakakis, M.: Memory-efficient algorithms for the verification of temporal properties. Form. Methods Syst. Des. **1**(2–3), 275–288 (1992)

12. Darling, R., Norris, J.: Differential equation approximations for Markov chains. Probab. Surv. **5**, 37–79 (2008)
13. Della Penna, G., Intrigila, B., Melatti, I., Tronci, E., Zilli, M.V.: Bounded probabilistic model checking with the murφ verifier. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 214–229. Springer, Heidelberg (2004)
14. Gast, N., Gaujal, B.: A mean field model of work stealing in large-scale systems. In: Misra, V., Barford, P., Squillante, M.S. (eds.) SIGMETRICS. pp. 13–24. ACM (2010)
15. Gnesi, S., Mazzanti, F.: An abstract, on the fly framework for the verification of service-oriented systems. In: Wirsing, M., Hölzl, M. (eds.) SENSORIA. LNCS, vol. 6582, pp. 390–407. Springer, Heidelberg (2011)
16. Guirado, G., Hérault, T., Lassaigne, R., Peyronnet, S.: Distribution, approximation and probabilistic model checking. Electr. Notes Theor. Comput. Sci. **135**(2), 19–30 (2006). http://dx.doi.org/10.1016/j.entcs.2005.10.016
17. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: INFAMY: an infinite-state Markov model checker. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 641–647. Springer, Heidelberg (2009)
18. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects Comput. **6**, 512–535 (1994)
19. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 73–84. Springer, Heidelberg (2004)
20. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Reading (2004)
21. Kolesnichenko, A., Remke, A., de Boer, P.T.: A logic for model-checking of mean-field models. Technical report TR-CTIT-12-11. http://doc.utwente.nl/80267/ (2012)
22. Kolesnichenko, A., Remke, A., de Boer, P.T.: A logic for model-checking of mean-field models. In: Dependable Systems and Networks DSN13 (2013)
23. Kwiatkowska, M., Norman, G., Parker, D.: Probabilistic symbolic model checking using PRISM: a Hybrid approach. STTT **6**(2), 128–142 (2004)
24. Latella, D., Loreti, M., Massink, M.: On-the-fly fast mean-field model-checking: full version. Technical report. http://arxiv.org/abs/1312.3416 (2013)
25. Latella, D., Loreti, M., Massink, M.: On-the-fly probabilistic model-checking: full version. Technical report. http://goo.gl/uVkPP6/ (2013)
26. Le Boudec, J.Y., McDonald, D., Mundinger, J.: A generic mean field convergence result for systems of interacting objects. In: QEST07. pp. 3–18. IEEE Computer Society Press (2007). ISBN 978-0-7695-2883-0
27. McCaig, C., Norman, R., Shankland, C.: From individuals to populations: a mean field semantics for process algebra. Theor. Comput. Sci. **412**(17), 1557–1580 (2011)
28. Montes de Oca, M.A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., Dorigo, M.: Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making. Swarm Intell. **5**(3–4), 305–327 (2011)
29. Stefanek, A., Hayden, R.A., Bradley, J.T.: A new tool for the performance analysis of massively parallel computer systems. In: QAPL 2010. EPTCS, vol. 28. pp. 159–181 (2010)