CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Edinburgh Research Explorer

THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

Checking Individual Agent Behaviours in Markov Population Models by Fluid Approximation

Citation for published version:
Bortolussi, L & Hillston, J 2013, Checking Individual Agent Behaviours in Markov Population Models by Fluid Approximation. in M Bernardo, E Vink, A Pierro & H Wiklicky (eds), Formal Methods for Dynamical Systems: 13th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2013, Bertinoro, Italy, June 17-22, 2013. Advanced Lectures. Lecture Notes in Computer Science, vol. 7938, Springer-Verlag GmbH, pp. 113-149. DOI: 10.1007/978-3-642-38874-3_4

Digital Object Identifier (DOI):
10.1007/978-3-642-38874-3_4

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Formal Methods for Dynamical Systems

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. Apr. 2019

Bortolussi, L., & Hillston, J. (2013). Checking Individual Agent Behaviours in Markov Population Models by Fluid Approximation. In Bernardo, M., Vink, E., Pierro, A., & Wiklicky, H. (Eds.), Formal Methods for Dynamical Systems. (pp. 113-149). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. doi: 10.1007/978-3-642-38874-3_4

Checking Individual Agent Behaviours in Markov Population Models by Fluid Approximation

Luca Bortolussi[1] and Jane Hillston[2]

[1] Department of Mathematics and Geosciences
University of Trieste, Italy.
CNR/ISTI, Pisa, Italy.
luca@dmi.units.it
[2] Laboratory for the Foundations of Computer Science,
School of Informatics, University of Edinburgh, UK.
jane.hillston@ed.ac.uk

Abstract. In this chapter, we will describe, in a tutorial style, recent work on the use of fluid approximation techniques in the context of stochastic model checking. We will discuss the theoretical background and the algorithms working out an example.
This approach is designed for population models, in which a (large) number of individual agents interact, which give rise to continuous time Markov chain (CTMC) models with a very large state space. We then focus on properties of individual agents in the system, specified by Continuous Stochastic Logic (CSL) formulae, and use fluid approximation techniques (specifically, the so called fast simulation) to check those properties. We will show that verification of such CSL formulae reduces to the computation of reachability probabilities in a special kind of time-inhomogeneous CTMC with a small state space, in which both the rates and the structure of the CTMC can change (discontinuously) with time. In this tutorial, we will discuss only briefly the theoretical issues behind the approach, like the decidability of the method and the consistency of the approximation scheme.

Keywords: Stochastic model checking; fluid approximation; mean field approximation; reachability probability; time-inhomogeneous Continuous Time Markov Chains

1 Introduction

In recent years there has been a growing interest in the use of mean field or fluid approximation techniques in the analysis of large scale models of dynamic behaviour. In particular there have been a number of attempts to integrate these mathematical approximations with formal modelling techniques originating in theoretecial computer science. Specifically, fluid approximation techniques are cross-fertilising with quantitative formal methods, mainly Stochastic Process

Algebras (SPA), giving birth to a new class of quantitative analysis techniques for large scale population models, described by continuous time Markov chains (CTMC) [7, 8, 31, 27, 16, 9].

Whereas process algebra models can be viewed as agent-based descriptions in which the behaviour of each entity in the system is described in detail, fluid or mean field methods focus instead on the more abstract population view of the system. In order to move from the process algebra description to a mean field approximation, individuality of agents must be abstracted and a collective or counting abstraction employed [27]. This leads to an aggregated state representation, in which the system is described by variables counting the number of agents in each possible state. Then, the discrete state space of the CTMC is approximated by a continuous one, and the stochastic dynamics of the process is approximated by a deterministic one, meaning that states no longer evolve through discrete jumps based on the interleaved state changes of individuals. Instead the state variables are assumed to be subject to continuous change expressed by means of a set of differential equations. The correctness of this approach is guaranteed by limit theorems [34, 21, 22], showing the convergence of the CTMC to the fluid ODE for systems of increasing population size.

On the one hand, fluid approximation has been used in the context of stochastic process algebras mainly to approximately compute the average transient dynamics, or to approximate the average steady state, when possible [49, 48]. It has also been used to estimate fluid passage times [25, 26].

On the other hand, stochastic process algebra models can also be analysed using quantitative model checking. These techniques have a long tradition in computer science and provide powerful ways of querying a model and extracting information about its behaviour. In the case of stochastic model checking, there are some consolidated approaches, principally based on checking Continuous Stochastic Logic (CSL) formulae [5, 4, 45], and these are supported by software tools which are in widespread use [36, 37]. All these methods, however, suffer (in a more or less relevant way) from the curse of state space explosion, which severely hampers their practical applicability particularly for population models.

One possibility to mitigate the state space explosion problem is to combine fluid approximation techniques with stochastic model checking, obtaining efficient approximate algorithms for checking formulae against population models. In this tutorial, we will present a first attempt in this direction [12, 13], in which mean field approximation is used to carry out approximate model checking of behaviours of individual agents in large population models, specified as CSL formulae. This is made possible by a corollary of the fluid convergence theorems, known by the name of *fast simulation* [24, 22], which characterises the limit behaviour of a single agent in terms of the solution of the fluid equation: the agent senses the rest of the population only through its "average" evolution, as given by the fluid equation. The idea of [12, 13] is to check individual properties in this limit model, rather than on the full model with N interacting agents. In fact, extracting metrics from the description of the global system can be extremely expensive from a computational point of view. Fast simulation, instead, is a very

compact abstraction of the system and the evolution of a single agent (or of a subset of agents) can be computed efficiently, by decoupling its evolution from the evolution of its environment and hence reducing the dimensionality of the state space by several orders of magnitude. A central feature of the abstraction based on fluid approximation is that the limit model of an individual agent is a time-inhomogeneous CTMC (ICTMC). This introduces some additional complexity in the approach, as model checking of ICTMC is far more difficult than the homogeneous-time counterpart. Nevertheless we can learn from the previous techniques for model-checking CSL properties on time-homogeneous CTMC, and develop suitable approaches for ICTMCs.

In this tutorial, we will present the work of [12, 13] in detail, using a network epidemic model as a simple running example. We will start by setting the context by presenting a simple modelling language for population CTMC (Section 2), discussing CSL and properties for individual agent (Section 3), and introducing some fundamental concepts about fluid approximation and fast simulation (Section 4). We will then turn to explain in detail the model checking procedure for ICTMC, considering first non-nested properties (Section 5), and then turning to nested CSL formulae (Section 7). The difficulty in this case is that the truth of a formula depends on the time at which the formula is evaluated, hence we need algorithms to compute this functional dependence (Section 6). The algorithms to model check nested formulae are presented only informally, by means of the running example (Section 7). We also discuss briefly two theoretical aspects of the work in [12, 13], namely decidability of the model checking algorithm for ICTMC and convergence of the truth value of CSL formulae for an individual in a system with a finite population level to the truth for the limit individual model (Section 8). We discuss the related work in Section 9 and finally, we sketch some conclusions in Section 10.

2 Population Models

In this section, we will introduce a simple language to construct Markov models of populations of interacting agents. We will consider models of processes evolving in continuous time, although a similar theory can be considered for discrete-time models (see, for instance, [14]). In principle, we can have different classes of agents, and many agents for each class in the system. Furthermore, the number of agents can change at runtime, due to birth or death events. Models of this kind include computer networks, where each node (e.g. server, client) of the network is an agent [38], biological systems (in which molecules are the agents) [47], ecological systems (in which individual animals are the agents) [11], and so on. However, to keep notation simple, we will assume here that the number of agents is constant and equal to N (making a closed world assumption). Furthermore, in the notation we do not distinguish between different classes of agents.

In particular, let us assume that each agent is a finite state machine, with internal states taken from a finite set S, and labelled by integers: $S = \{1, 2, \ldots, n\}$.

We have a population of N agents, and denote the state of agent i at time t, for $i = 1, \ldots, N$, by $Y_i^{(N)}(t) \in S$. Note that we made explicit the dependence on N, the total population size.
A configuration of a system is thus represented by the tuple $(Y_1^{(N)}, \ldots, Y_N^{(N)})$. This represention is based on treating each agent as a distinct individual with identity conferred by the position in the vector. However, when dealing with population models, it is customary to assume that single agents in the same internal state cannot be distinguished, hence we can move from the *individual representation* to the *collective representation* by introducing n variables counting how many agents are in each state. Hence, we define

$$X_j^{(N)} = \sum_{i=1}^{N} \mathbf{1}\{Y_i^{(N)} = j\}. \tag{1}$$

Note that the vector $\mathbf{X}^{(N)} = (X_1^{(N)}, \ldots, X_n^{(N)})$ has a dimension independent of N, and will be equivalently referred to as the collective, population, or counting vector. The domain of each variable $X_j^{(N)}$ is obviously $\{0, \ldots, N\}$, and, by the closed world assumption, it holds that $\sum_{j=1}^{n} X_j^{(N)} = N$. Let us denote with $\mathcal{S}^{(N)}$ the subset of vectors of $\{1, \ldots, N\}^n$ that satisfy such constraint.

Up to now, we just described the state space of our population models. In order to capture their dynamics, we will specify a set of possible events, or transitions, that can change the state of the system. Each such event will involve just a small, fixed, number of agents, usually one or two, but we will in any case describe it from the perspective of the collective system.
Events are stochastic, and take an exponentially distributed time to happen, with a rate depending on the current global state of the system. Hence, each event will be specified by a rate function, and by a set of update rules, telling us how many and which agents are involved and how they will change state.
The set of events, or transitions, $\mathcal{T}^{(N)}$, is made up of elements $\tau \in \mathcal{T}^{(N)}$, which are pairs $\tau = (R_\tau, r_\tau^{(N)})$. More specifically, R_τ is a *multi-set* of *update rules* of the form $i \to j$, specifying that an agent changes state from i to j when the event fires. As R_τ is a multiset, we can describe events in which two or more agents in state i synchronise and change state to j. The exact number of agents synchronising can be extracted looking at the multiplicity of rule $i \to j$ in R_τ; let us denote such a number by $m_{\tau, i \to j}$. Note also that R_τ is independent of N, so that each transition involves a finite and fixed number of individuals.
In order to model the effect of event τ on the population vector, we will construct from R_τ the *update vector* $\mathbf{v}_\tau$ in the following way:

$$\mathbf{v}_{\tau,i} = \sum_{(i \to j) \in R_\tau} m_{\tau, i \to j} \mathbf{e}_j - \sum_{(i \to j) \in R_\tau} m_{\tau, i \to j} \mathbf{e}_i,$$

where $\mathbf{e}_i$ is the vector equal to one in position i and zero elsewhere. Then, event τ changes the state from $\mathbf{X}^{(N)}$ to $\mathbf{X}^{(N)} + \mathbf{v}_\tau$.
The other component of event τ is the rate function $r_\tau^{(N)} : \mathcal{S}^{(N)} \to \mathbb{R}_{\geq 0}$, which

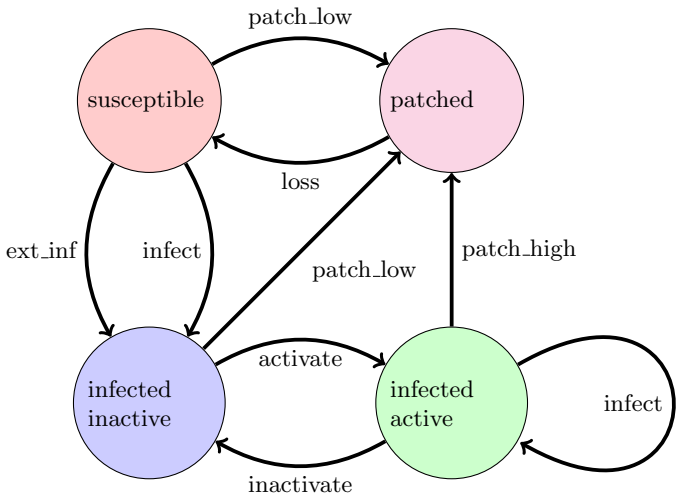


Fig. 1. States and transitions of a single computer in the p2p network epidemic model.

depends on the current state of the system, and specifies the speed of the corresponding transition. It is assumed to be equal to zero if there are not enough agents available to perform a τ transition, and it is required to be *Lipschitz continuous* (when interpreted as a function on real numbers).

All these bits of information are collected together in the population model $\mathcal{X}^{(N)} = (\mathbf{X}^{(N)}, \mathcal{T}^{(N)}, \mathbf{x}_0^{(N)})$, where $\mathbf{x}_0^{(N)}$ is the initial state. Given such a model, it is straightforward to construct the CTMC $\mathbf{X}^{(N)}(t)$ associated with it, exhibiting its infinitesimal generator matrix. First, its state space is $\mathcal{S}^{(N)}$, while its infinitesimal generator matrix $Q^{(N)}$ is the $|\mathcal{S}^{(N)}| \times |\mathcal{S}^{(N)}|$ matrix defined by

$$q_{\mathbf{x},\mathbf{x}'} = \sum \{ r_\tau(\mathbf{x}) \mid \tau \in \mathcal{T},\ \mathbf{x}' = \mathbf{x} + \mathbf{v}_\tau \}.$$

Remark 1. We note here that in this formalism we can still easily model multiple classes of agents. This can be done by partitioning the state space S into subsets, and allowing state changes only within a single subset. Furthermore, the rule set can be easily modified to include the possibility of birth and death events: we just need to add rules of the form $\emptyset \to i$ (birth of an agent in state i) or $i \to \emptyset$ (death of an agent in state i). Most of the theory presented below works for open models as well, see [13] for further details, but here we stick to the closed world assumption to simplify the presentation.

2.1 Running Example: a Worm Epidemic in a P2P Network

We introduce now the running example of this tutorial, which will be used to discuss the main ideas and algorithms. We consider a model of a worm epidemic in a peer-to-peer (P2P) network, which is comprised of N computers. For simplicity, we ignore new connections and disconnections, so that the number of nodes is constant (thus keeping the closed world assumption). Initially, nodes are vulnerable to the infection (susceptible S), and they can be infected by the worm in two ways, either by external infection (`ext_inf`), for instance by receiving an infected email message, or by the malicious action of an active infected node (`infect`). Infected nodes can themselves be in two states: active and inactive. An inactive infected node remains dormant and does not spread infection. In this way, the worm is harder to detect. An active node, instead, spreads the infection by sending messages to other computers in the network. The worm policy is to alternate between active and inactive states (`activate` and `deactivate`), to minimise the chances of being patched. All newly infected nodes start in the inactive state. Patching happens in all computers of the network (`patch_s`, `patch_d`, and `patch_i`), but we assume that the patching rate is higher for active nodes (`patch_i`), due to their anomalous activity in the P2P network. Patched nodes are immune, and cannot be infected by the worm. However, after some time the worm mutates, and immunity is lost (`loss`). A diagrammatic view of a network node is given in Figure 1.

To describe this system in the modelling language of this section, we need to specify the collective variables, which in this case are four: X_s, for susceptible nodes, X_d, for infected and dormant nodes, X_i for infected and active nodes, and X_p for patched nodes. Furthermore, we need 8 transitions or events whose rate and rule sets are described below:

`ext_inf`:	$R_{\texttt{ext\_inf}} = \{s \to d\}$,	$r^{(N)}_{\texttt{ext\_inf}} = k_{ext} X_s$;
`infect`:	$R_{\texttt{infect}} = \{s \to d, i \to i\}$,	$r^{(N)}_{\texttt{infect}} = \frac{k_{inf}}{N} X_s X_i$;
`activate`:	$R_{\texttt{activate}} = \{d \to i\}$,	$r^{(N)}_{\texttt{activate}} = k_{act} X_d$;
`deactivate`:	$R_{\texttt{deactivate}} = \{i \to d\}$,	$r^{(N)}_{\texttt{deactivate}} = k_{deact} X_i$;
`patch_s`:	$R_{\texttt{patch\_s}} = \{s \to p\}$,	$r^{(N)}_{\texttt{patch\_s}} = k_{low} X_s$;
`patch_d`:	$R_{\texttt{patch\_d}} = \{d \to p\}$,	$r^{(N)}_{\texttt{patch\_d}} = k_{low} X_d$;
`patch_i`:	$R_{\texttt{patch\_i}} = \{i \to p\}$,	$r^{(N)}_{\texttt{patch\_i}} = k_{high} X_i$;
`loss`:	$R_{\texttt{loss}} = \{p \to s\}$,	$r^{(N)}_{\texttt{loss}} = k_l X_p$;

In the previous list, the symbols $k\cdot$ appearing in the rate functions are model parameters that describe the rate of an action involving a single object or a single pair of objects (for `infect`). Note that the parameter for the internal infection rate is divided by N. This corresponds to the classical *density dependent* assumption for epidemic models: each infected node sends messages to other random network nodes with rate k_i, thus hitting a susceptible node with probability X_s/N. The total rate of infection is then obtained by multiplying $k_i X_s/N$ by the number of infected nodes X_i.

3 Individual Agents Properties

We turn now to discuss the class of properties we are interested to check. As announced in the introduction, we will focus on individual agents, asking questions about the behaviour of an arbitrary individual agent in the system. These properties are quite common in performance models and in network epidemics [26], whenever we are interested in checking some aspect of the system from the point of view of a single user. For instance, in client/server systems, we may be interested in quality-of-service metrics, like the expected service time [38]. In network epidemics, instead, we may be interested in properties connected with the probability of a single node being infected in a certain amount of time, or in the probability of being patched before being infected [31]. Other classes of systems can be naturally queried from the perspective of a single agent, including ecological models [46] (survival chances of an individual or foraging patterns), single enzyme kinetics in biochemistry [43] (performance of an enzyme), but also crowd models [39] or public transportation models in a smart city.

Running example. Some properties of interest of individual nodes in the worm epidemic model are listed below:

- What is the probability of a node being infected within T units of time?
- Is the probability of a single node remaining infected for T units of time smaller than p_1?
- Is the probability of a node being patched before getting infected larger than p_2?
- What is the probability of being patched within time T_1, and then remaining uninfected with probability at least p_3 for T_2 units of time?

What is shared by all those properties is the fact that they can be expressed in Continuous Stochastic Logic (CSL) [5], a well known extension to the stochastic setting of the non-deterministic Computational Tree Logic [20], which is also supported by the probabilistic model checker PRISM [37]. We will now introduce CSL, and then show how the previous properties can be expressed in this language.

3.1 Continuous Stochastic Logic

In this section we consider generic labelled stochastic processes [4, 5]. A labelled stochastic process is a random process $Z(t)$, with state space S and a labelling function $L : S \to 2^{\mathcal{P}}$, associating with each state $s \in S$, a subset of atomic propositions $L(s) \subset \mathcal{P} = \{a_1, \ldots, a_k \ldots\}$ true in that state: each atomic proposition $a_i \in \mathcal{P}$ is true in s if and only if $a_i \in L(s)$. We require that all subsets of paths considered are measurable[3].

[3] Measurability is a technical condition that guarantees that the probability of a set is defined.

This is a very general definition, and encompasses all the cases we will encounter in the rest of the paper: CTMC, time-inhomogeneous CTMC, projections of CTMC on a subset of variables. In particular, the condition on measurability will always be satisfied. From now on, we always assume we are working with labelled stochastic processes.

A path of $Z(t)$ is a sequence $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots$, such that the probability of going from s_i to s_{i+1} at time $t_\sigma[i] = \sum_{j=0}^{i} t_j$, is greater than zero. For CTMCs, this condition is equivalent to $q_{s_i,s_{i+1}}(t_\sigma[i]) > 0$, where $Q = (q_{ij})$ is the infinitesimal generator matrix. We denote by $\sigma@t$ the state of σ at time t, with $\sigma[i]$ the i-th state of σ, and with $t_\sigma[i]$ the time of the i-th jump in σ.

A time-bounded CSL formula φ is defined by the following syntax:

$$\varphi ::= true \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid P_{\bowtie p}(\psi)$$
$$\psi ::= \mathbf{X}^{[T_1,T_2]}\varphi \mid \varphi_1 \mathbf{U}^{[T_1,T_2]}\varphi_2$$

where a is an atomic proposition, $p \in [0,1]$ and $\bowtie \in \{<, >, \leq, \geq\}$. φ are known as *state formulae* and ψ are *path formulae*.

The satisfiability relation of φ with respect to a labelled stochastic process $Z(t)$ is given by the following rules:

- $s, t_0 \models a$ if and only if $a \in L(s)$;
- $s, t_0 \models \neg\varphi$ if and only if $s, t_0 \not\models \varphi$;
- $s, t_0 \models \varphi_1 \wedge \varphi_2$ if and only if $s, t_0 \models \varphi_1$ and $s, t_0 \models \varphi_2$;
- $s, t_0 \models P_{\bowtie p}(\psi)$ if and only if $\mathbb{P}\{\sigma \mid \sigma,\, t_0 \models \psi\} \bowtie p$.
- $\sigma, t_0 \models \mathbf{X}^{[T_1,T_2]}\varphi$ if and only if $t_\sigma[1] \in [T_1, T_2]$ and $\sigma[1], t_0 + t_\sigma[1] \models \varphi$.
- $\sigma, t_0 \models \varphi_1 \mathbf{U}^{[T_1,T_2]}\varphi_2$ if and only if $\exists \bar{t} \in [t_0 + T_1, t_0 + T_2]$ s.t. $\sigma@\bar{t}, \bar{t} \models \varphi_2$ and $\forall t_0 \leq t < \bar{t},\, \sigma@t, t \models \varphi_1$.

Note that the restriction to the time-bounded fragment of CSL means that we do not consider the steady state operator and we allow only time-bounded properties. This last restriction is connected with the nature of the fluid approximation (see Theorems 1 and 2 below), which hold only on finite time horizons. See [12, 13] and Section 7 for further details.

Running example. Consider the informal properties of the network epidemic model listed at the beginning of this section. We can easily rephrase them as CSL formulae (either state or path formulae), using the following atomic propositions, interpreted in the obvious way on the states of Figure 1: $a_{infected}$, $a_{patched}$. In the following, we use the convention that path formulae are denoted by ψ and state formulae are denoted by φ.

- $\psi_1 = F^{[0,T]} a_{infected}$ (a node is infected within T units of time);
- $\varphi_1 = P_{<p_1}(G^{[0,T]} a_{infected})$ (the probability of a single node remaining infected for T units of time is smaller than p_1);

– $\varphi_2 = P_{>p_2}(\neg a_{infected} \mathbf{U}^{[0,T]} a_{patched})$ (the probability of a node being patched before getting infected is larger than p_2);

– $\psi_2 = F^{[0,T_1]}(a_{patched} \wedge P_{\geq p_3}(G^{[0,T_2]} \neg a_{infected}))$ (a node is patched within time T_1, and then remains not infected with probability at least p_3 for T_2 units of time).

where $F^{[0,T]}\varphi$ is an abbreviation for $true\ \mathbf{U}^{[0,T]}\varphi$ and $G^{[0,T]}$ is an abbreviation for $\neg F^{[0,T]} \neg \varphi$.

Model checking of CSL formulae for time-homogeneous CTMC proceeds bottom-up on the parse tree of the formula [5]. Checking atomic propositions and boolean operators is trivial. The difficult part is to compute the probability of path formulae. Then this probability is compared with the threshold in the quantifier operator to establish the truth of quantified path formulae. Computing the probability of a next CSL formula $P_{\bowtie p}(\mathbf{X}^{[T_1,T_2]}\varphi)$ is usually reduced to the computation of the integral, based on the probability of making a jump in the time interval and the probability that the new state satisfies φ. For a time-homogeneous CTMC this can be solved analytically. Dealing with until CSL formula $P_{\bowtie p}(\varphi_1 \mathbf{U}^{[T_1,T_2]}\varphi_2)$ is more complex. For a time-homogeneous CTMC $Z(t)$, it can be reduced to the computation of two reachability problems, which themselves can be solved by transient analysis [5]. In particular, consider the sets of states $U = [\![\neg\varphi_1]\!]$ and $G = [\![\varphi_2]\!]$ and compute the probability of going from state $s_1 \notin U$ to a state $s_2 \notin U$ in T_1 time units, in the CTMC in which all U-states are made absorbing, $\pi^1_{s_1,s_2}(T_1)$. Furthermore, consider the modified CTMC in which all U and G states are made absorbing, and denote by $\pi^2_{s_2,s_3}(T_2 - T_1)$ the probability of going from a state $s_2 \notin U$ to a state $s_3 \in G$ in $T_2 - T_1$ units of time in such a CTMC. Then the probability of the until formula in state s can be computed as $P_s(\varphi) = \sum_{s_3 \in G, s_2 \notin U} \pi^1_{s_1,s_2}(T_1)\pi^2_{s_2,s_3}(T_2 - T_1)$. The probabilities π^1 and π^2 can be computed using standard methods for transient analysis (e.g. by uniformisation [28] or by solving the Kolmogorov equations [42]).

4 Fluid Approximation

In this section we will introduce some concepts of fluid approximation and mean field theory. The basic idea is to approximate a CTMC by an Ordinary Differential Equation (ODE), which can be interpreted in two different ways: it can be seen as an approximation of the average of the system (usually a first order approximation, see [15, 50]), or as an approximate description of system trajectories for large populations. We will focus on this second interpretation, which corresponds to a functional version of the law of large numbers: instead of having a sequence of random variables, like the sample mean, converging to a deterministic value, like the true mean, in this case we have a sequence of CTMC (which can be seen as random trajectories in $\mathbb{R}^n$) for increasing population size, which converge to a deterministic trajectory, the solution of the fluid ODE.

In order to properly speak about convergence, we need to formally define the sequence of CTMC to be considered. The collective model of Section 2 depends on the total population N, yet models of different population sizes cannot be directly compared, as it would not make sense to compute a distance between a population of the size of hundreds with a population of the size of billions: the distance will be astronomically large because of the difference in population sizes. Hence, to make the comparison meaningful, we normalise the populations, by dividing each variable for the total population N. In this way, the normalised population variables $\hat{\mathbf{X}}^{(N)} = \frac{\mathbf{X}^{(N)}}{N}$, or population densities, will always range between 0 and 1 (for the closed world models we consider here), and so the behaviour for different population sizes can be compared. In the case of a constant population, normalised variables are usually referred to as the *occupancy measure*, as they represent the fraction of agents in each state.

When we perform the normalisation, we need to impose an appropriate scaling to update vectors, initial conditions, and rate functions of the normalised models. Let $\mathcal{X}^{(N)} = (\mathbf{X}^{(N)}, \mathcal{T}^{(N)}, \mathbf{X_0}^{(N)})$ be the non-normalised model with total population N and $\hat{\mathcal{X}}^{(N)} = (\hat{\mathbf{X}}^{(N)}, \hat{\mathcal{T}}^{(N)}, \hat{\mathbf{X}}_{\mathbf{0}}^{(N)})$ the corresponding normalised model. We require that:

- initial conditions scale appropriately: $\hat{\mathbf{X}}_{\mathbf{0}}^{(N)} = \frac{\mathbf{X_0}^{(N)}}{N}$;
- for each transition $(R_\tau, r_\tau^{(N)}(\mathbf{X}))$ of the non-normalised model, define $\hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$ to be the rate function expressed in the normalised variables (obtained from $r_\tau^{(N)}$ by a change of variables). The corresponding transition in the normalised model is $(R_\tau, \hat{r}_\tau^{(N)}(\hat{\mathbf{X}}))$, with update vector equal to $\frac{1}{N}\mathbf{v}_\tau$.

We further assume, for each transition τ, that there exists a bounded and Lipschitz continuous function $f_\tau(\hat{\mathbf{X}}) : E \to \mathbb{R}^n$ on normalised variables (where E contains all domains of all $\hat{\mathcal{X}}^{(N)}$), independent of N, such that $\frac{1}{N}\hat{r}_\tau^{(N)}(\mathbf{x}) \to f_\tau(\mathbf{x})$ *uniformly* on E. In accordance with the previous subsection, we will denote the state of the CTMC of the N-th non-normalised (resp. normalised) model at time t as $\mathbf{X}^{(N)}(t)$ (resp. $\hat{\mathbf{X}}^{(N)}(t)$).

Running example. Consider again the network epidemic model, which is easily seen to satisfy all the assumptions before. The conditions for the rate functions are easily verified. They hold trivially for linear rate functions, for instance $k_{ext}X_s = Nk_{ext}\frac{X_s}{N}$, and they also hold for the non-linear rate function modelling internal infections, due to the density dependent scaling of the rate constant with respect to the total population N, i.e. $\frac{k_{inf}}{N}X_sX_i = Nk_{inf}\frac{X_s}{N}\frac{X_i}{N}$.

4.1 Deterministic limit theorem

In order to present the "classic" deterministic limit theorem, consider a sequence of normalised models $\hat{\mathcal{X}}^{(N)}$ and let $\mathbf{v}_\tau$ be the (non-normalised) update vectors. The *drift* $F^{(N)}(\hat{\mathbf{X}})$ of $\hat{\mathcal{X}}$, which is formally the mean instantaneous increment of

model variables in state $\hat{\mathbf{X}}$, is defined as

$$F^{(N)}(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \frac{1}{N} \mathbf{v}_\tau \hat{r}_\tau^{(N)}(\hat{\mathbf{X}}) \tag{2}$$

Furthermore, let $f_\tau : E \to \mathbb{R}^n$, $\tau \in \hat{\mathcal{T}}$ be the limit rate functions of transitions of $\hat{\mathcal{X}}^{(N)}$. The *limit drift* of the model $\hat{\mathcal{X}}^{(N)}$ is therefore

$$F(\hat{\mathbf{X}}) = \sum_{\tau \in \hat{\mathcal{T}}} \mathbf{v}_\tau f_\tau(\hat{\mathbf{X}}), \tag{3}$$

and $F^{(N)}(\mathbf{x}) \to F(\mathbf{x})$ uniformly as $N \longrightarrow \infty$, as easily checked. The fluid ODE is

$$\frac{d\mathbf{x}}{dt} = F(\mathbf{x}), \quad with\ \mathbf{x}(0) = \mathbf{x_0} \in S.$$

Given that F is Lipschitz in E (since all f_τ are), this ODE has a unique solution $\mathbf{x}(t)$ in E starting from $\mathbf{x_0}$. Then, one can prove the following theorem:

Theorem 1 (Deterministic approximation [34, 21]). *Let the sequence $\hat{\mathbf{X}}^{(N)}(t)$ of Markov processes and $\mathbf{x}(t)$ be defined as before, and assume that there is some point $\mathbf{x_0} \in S$ such that $\hat{\mathbf{X}}^{(N)}(0) \to \mathbf{x_0}$ in probability. Then, for any* finite *time horizon $T < \infty$, it holds that as $N \longrightarrow \infty$:*

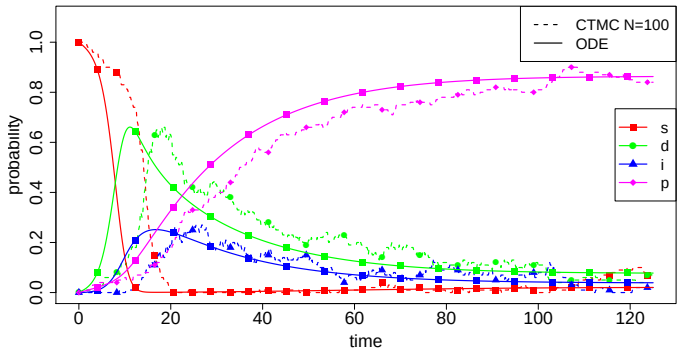
$$\mathbb{P}\left\{ \sup_{0 \le t \le T} ||\hat{\mathbf{X}}^{(N)}(t) - \mathbf{x}(t)|| > \varepsilon \right\} \to 0.$$

Running example. Focus on the internal infection `infect`. It can be easily seen that the update vector associated with the rule set $R_{\texttt{infect}}$ is $\mathbf{v}_{\texttt{infect}} = (-1, 1, 0, 0)$, given the ordering (s, d, i, p) of S. Similarly for each of the other transitions. Hence, we obtain the following set of fluid ODEs, whose solution is compared with single trajectories of the CTMC, for different populations, in Figure 2:

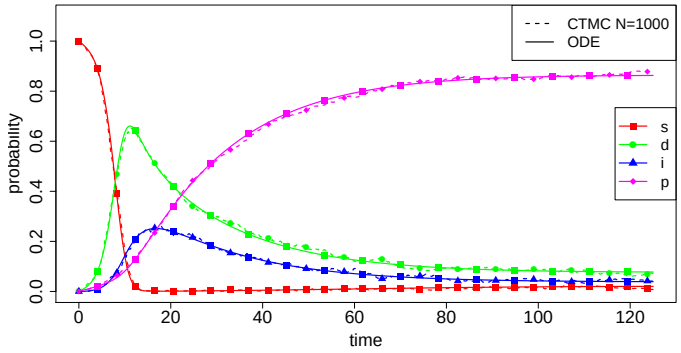
$$\begin{cases} \dfrac{dx_s(t)}{dt} = -k_{ext}x_s - k_{inf}x_s x_i - k_{low}x_s + k_{loss}x_p \\ \dfrac{dx_d(t)}{dt} = k_{ext}x_s + k_{inf}x_s x_i - k_{act}x_d - k_{low}x_d + k_{deact}x_i \\ \dfrac{dx_i(t)}{dt} = k_{act}x_d - k_{deact}x_i - k_{high}x_i \\ \dfrac{dx_p(t)}{dt} = k_{low}x_s + k_{low}x_d + k_{high}x_i - k_{loss}x_p \end{cases} \tag{4}$$

4.2 Fast simulation

We now consider what happens to a single individual in the population when the population size goes to infinity. Even if the collective behaviour tends to a



(a) $N = 100$



(b) $N = 1000$

Fig. 2. Comparison between the limit fluid ODE and a single stochastic trajectory of the network epidemic example, for total populations $N = 100$ and $N = 1000$. Parameters of the model are $k_{ext} = 0.01$, $k_{inf} = 5$, $k_{act} = 0.1$, $k_{deact} = 0.1$, $k_{low} = 0.005$, $k_{high} = 0.1$, $k_{loss} = 0.005$, while initial conditions are $\bar{X}_s(0) = 1$, $\bar{X}_d(0) = 0$, $\bar{X}_i(0) = 0$, and $\bar{X}_p(0) = 0$.

deterministic process, an individual agent will still behave randomly. However, the fluid limit theorem implies that the dynamics of a single agent, in the limit, becomes independent of other agents, and it will sense them only through the collective system state, described by the fluid limit. This asymptotic decoupling allows us to find a simple, time-inhomogenous, Markov chain for the evolution of the single agent, a result often known as *fast simulation* [22, 24].

Let us explain this point formally. We focus on a single individual $Y_h^{(N)}(t)$, which is a (Markov) process on the state space $S = \{1, \ldots, n\}$, conditional on

the global state of the population $\hat{\mathbf{X}}^{(N)}(t)$. Denote by $Q^{(N)}(\mathbf{x})$ the infinitesimal generator matrix of $Y_h^{(N)}$, described as a function of the normalised state of the population $\hat{\mathbf{X}}^{(N)} = \mathbf{x}$, i.e.

$$\mathbb{P}\{Y_h^{(N)}(t+dt) = j \mid Y_h^{(N)}(t) = i,\, \hat{\mathbf{X}}^{(N)}(t) = \mathbf{x}\} = q_{i,j}^{(N)}(\mathbf{x})dt.$$

We stress that $Q^{(N)}(\mathbf{x})$ describes the exact dynamics of $Y_h^{(N)}$, conditional on $\hat{\mathbf{X}}^{(N)}(t)$, and that this process is *not independent* of $\hat{\mathbf{X}}^{(N)}(t)$. In fact, the marginal distribution of $Y_h^{(N)}(t)$ is not a Markov process.

This means that in order to capture its evolution in a Markovian setting, one has to consider the whole Markov chain $(Y_h^{(N)}(t),\, \hat{\mathbf{X}}^{(N)}(t))$.

The rate matrix $Q^{(N)}(\mathbf{x})$ can be constructed from the rate functions of global transitions by computing the fraction of the global rate seen by an individual agent that can perform it. To be more precise, let $r_\tau^{(N)}(\mathbf{X})$ be the rate function of transition τ, and suppose $i \to j \in R_\tau$ (and each update rule in R_τ has multiplicity one). Then, transition τ will contribute to the ij-entry $q_{ij}^{(N)}(\mathbf{X})$ of the matrix $Q^{(N)}(\mathbf{X})$ with the term $\frac{1}{X_i} r_\tau^{(N)}(\mathbf{X}) = \frac{1}{\hat{X}_i} \hat{r}_\tau^{(N)}(\hat{\mathbf{X}})$, which converges to $\frac{1}{\hat{X}_i} f_\tau(\hat{\mathbf{X}})$. Additional details about this construction (taking multiplicities properly into account) can be found in [12, 13], see also the example below. From the previous discussion, it follows that the local rate matrix $Q^{(N)}(\mathbf{x})$ converges uniformly to a rate matrix $Q(\mathbf{x})$, in which all rate functions $\hat{r}_\tau^{(N)}$ are replaced by their limit f_τ. We now define two processes:

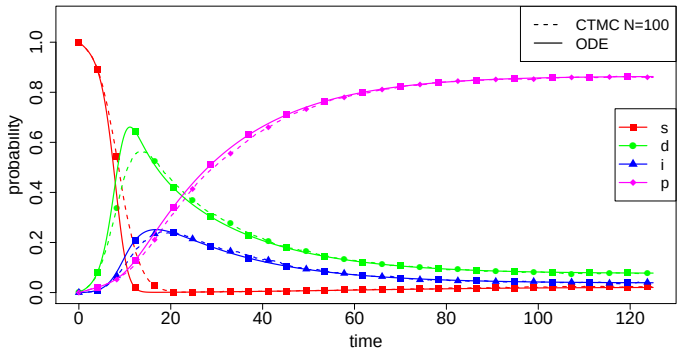
- $Z^{(N)}(t)$, which is the stochastic process describing the state of a random individual $Y_h^{(N)}(t)$ in a population of size N, marginalised with respect to the collective state $\hat{\mathbf{X}}^{(N)}(t)$.
- $z(t)$, which is a time-inhomogeneous CTMC (ICTMC), on the same state space S of $Z^{(N)}$, with time-dependent rate matrix $Q(\hat{\mathbf{x}}(t))$, where $\hat{\mathbf{x}}(t)$ is the solution of the fluid equation.

The following theorem can be proved [22]:

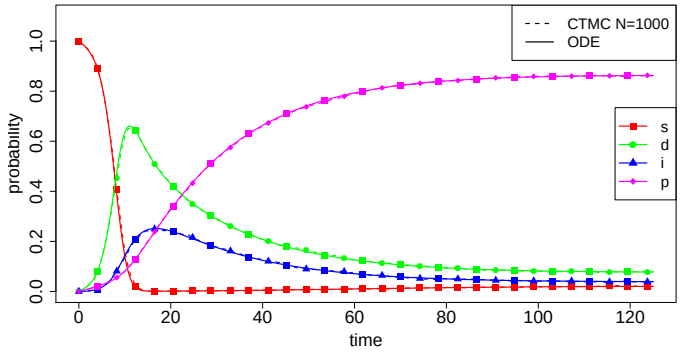
Theorem 2 (Fast simulation theorem). *For any $T < \infty$, $\mathbb{P}\{Z^{(N)}(t) \neq z(t),$ for some $t \leq T\} \to 0$, as $N \to \infty$.*

This theorem states that, in the limit of an infinite population, each agent will behave independently from all the others, sensing only the mean state of the global system, described by the fluid limit $\mathbf{x}(t)$. This *asymptotic decoupling* of the system, which can be generalised to any subset of $k \geq 1$ agents, is also known in the literature under the name of *propagation of chaos* [9].

Running example. Consider again the worm epidemic, and focus on a single node in the network. In order to construct the local rate matrix $Q^{(N)}(\mathbf{x})$, we need to consider each single transition and compute the portion of rate function seen from a single node, dividing by the population variable corresponding to



(a) $N = 100$



(b) $N = 1000$

Fig. 3. Comparison between the solution of the Kolmogorov equations for the limit model of an individual agent and an estimate of the solution for an individual agent in a finite population, of size $N = 100$ or $N = 1000$. The estimate for the finite population has been obtained by statistical means, taking the sample average of the indicator functions of each local state, for a grid of 1000 time points. Averages have been taken from 10000 samples. Parameters of the model are as in Figure 2.

the local state involved in the transition. With reference to Figure 1, we obtain the following local rate functions:

$$
\begin{aligned}
&\texttt{ext\_inf}: && s \to d, && \frac{1}{X_s} r^{(N)}_{\texttt{ext\_inf}} = k_{ext};\\
&\texttt{infect}: && s \to d, && \frac{1}{X_s} r^{(N)}_{\texttt{infect}} = \frac{1}{N} k_{inf} X_i = k_{inf} \hat{X}_i;\\
&\texttt{activate}: && d \to i, && \frac{1}{X_d} r^{(N)}_{\texttt{activate}} = k_{act};\\
&\texttt{deactivate}: && i \to d, && \frac{1}{X_i} r^{(N)}_{\texttt{deactivate}} = k_{deact};\\
&\texttt{patch\_s}: && s \to p, && \frac{1}{X_s} r^{(N)}_{\texttt{patch\_s}} = k_{low};\\
&\texttt{patch\_d}: && d \to p, && \frac{1}{X_d} r^{(N)}_{\texttt{patch\_d}} = k_{low};\\
&\texttt{patch\_i}: && i \to p, && \frac{1}{X_i} r^{(N)}_{\texttt{patch\_i}} = k_{high};\\
&\texttt{loss}: && p \to s, && \frac{1}{X_p} r^{(N)}_{\texttt{loss}} = k_{loss};
\end{aligned}
$$

Note that all the local rate functions are independent of N, and so $Q^{(N)}(\mathbf{x}) = Q(\mathbf{x})$. Ordering S as (s, d, i, p), it follows that

$$
Q(\mathbf{x}) = \begin{pmatrix}
-k_{ext} - k_{inf} x_i - k_{low} & k_{ext} + k_{inf} x_i & 0 & k_{low} \\
0 & -k_{act} - k_{low} & k_{act} & k_{low} \\
0 & k_{deact} & -k_{deact} - k_{high} & k_{high} \\
k_{loss} & 0 & 0 & -k_{loss}
\end{pmatrix}
$$

Therefore, the limit ICTMC of an individual agent depends on the solution of the fluid equation only via the fraction of infected and active nodes, $x_i(t)$. A numerical comparison of the transient probability for the limit individual agent $z(t)$ and an individual agent $Z^{(N)}(t)$ in a finite population model (for $N = 100$ and $N = 1000$) is shown in Figure 3. For $N = 1000$, it is almost impossible to distinguish between the two transient probabilities.

5 Checking CSL properties for individual agents

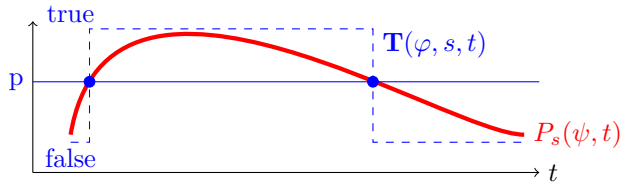
CSL model checking is computationally expensive and can become prohibitively so for large CTMC models, such as population models. The same is true of transient analysis of CTMCs but fluid approximation has provided a highly efficient means to obtain high quality approximations for population models in this case. Therefore it is natural to consider whether fluid approximation can also be exploited to find good, efficient approximations in CSL model checking. For properties that relate to a single arbitrary agent in a population model, we will demonstrate that this is indeed the case if we exploit the fast simulation property established in Theorem 2.

In the fluid approximation of a CSL property φ for an arbitrary individual agent $Z^{(N)}(t)$ in a population of size N we exploit Theorem 2 and replace $Z^{(N)}(t)$ by its fluid limit $z(t)$, checking φ on $z(t)$. The essential advantage in doing this is

that, in order to properly compute the satisfaction of CSL formulae for $Z^{(N)}(t)$ we need to take into account the whole population model $\hat{\mathbf{X}}^{(N)}(t)$. This results in a huge state space that is out of reach of CSL model checkers. Simulation-based methods, like statistical model checking [29], may be exploited for this purpose for moderately sized populations (this is what we do to compare our approximate method with the results for the proper stochastic model), but simulation becomes extremely costly when the population increases.

As we will show in the rest of the chapter, checking properties on $z(t)$ is much more efficient, and the error seems to remain small. In addition to experimental validation, Theorem 2 provides us with the means of formally showing that the result of checking CSL properties on $z(t)$ and on $Z^{(N)}(t)$ will be the same, provided N is sufficiently large.

In replacing $Z^{(N)}(t)$ with $z(t)$, however, we have to face the fact that $z(t)$ is a time-inhomogeneous CTMC. It turns out that working with ICTMC is much more complex, because of the dependency of the satisfaction of a formula on time. In fact, if we inspect the definition of CSL semantics in Section 3.1, we can observe that the satisfaction relation depends on a state of the model and on the time at which the formula is evaluated. This particularly affects the computation of the probabilities of path formulae. In the case of time-homogeneous CTMC, time dependency is not an issue, as rates are constant, hence starting the system at time $t_0 > 0$ is the same as starting it at time 0. But when rates depend on time, this is no longer the case. What can happen is schematically depicted in the figure below.



In this figure, we show a hypothetical scenario in which the probability of a path formula ψ is plotted against the time t at which the formula is evaluated. When we compare this function with the threshold p in the probability operator of $\varphi = P_{\bowtie p}(\psi)$, it can happen that this function is above p for some time instants and below it for some other time instants. It follows that the truth $\mathbf{T}(\varphi, s, t)$ of a CSL temporal formula in a state $s \in S$, can itself depend on the time at which the formula is evaluated. This makes CSL model checking for time-inhomogeneous CTMC a much more delicate business: we need to compute not a single probability for a path formula, but its probability as a function of time, and we further need to take into account the time-dependent truth of CSL formulae when checking nested formulae.

In the rest of the chapter, we will first discuss how to compute next state and reachability probabilities (the two main building blocks of CSL algorithms) when the next-state set or the goal/unsafe sets are independent of time (Sections 5.1 and 5.2), facing also the problem of computing the dependency of such

probabilities on the initial time (Section 6). Then, we will move to nested CSL formulae, and give an intuition on how to compute path probabilities in the case of nested temporal operators (Section 7). Finally, in Section 8 we will briefly discuss theoretical properties, like decidability of the algorithms and convergence of CSL truth as population increases.

5.1 Next State Probabilities

In this section, we will show how to compute the probability that the next state in which an agent jumps belongs to a given set of states $S_0 \subseteq S$, constraining the jump to happen between time $[t_0 + T_1, t_0 + T_2]$, where t_0 is the current time. This is clearly at the basis of the computation of the probability of next path formulae.
Let $Z(t)$ be a CTMC with state space S and infinitesimal generator matrix $Q(t)$. We indicate with $P_{next}(Z, t_0, T_1, T_2, S_0)[s]$ the probability of the set of trajectories of Z jumping into a state in S_0, starting at time t_0 in state s, within time $[t_0 + T_1, t_0 + T_2]$. Hence, $P_{next}(Z, t_0, T_1, T_2, S_0)$ is a vector of probabilities, one for each state $s \in S$.

For any fixed t_0, the probability $P_{next}(Z, t_0, T_1, T_2, S_0)[s]$ is given by the following integral [51, 30]

$$P_{next}(Z, t_0, T_1, T_2, S_0)[s] = \int_{t_0+T_1}^{t_0+T_2} q_{s,S_0}(t) \cdot e^{-\Lambda(t_0,t)[s]} dt, \tag{5}$$

where $\Lambda(t_0, t)[s] = \int_{t_0}^{t} -q_{s,s}(\tau) d\tau$ is the cumulative exit rate of state s from time t_0 to time t, and $q_{s,S_0}(t) = \sum_{s' \in S_0, s' \neq s} q_{s,s'}(t)$ is the rate of jumping from s to a state $s' \in S_0$, $s' \neq s$, at time t.

Equation 5 can be explained as follows: first of all, remember that for an exponential distribution, the probability density of the first jump happening at time t, given that we are in state s at time t_0, is $\Lambda(t_0, t)[s] e^{-\Lambda(t_0,t)[s]}$. For a time homogeneous CTMC, it holds that $\Lambda(t_0, t)[s] = \lambda_s(t - t_0)$, where λ_s is the exit rate from state s, hence the density takes the more common form. Furthermore, if we jump at time t, then the probability of landing in a state of S_0 is $q_{s,S_0}(t)/\Lambda(t_0, t)[s]$. Multiplying this for the probability density above, we obtain the probability density of jumping in a state of S_0 at time t, which is the argument of the integral (5). Then we only need to integrate it from time $t_0 + T_1$ to time $t_0 + T_2$ to compute the desired quantity.

In order to practically compute $P_{next}(Z, t_0, T_1, T_2, S_0)[s]$ for a given t_0, we can either numerically compute the integral, or transform it into a differential equation, and integrate the so-obtained ODE with standard numerical methods. This second method is preferrable, as it can be extended to compute the next probability as a function of the initial time, see Section 6. More specifically, we can introduce two variables, P (giving the desired probability) and L (giving the cumulative rate Λ), initialise $P(t_0 + T_1) = 0$ and $L(t_0 + T_1) = \Lambda(t_0, t_0 + T_1)$, and

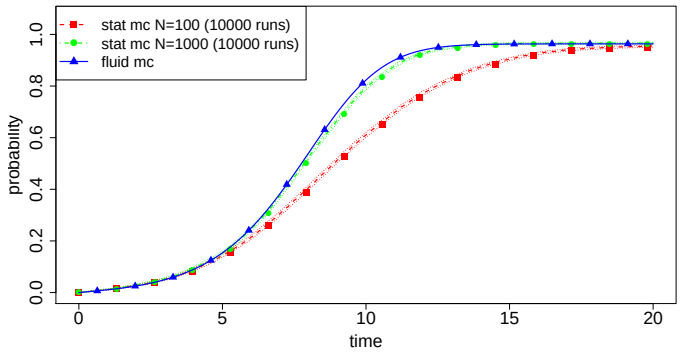


Fig. 4. Path probability of $\mathbf{X}^{[0,T]}a_{infected}$, as a function of T, starting in state s at time 0. The fluid approximation (continuous line) is compared with the statistical estimate (computed using statistical model checking out of an ensemble of 10000 runs) for population levels of $N = 100$ and $N = 1000$. Binomial confidence intervals are reported in the plot (they are quite narrow) and parameters of the model are as in Figure 2.

then integrate the following two ODEs from time $t_0 + T_1$ to time $t_0 + T_2$:

$$\begin{cases} \dfrac{d}{dt}P(t) = q_{s,S_0}(t) \cdot e^{-L(t)} \\ \dfrac{d}{dt}L(t) = -q_{s,s}(t) \end{cases} \tag{6}$$

Running example. We consider the path formula $\psi = \mathbf{X}^{[T_1,T_2]}a_{infected}$, which expresses the fact that a node of the network will change state between time $t_0 + T_1$ and $t_0 + T_2$, and it will become (or remain) infected. In Figure 4, we show the probability of the path formula for the fluid limit model of a single node in the network, initially in the susceptible state s, for $t_0 = 0$ and $T_1 = 0$, as a function of $T_2 = T$.

5.2 Reachability Probabilities

We now turn to the computation of reachability probabilities of an individual agent. Essentially, we want to compute the probability of the set of traces reaching some goal state $G \subseteq S$ within T units of time, given that we are in state $s \in S$ at time t_0, and avoiding unsafe states $U \subseteq S$, which will be denoted by $P_{reach}(Z, t_0, T, G, U)[s]$, where $Z(t)$ is a ICTMC on S, with rate matrix $Q(t)$ and initial state $Z(0) = Z_0 \in S$.

The computation of reachability probabilities is the key operation needed to compute probabilities of until formulae. In fact, the probability of the path

formula $\varphi_1 \mathbf{U}^{[0,T]} \varphi_2$ is the probability of reaching a goal set $G = \{s \mid s \models \varphi_2\}$, avoiding unsafe states $U = \{s \mid s \models \neg\varphi_1\}$. Here we are assuming the satisfaction of φ_1 and φ_2 does not depend on time. We will solve this reachability problem in a standard way, by reducing it to the computation of transient probabilities in a modified ICTMC [5], similarly to [18].

Let $\Pi(t_1, t_2)$ be the probability matrix of $Z(t)$, in which entry $\pi_{s_1,s_2}(t_1, t_2)$ gives the probability of being in state s_2 at time t_2, given that Z was in state s_1 at time t_1. The *Kolmogorov forward and backward equations* [42] describe the time evolution of $\Pi(t_1, t_2)$ as a function of t_2 and t_1, respectively. More precisely, the forward equation is

$$\frac{\partial \Pi(t_1, t_2)}{\partial t_2} = \Pi(t_1, t_2) Q(t_2),$$

while the backward equation is

$$\frac{\partial \Pi(t_1, t_2)}{\partial t_1} = -Q(t_1) \Pi(t_1, t_2).$$

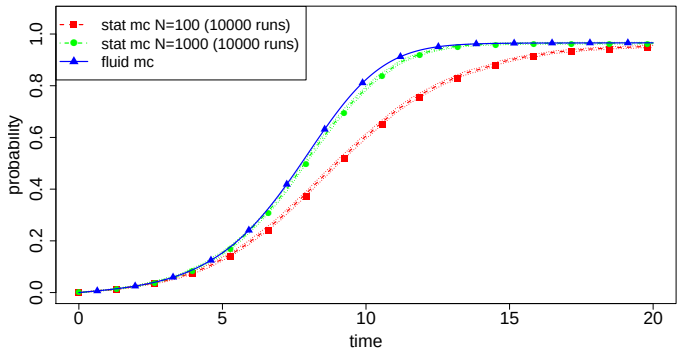
The probability $P_{reach}(Z, t_0, T, G, U)$, for a given initial time t_0, can be solved integrating the forward Kolmogorov equation (with initial value given by the identity matrix) in the ICTMC $Z'(t)$ in which all unsafe states and goal states are made absorbing [5]. The infinitesimal generator matrix $Q'(t)$ of $Z'(t)$ is obtained from $Q(t)$ by setting $q'_{s_1,s_2}(t) = 0$, for each $s_1 \in G \cup U$.
In particular, $P_{reach}(Z, t_0, T, G, U) = \Pi'(t_0, t_0 + T)\mathbf{e}_G$, where $\mathbf{e}_G$ is an $n \times 1$ vector equal to 1 if $s \in G$ and 0 otherwise, and Π' is the probability matrix of the modified ICTMC Z'.[4] We emphasise that, in order for the initial value problem defined by the Kolmogorov forward equation to be well posed, the infinitesimal generator matrix $Q(t)$ has to be sufficiently regular (e.g. bounded and integrable).

Running example. Consider again the running example, and the path formula $\psi_1 = F^{[0,T]} a_{infected}$, expressing the fact that a node will be infected within T units of time, starting from time t_0. The path probability of ψ_1 can be recast into the computation of a reachability probability, with goal set $G = \{d, i\}$ and unsafe set $U = \emptyset$. Applying the method discussed above, we just need to compute the transient probability for the ICTMC with rate matrix

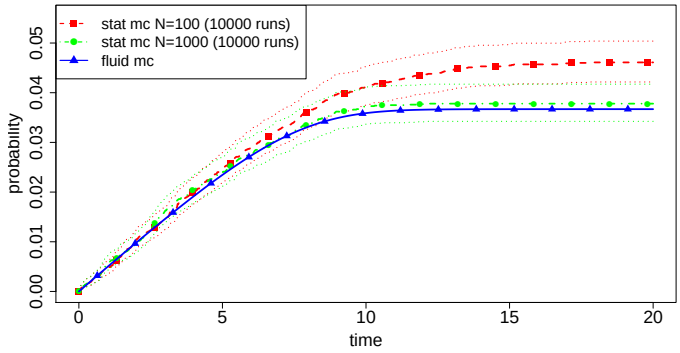
$$Q'(\mathbf{x}) = \begin{pmatrix} -k_{ext} - k_{inf} x_i - k_{low} & k_{ext} + k_{inf} x_i & 0 & k_{low} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k_{loss} & 0 & 0 & -k_{loss} \end{pmatrix}$$

in which states d and i are made absorbing. The result, starting from $t_0 = 0$ and as a function of T, is shown in Figure 5(a).

[4] Clearly, alternative ways of computing the transient probability, like uniformization for ICTMC [3], could also be used. However, we stick to the ODE formulation in order to deal with dependency on the initial time t_0.



(a) $F^{[0,T]}a_{infected}$



(b) $\neg a_{infected}\mathbf{U}^{[0,T]}a_{patched}$

Fig. 5. Path probability of the formulas $F^{[0,T]}a_{infected}$ and $\neg a_{infected}\mathbf{U}^{[0,T]}a_{patched}$, as a function of T, starting in state s at time 0. The fluid approximation (continuous line) is compared with the statistical estimate (computed using statistical model checking out of an ensemble of 10000 runs) for population levels of $N = 100$ and $N = 1000$. Binomial confidence intervals are reported in the plot, and parameters of the model are as in Figure 2.

In Figure 5(b), instead, we show the result of computing the probability of the path formula $\psi_3 = a_{not\ infected}\mathbf{U}^{[0,T]}a_{patched}$, for $t_0 = 0$, as a function of T. Here, we just need to compute the reachability probability for the goal set $G = \{p\}$ and unsafe set $U = S \setminus \{s, p\} = \{d, i\}$.

6 Time Dependent Path Probabilities

In this section we will present a method to compute next state and reachability probabilities as a function of the time t_0 at which the property is evaluated. As we have already discussed, this is the crucial step to check nested CSL formulae. In fact, the satisfaction of a CSL formula depends on the time at which the formula is evaluated. This is particularly the case for a quantified path formula like $\varphi = P_{\leq p}(\psi)$, where ψ can be an until or a next formula. In this case, the probability of the path formula ψ from state $s \in S$, $P(t_0, \psi)[s]$, depends on the initial time t_0 at which we start evaluating such a formula. Hence, $P(t_0, \psi)[s]$ is a function of t_0, and when we evaluate the inequality $P(t_0, \psi)[s] \leq p$, needed to establish the truth of φ, we may find that the inequality holds for some t_0 and is falsified for other t_0 (see again the figure in Section 5).

Hence, we need a way to compute the probability of a path formula as a function of time. The starting point for this will be the formulation of next state and reachability probabilities as solutions of a differential equation. In fact, we will derive other differential equations whose solution will return the probability of path formulae as a function of the initial time.

Before presenting the derivation of the ODE more formally, let us comment on the choice of using ODE-based methods to compute transient probabilities, rather than more standard unifomisation-based algorithms. The first and more fundamental reason is precisely connected with the dependency of path probabilities on the initial time: uniformisation algorithms do not generalise easily to such scenarios. Moreover, the size of the state space of a single individual is usually very small, even when the collective system has a huge state space. Hence, numerical solvers for ODEs will work fine and will be efficient. Additionally, the fluid limit for an individual agent depends on the solution of the fluid ODE, so in any case we need to resort to ODE solvers. Coupling all the ODEs together allows us to exploit adaptive solvers [17] in order to control and reduce the global error.

Time-dependent next probabilities We will start by showing how to compute the next-state probability $P_{next}(Z, t_0, T_1, T_2, S_0)[s]$ as a function of t_0: $\bar{P}_s(t_0) = P_{next}(Z, t_0, T_1, T_2, S_0)[s]$. Computing the integral (5) for any t_0 is obviously not feasible. However, we can exploit the differential formulation of the problem, and define a set of ODEs with the initial time t_0 as an independent variable. First, we can compute the derivative of $\bar{P}_s(t_0)$ with respect to t_0 and obtain

$$\begin{aligned}\frac{d}{dt_0}\bar{P}_s(t_0) &= q_{s,S_0}(t_0+T_2)\cdot e^{-\Lambda(t_0,t_0+T_2)} - q_{s,S_0}(t_0+T_1)\cdot e^{-\Lambda(t_0,t_0+T_1)} \\ &\quad + \int_{t_0+T_1}^{t_0+T_2} \frac{\partial}{\partial t_0} q_{s,S_0}(t)\cdot e^{-\Lambda(t_0,t)}dt \\ &= q_{s,S_0}(t_0+T_2)\cdot e^{-\Lambda(t_0,t_0+T_2)} - q_{s,S_0}(t_0+T_1)\cdot e^{-\Lambda(t_0,t_0+T_1)} \\ &\quad - q_{s,s}(t_0)\bar{P}_s(t_0)\end{aligned}$$

Consequently, we can compute the next-state probability as a function of t_0 by solving the following set of ODEs:

$$\begin{cases} \dfrac{d}{dt}\bar{P}_s(t) = q_{s,S_0}(t+T_2)\cdot e^{-L_2(t)} - q_{s,S_0}(t+T_1)\cdot e^{-L_1(t)} - q_{s,s}(t)\bar{P}_s(t) \\ \dfrac{d}{dt}L_1(t) = q_{s,s}(t) - q_{s,s}(t+T_1) \\ \dfrac{d}{dt}L_2(t) = q_{s,s}(t) - q_{s,s}(t+T_2) \end{cases} \tag{7}$$

where $L_1(t) = \Lambda(t, t+T_1)$ and $L_2(t) = \Lambda(t, t+T_2)$.
Initial conditions are $P_s(t_0) = P_{next}(Z, t_0, T_1, T_2, S_0)[s]$, $L_1(t_0) = \Lambda(t_0, t_0+T_1)$, and $L_2(t_0) = \Lambda(t_0, t_0+T_2)$, and are computed solving the equations (6).

Running example. We consider again the next path formula $\psi = \mathbf{X}^{[0,T]}a_{infected}$, fix $T = 7.5$, and compute its path probability $\bar{P}_s(t_0)$, from the susceptible state s, as a function of t_0. In order to do this, we need to first solve the ODEs (6) for $t_0 = 0$ and $T = 7.5$, in order to obtain the initial conditions of the ODEs (7). Then, we need to solve the following ODE system:

$$\begin{cases} \dfrac{d}{dt}\bar{P}_s(t) = (k_{ext} + k_{inf}x_i(t+T))e^{-L_2(t)} - (k_{ext} + k_{inf}x_i(t)) + \dots \\ \qquad\qquad + (k_{ext} + k_{inf}x_i(t) + k_{low})\bar{P}_s(t) \\ \dfrac{d}{dt}L_2(t) = k_{inf}(x_i(t+T) - x_i(t)) \\ L_1(t) = 0 \end{cases}$$

Its solution is shown in Figure 6, together with the truth of the formula $P_{\leq 0.8}(\mathbf{X}^{[0,T_2]}a_{infected})$ in state s, for $t_0 \in [0, 10]$. As we can see, the formula is initially true and then, at time $t = 2.26$, it changes truth status and becomes false.

Time-dependent reachability We now turn to the problem of computing $P(t) = P_{reach}(Z, t, T, G, U)$ as a function of $t \in [t_0, t_1]$. Recall that in Section 5.2 we reduced the computation of $P(t)$, for a fixed initial time t, to the solution of the Kolmogorov forward equation of the modified ICTMC, in which goal and unsafe sets are made absorbing. Stated otherwise, we used the forward equation to compute $\Pi(t, t')$, from $t' = t$ to $t' = t + T$. To compute the reachability probability as a function of the initial time $t \in [t_0, t_1]$, for T fixed, we need to compute $\Pi(t, t+T)$ for $t \in [t_0, t_1]$. We can do this using the chain rule and combining the forward and the backward Kolmogorov equation to obtain the following ODE for $\Pi(t, t+T)$:

$$\begin{aligned} \frac{d\Pi(t,t+T)}{dt} &= \frac{\partial\Pi(t,t+T)}{\partial t} + \frac{\partial\Pi(t,t+T)}{\partial(t+T)}\frac{d(t+T)}{dt} \\ &= -Q(t)\Pi(t,t+T) + \Pi(t,t+T)Q(t+T). \end{aligned} \tag{8}$$

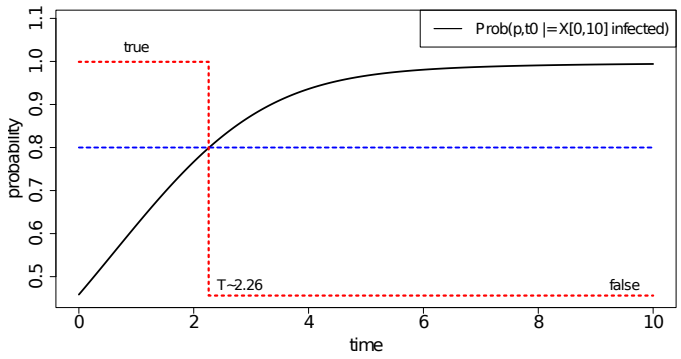


Fig. 6. Fluid estimate of the path probability of the formula $\mathbf{X}^{[0,T_2]}a_{infected}$, for $T_2 = 7.5$, as a function of the initial time t at which the formula is evaluated. We assume we start in state s at time t. The red dotted line represents the time-dependent truth value of the formula $P_{\leq 0.8}(\mathbf{X}^{[0,7.5]}a_{infected})$ in state s. Parameters of the model are as in Figure 2.

Here the initial condition is $\Pi(t_0, t_0+T)$, and it can be computed by solving the Kolmogorov forward equation. Using a numerical ODE solver, we can integrate this equation and obtain $\Pi(t, t+T)$ for $t \in [t_0, t_1]$. This gives the basic algorithm to compute probabilities $P_s(\psi, t)$ of until path formulae like $\psi = \varphi_1 \mathbf{U}^{[0,T]} \varphi_2$ from a state s: we just need to compute the reachability probability $\pi_{s,s'}(t, t+T)$ and add it over states $s' \in G$.[5] Once the until probability is computed, we can check if state s satisfies $P_{\bowtie p}(\psi)$ at time t by comparing the value $P_s(t)$ with the threshold p. Doing this for all times $t \in [t_0, t_1]$ requires us to find all zeros of the function $P_s(t) - p$. This can be done during the integration of the ODEs, using event detection routines, provided the number of zeros is finite and the function $P_s(t) - p$ always changes sign around a zero. This does not hold in all cases, and further restrictions on the rate functions and the thresholds p have to be made, see [12, 13] and Section 8 below for more details.

Running example. Consider the formula $\psi_4 = G^{[0,T_2]} \neg a_{infected}$, fix T_2 to 10, and evaluate it as a function of the initial time. In order to apply the reachability algorithm above, we need to turn the always operator into an until. This is done in the standard way, as $G^{[0,T_2]} \neg a_{infected} \equiv \neg F^{[0,T_2]} a_{infected} \equiv \neg(true \mathbf{U}^{[0,T_2]} a_{infected})$. Hence, we need to compute the reachability probability for the goal set $G = \{i, d\}$ and the unsafe set $U = \emptyset$, and then compute 1 minus this probability. The result for the patched state p is shown in Figure 7, for the initial time varying between 0 and 150.
If we now consider the state formula $P_{\geq 0.97}(\psi_4)$ and focus again on the patched

[5] More general until formulae $\varphi_1 \mathbf{U}^{[T_1,T_2]} \varphi_2$, for $T_1 > 0$, can be dealt by a minor modification of the approach, see [12, 13] for further details.

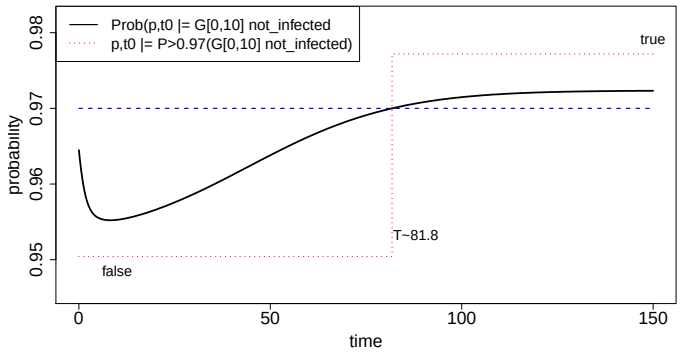


Fig. 7. Fluid estimate of the path probability of the formula $G^{[0,T_2]}\neg a_{infected}$, for $T_2 = 10$, as a function of the initial time t at which the formula is evaluated, starting from the patched state p at time t (continuous black line). Then the path probability is compared with the threshold $p = 0.97$, and the truth value of the CSL state formula $P_{\geq 0.97}(G^{[0,10]}\neg a_{infected})$ is computed, as a function of the time t at which it is evaluated. The time-dependent truth is depicted in the red dotted line. Parameters of the model are as in Figure 2.

state p, then we see in Figure 7 that the formula is false from time 0 to time $T \approx 81.8$ and then becomes true.

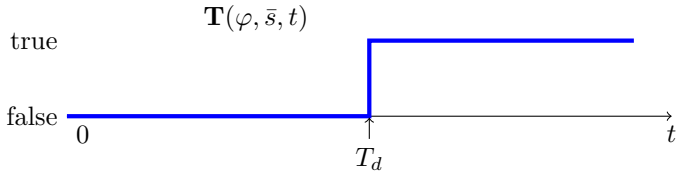
7 Nested CSL Formulae

Computing the truth of nested CSL formulae for time-homogeneous CTMC is the same as for non-nested formulae. For an until or next temporal operator, we first solve the model checking problem for its subformulae, hence establish if a state satisfies or falsifies them, and then we use this information in the standard algorithms (e.g. the reachability algorithm based on transient analysis for the until case). Unfortunately, this simple recipe does not work for time-inhomogeneous CTMC. The problem is that a state satisfies a subformula containing a temporal operator depending on the time at which the formula is evaluated. This fact introduces an extra dimension of complexity into the algorithms.

To give a flavour of the problems involved, let us discuss the nested path formula $\psi = F^{[0,T]}(P_{\geq p}(\varphi_1 \mathbf{U}^{[0,T_1]}\varphi_2))$, where φ_1 and φ_2 are boolean combinations of atomic propositions, so that their truth in a state does not depend on the time at which we evaluate them. Clearly, using the procedure put forward in the previous section, we can compute, for each state $s \in S$, the probability $P_s(t)$ of the path formula $\varphi_1 \mathbf{U}^{[0,T_a]}\varphi_2$, as a function of the time at which we evaluate it. Therefore, we can compute the time-dependent truth function $\mathbf{T}(\varphi, s, t)$ of the state formula $\varphi = P_{\geq p}(\varphi_1 \mathbf{U}^{[0,T_a]}\varphi_2)$ by finding the zeros of $P_s(t) - p$, for

each state $s \in S$. Specifically, we obtain that $\mathbf{T}(\varphi, s, t) = \mathit{true}$ if and only if $s, t \models \varphi$, and false otherwise. In general, this function will keep jumping between 0 (false) and 1 (true), and we can represent it by identifying and storing in a list all time instants T_d at which a change in the truth of $s, t \models \varphi$ happens.

Fix now a state $\bar{s}$, and assume that at time T_d the function $\mathbf{T}(\varphi, \bar{s}, t)$ has a discontinuity, and that φ was false in $\bar{s}$ before time T_d and it is true afterwards. Now, focus the attention on the path formula $\psi = F^{[0,T]}(\varphi)$. To compute its path probability, we need to compute the probability of reaching a state satisfying φ within T time units. This is done by making φ-states absorbing, and computing the transient probability in the so-modified Markov chain. If $T_d < T$, then $\bar{s}$ is not a goal state from time 0 to time T_d, and becomes a goal state at time T_d, as shown in the figure below.



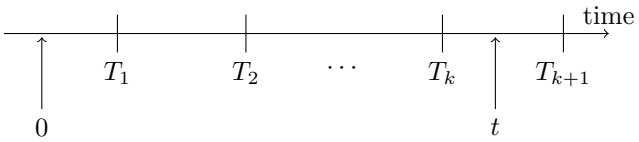
The first consequence of this fact is that the modified Markov chain in which goal states are made absorbing has a structure that changes in time, according to the truth of formula φ. In particular, $\bar{s}$ is not absorbing from time 0 to T_d, and becomes absorbing at time T_d.

Now fix a non-goal state s' (remaining non-goal for the whole time interval $[0, T]$) and focus on the reachability probability starting from this state. In particular, there can be a non-null probability to go from s' to $\bar{s}$ in T_d units of time, starting at time 0. This means that the probability of the set of trajectories starting at s' at time 0 and being in $\bar{s}$ at time T_d, without passing from a goal state, has non-null probability. Pick one such trajectory, which clearly does not contribute to the reachability probability from s'. At time T_d, however, the structure of the modified CTMC changes, and this trajectory suddenly satisfies the reachability condition. In particular, this holds at time T_d for all the trajectories that are in $\bar{s}$. Hence, at time T_d the probability $\pi_{s',\bar{s}}(0, T_d)$ has to be added to the reachability probability $P_{s'}(T_d^-)$, as computed before $\bar{s}$ becomes a goal state.

Stated otherwise, a change in the truth status of the formula φ not only forces us to *change the topology* of the modified CTMC, by altering the set of absorbing states, but it may also *induce a discontinuity* in the path probability.

Computing reachability probabilities for time-varying sets. We will quickly sketch now an algorithmic procedure to compute reachability probabilities when the goal and unsafe sets vary with time. This will provide the key procedure to compute path probabilities for nested until formulae of the form $\varphi_1 \mathbf{U}^{[0,T]} \varphi_2$, for general CSL formulae φ_1 and φ_2. For more general until formulae $\varphi_1 \mathbf{U}^{[T_a,T_b]} \varphi_2$, and for next path formulae, we refer the reader to [12, 13].

We first discuss the case in which the unsafe set U is always empty, corresponding to eventually path formulae. To compute the reachability probability, we need to take the double nature of states into account: a state s can be either goal or non-goal, and its status can vary with time. To better describe this scenario, we will double the state space, creating for each state $s \in S$ a shadow copy $\bar{s}$, which represents the goal version of s. Shadow states $\bar{s}$ are always absorbing. Each transition in the CTMC entering an s state, instead, is directed towards s if and only if s is non-goal. Otherwise, it is rerouted towards $\bar{s}$.[6] This routing has to be changed whenever we hit a discontinuity in the time-dependent truth function. The situation is depicted below.



Let $T_1, \ldots, T_k, T_{k+1}, \ldots$ be the times in which any state of S changes status (from goal to non-goal, or vice versa). Note that these time instants are fixed, and we can assume them to be known. To compute the reachability probability within time $[0, T]$, we start in $T_0 = 0$ by constructing the modified CTMC according to the goal set $G(0)$ at time 0. In between T_0 and T_1, the structure of the modified CTMC does not change, hence we can integrate the forward Kolmogorov equations, until the time t hits the first discontinuity time T_1. At this time, we need to perform some operations, according to whether the state s changes status to become a goal state or a non-goal state.

Goal to non-goal: in this case, we only need to reroute the transition matrix of the CTMC: all transitions entering $\bar{s}$, the shadow version of s, must now point to s. This is obtained by modifying the Q-matrix accordingly, deleting entries in the column $\bar{s}$ and adding entries in the column s.

Non-goal to goal: In this case, we need to reroute transitions which enter s to now point at $\bar{s}$ entering s, and also add the probability $\pi_{s',s}(0,t)$ to $\pi_{s',\bar{s}}(0,t)$, afterwards set $\pi_{s',s}(0,t)$ to zero.

Once these bookkeeping operations have been performed, and the new probability matrix $\Pi(0, T_1^+)$ has been computed if needed, we can restart the integration of the forward Kolmogorov equations, with initial conditions given precisely by $\Pi(0, T_1^+)$. We can then iterate this procedure, until the final time T is reached.

Running example. We consider the running example, and compute the probability of the nested path formula $\psi = F^{[0,T]}(a_{patched} \wedge P_{\geq p}(G^{[0,T_a]} \neg a_{infected}))$, for $T_a = 10$. The time dependent truth of the inner temporal formula $\varphi = P_{\geq p}(G^{[0,T_a]} \neg a_{infected})$ has already been computed in the previous section (see Figure 7). Furthermore, the formula $a_{patched} \wedge P_{\geq p}(G^{[0,T_1]} \neg a_{infected})$ is false

[6] Alternatively, we can add a new goal state s^*, as done in [32], and redirect all transitions entering any goal state to s^*.

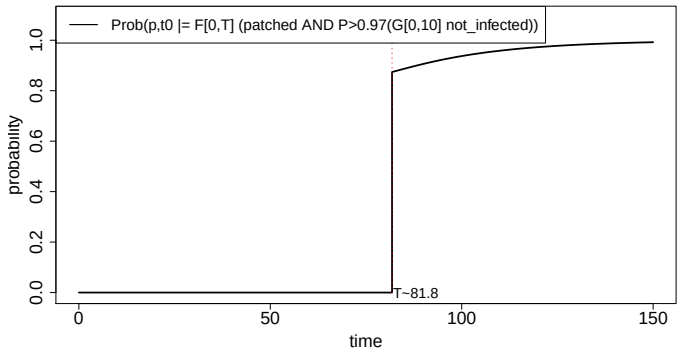


Fig. 8. Fluid estimate of the path probability of the formula $F^{[0,T]}(a_{patched} \wedge P_{\geq 0.97}(G^{[0,10]} \neg a_{infected}))$, as a function of T, starting from state p at time 0 (continuous black line). We can see that the probability is discontinuous, and there is a jump at time $T = 81.8$, corresponding to the time at which the truth value of $P_{\geq 0.97}(G^{[0,10]} \neg a_{infected})$ changes for state p, see Figure 7. Parameters of the model are as in Figure 2.

for any state different from p, and equal to $\mathbf{T}(\varphi, p, t)$ for the patched state p. Hence, the state p will be non-goal until time $T_d = 81.8$, and then it will become a goal state.
Thus, when computing the probability of the path formula ψ, we have that no state is goal until time T_d, and after T_d p will be the only goal state. If we look at the path probability of ψ as a function of T (Figure 8), we observe that this probability is zero for $T < T_d$, and then suddenly jumps at time T_d to $\pi_{x,p}(0, T_d)$, for $x \in S$, and keeps on increasing afterwards.

The algorithm to compute the path probability for a general reachability problem, with both unsafe and goal states, is similar to the one sketched above. In general, for an unsafe state s we disable all outgoing transitions, making it absorbing. The only difference resides in the bookkeeping operations that need to be performed when a state s changes its unsafe status.

Unsafe to safe: in this case, we just need to re-enable all the outgoing transitions from s.

Safe to unsafe: In this case, we disable all outgoing transitions from s, but we also discard the probability $\pi_{s',s}(0, t)$, setting it to zero. This is because all trajectories started from s' at time 0 and being in s when it becomes unsafe become trajectories that can no longer reach a goal state avoiding unsafe ones, because they suddenly find themselves in an unsafe state.

A minor caveat when we have both unsafe and goal sets is to decide how a state that is both goal and unsafe behaves. In this case, by the definition of the until semantics, its goal nature will prevail.

Reachability probabilities as a function of time Up to now, we sketched an algorithm to compute the reachability probability starting from time zero up to time T, call it $\Upsilon(0,T)$. In order to extend the method to compute $\Upsilon(t,t+T)$, as a function of the initial time t, we will follow a similar approach to that of Section 6, finding an expression for $\Upsilon(t,t+T)$ and applying a generalised version of the forward and backward Kolmogorov equations to it, in order to obtain an ODE for Υ.

In this tutorial, we will just present this expression for Υ, which is obtained by combining Chapman Kolmogorov equations [42] with suitable matrices encoding the bookkeeping operations. Further details on the algorithm can be found in [12, 13].

The first step is the definition of a matrix ζ encoding the bookkeeping operations. Let $G(t)$ and $U(t)$ be the time-varying goal and unsafe sets, and let $T_1, \ldots, \mathbf{true}_k, \ldots$ be the time instants at which one state changes status. Define the set of safe states $W(t) = S \setminus (G(t) \cup U(t))$. We define the following matrices for each discontinuity time T_i:

- $\zeta_W(T_i)$ is the $n \times n$ matrix, $|S| = n$, equal to 1 only on the diagonal elements corresponding to states s_j belonging to both $W(T_i^-)$ and $W(T_i^+)$ (i.e. states that are safe and not goals both before and after T_i), and equal to 0 elsewhere;
- $\zeta_G(T_i)$ is the $n \times n$ matrix equal to 1 in the diagonal elements corresponding to states s_j belonging to $W(T_i^-) \cap G(T_i^+)$ (safe and non-goal states becoming goal), and zero elsewhere;
- $\zeta(T_i)$ is the $2n \times 2n$ matrix defined by:

$$\zeta(T_i) = \begin{pmatrix} \zeta_W(T_i) & \zeta_G(T_i) \\ 0 & I \end{pmatrix}.$$

Now, assume in $[t_1, t_2]$ no discontinuity occurs, so that the Q-matrix of the modified CTMC does not change structure in $[t_1, t_2]$, and let $\tilde{\Pi}(t_1, t_2)$ be the probability matrix computed by solving the forward Kolmogorov equations, with $\tilde{\Pi}(t_1, t_1) = I$. Then, recalling that the Chapman Kolmogorov equations state that $\tilde{\Pi}(t_1, t_3) = \tilde{\Pi}(t_1, t_2)\tilde{\Pi}(t_2, t_3)$, we can compute Υ as follows [12, 13]:

$$\Upsilon(t, t+T) = \tilde{\Pi}(t, T_1)\zeta(T_1)\tilde{\Pi}(T_1, T_2)\zeta(T_2)\cdots\zeta(T_{k_I})\tilde{\Pi}(T_{k_I}, t+T), \qquad (9)$$

where $T_1, \ldots, T_{k_I}$ are all the discontinuity points between t and $t+T$. From this equation, observing it depends on t only in the first and last factor and using the backward and forward Kolmogorov equations, we can derive an ODE similar to equation (8), with Π replaced by Υ:

$$\frac{d\Upsilon(t, t+T)}{dt} = -\tilde{Q}(t)\Upsilon(t, t+T) + \Upsilon(t, t+T)\tilde{Q}(t+T), \qquad (10)$$

where $\tilde{Q}(t)$ is the modified Q-matrix according to the goal and unsafe sets at time t. See [12, 13] for further details, and for a sketch of the algorithms that can be used to integrate the so-obtained equation.

Steady state properties In this tutorial, like in [12, 13], we have considered only time bounded operators. This limitation is a consequence of the very nature of Theorem 2, which holds only for a finite time horizon. However, there are situations in which we can extend the validity of the theorem to the whole time domain, but this extension depends on properties of the phase space of the fluid ODE [10, 14]. In those cases, we can prove convergence of the steady state behaviour of $Z^{(N)}$ to that of z, hence we can incorporate also operators dealing with steady state properties.

Checking these properties is relatively simple: we need to compute the unique fixed point $\mathbf{x}^*$ of the fluid ODE, which will be also the steady state measure of the limit fluid agent $z(t)$, assuming it is irreducible. When at steady state, the rates of $z(t)$ do not depend on time anymore, hence it becomes a time-homogeneous CTMC. Therefore, to model check a formula like $S_{\bowtie}(\varphi)$, we just need to model check φ against this time-homogeneous CTMC, with standard algorithms, and then compute the satisfaction of the steady state operator as in the standard model checking for CTMC [5] (see also [32]).

Running example. As an example, consider the steady state property $S_{\geq 0.75}(P_{\leq 0.1}(F^{[0,10]}a_{infected}))$. The fluid ODE for our example has a unique, globally attracting, fixed point $\mathbf{x}^* = (0.0209, 0.0767, 0.0383, 0.8641)$. Substituting this into the time-dependent Q matrix of the fluid agent z we obtain a time-homogeneous CTMC, for which the probability of $F^{[0,10]}a_{infected})$ is $(0.8526, 1, 1, 0.0276)$ and so the formula $P_{\leq 0.1}(F^{[0,10]}a_{infected})$ is true in state p and false in states s, d, i. By using ergodicity of z and the fact that $\mathbf{x}^*$ is also the limit steady state measure of an individual agent, hence the steady state measure of z, we can compute the steady state probability of satisfying $P_{\leq 0.1}(F^{[0,10]}a_{infected})$ as $0*0.0209+0*0.0767+0*0.0383+1*0.8641 = 0.8641$, which makes the steady state property true in all states.

8 Decidability and Convergence

In this section we will briefly discuss some theoretical features of the approximate model checking algorithm presented.

We will consider two main issues related to decidability and accuracy. Firstly, we will discuss the decidability of the algorithm to model check CSL specifications against ICTMC models. The fluid approximation of single agent properties is based on this as we have shown, and it is important to assess that this algorithm will yield an answer. Secondly, we must also consider the relationship between the truth of a CSL formula with a single agent derived through consideration of the fluid limit (i.e. representing the rest of the population only through the mean field) and the truth of the CSL formula for the single agent in a finite

Checked property	Fluid MC	Stat MC ($N = 100$)	Stat MC ($N = 1000$)
Kolmogorov Equations	~ 0.1 sec	~ 64 sec	~ 101 sec
$\mathbf{X}^{[0,T]} a_{infected}$	~ 0.06 sec	~ 6 sec	~ 24 sec
$\neg a_{infected} \mathbf{U}^{[0,T]} a_{patched}$	~ 0.05 sec	~ 5 sec	~ 20 sec

Table 1. Comparison of running times of the Fluid Model Checking algorithm of some properties discussed in the paper, with the running time for their statistical estimate (statistical model checking), for different population levels, computed from 10000 runs.

population model (i.e. with all agents represented explicitly). The approach is only useful if this relationship is close to identity.

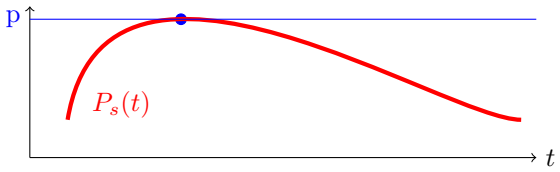
Efficiency and Decidability. Two desirable features of any model checking algorithm are decidability and computational efficiency. Efficiency, in particular, was the practical motivation of this work. Indeed, as can be seen in Table 1, already for the simple running example, the gain in computational time is remarkable: the fluid model checking approach is between 500 and 1000 times faster than statistical model checking for a population of 1000 nodes, with a negligible loss in accuracy. Furthermore, its complexity is independent of the population size, so that it can scale to very large systems.

The issue about decidability, instead, is more delicate, and depends heavily on the rate functions of the collective model and on the solution of the fluid ODE. In particular, the problem is with the nesting of CSL formulae. In order to model check a next or an until formula, containing nested temporal operators, we need to be able to perform a certain set of operations, specifically:

1. compute the set of zeros of $P(\psi, s, t) - p$ as a function of the time at which the formula $\varphi = P_{\bowtie p}(\psi)$ is evaluated;
2. check whether $P(\psi, s, t) < p$, $P(\psi, s, t) = p$, or $P(\psi, s, t) > p$ (to compute the truth function $\mathbf{T}(\varphi, s, t)$ for $\varphi = P_{\bowtie p}(\psi)$);
3. store in memory the truth function $\mathbf{T}(\varphi, s, t)$;

In order to deal with point 3 above, we need to guarantee that the number of zeros of the function $P(\psi, s, t) - p$ is finite in any finite time interval $[0, T]$. This is not true in general, but can be enforced by imposing additional regularity assumptions on the rate functions of the population model. Specifically, in [12, 13] we restricted the rate functions of the collective model to be (piecewise) *real analytic functions* [33]. This class of functions has nice closure properties (they are closed for arithmetic operations, integration, differentiation, and so on), which guarantees that all probability functions $P_s(t)$ will remain (piecewise) analytic. Furthermore, they are reasonably general, including most of the functions used in practice (for instance, polynomials, exponentials, and so on). Finally, they enjoy the property that either they are identically zero, or have only a finite number of zeros in any finite time interval. This settles point 3.

Points 1 and 2 above, instead, are much more delicate. First of all, finding all zeros of an analytic function is not an easy task, and in general may not be decidable. In particular, finding simple zeros (those around which a function change sign) is decidable (using interval methods [2, 41]), but we may not be able to find non-simple zeros, i.e. those in which the derivative is null, like local maxima or minima, see figure below.



Also the decidability of point 2 above, called the zero test problem, is unknown for analytic functions (in fact, its decidability is not known even for functions constructed using polynomials and the exponential, see [44]).

The way out this problem is to characterise precisely all the situations in which something bad can happen, and show that this are sufficiently *rare*. More precisely, we fixed a formula structure and the model parameters, and looked at what happens in terms of the thresholds p of the probability operators $P_{\bowtie p}$. If we have a formula with k temporal operators, then we have k such thresholds, which can take values in the hypercube $[0,1]^k$. We then looked at the subset R of points of $[0,1]^k$ for which we can guarantee that the algorithm terminates, and characterised it from a topological viewpoint. It turns out [12, 13] that R is an *open* subset of *Lebesgue measure one* of $[0,1]^k$. This means that almost any formula will be decidable, and furthermore that decidability is robust with respect to small perturbations of the probability thresholds. In [12, 13] this is termed *quasi-decidability*, and the CSL formulae which have thresholds probabilities that belong to the set R are called *robust*.

Convergence. We also investigated the limit behaviour of path probabilities and truth values of CSL formulae, evaluated for an individual agent $Z^{(N)}(t)$ in a finite population model, in the limit of $N \to \infty$. We proved that, in almost all cases, they converge to path probabilities and truth values computed for the limit individual agent $z(t)$. Convergence, however, does not hold always; it can fail exactly in those situations in which the limit model checking problem is not decidable. Given a CSL formula $\varphi = \varphi(\mathbf{p})$, with probability threshold arranged in a vector $\mathbf{p}$, we characterised the subset of $[0,1]^k$ of threshold for which convergence surely holds, obtaining that it coincides with the set R of thresholds making φ robust. More precisely, we have proved the following theorem [12, 13]:

Theorem 3. *Let $\mathcal{X}^{(N)}$ be a sequence of CTMC models and let $Z^{(N)}(t)$ and $z(t)$ be defined from $\mathcal{X}^{(N)}$ as in Section 4.2. Assume that $Z^{(N)}(t)$, $z(t)$ have piecewise analytic infinitesimal generator matrices.*

Let $\varphi(p_1, \ldots, p_k)$ be a robust CSL formula. Then, there exists an N_0 such that, for $N \geq N_0$ and each $s \in \mathcal{S}$

$$s, 0 \vDash_{Z^{(N)}} \varphi \Leftrightarrow s, 0 \vDash_z \varphi.$$

This theorem states that, for a given robust CSL formula φ, we can find an index N_0 such that, for populations larger than N_0, φ will hold in the limit model if and only if it holds in a model with population N. This shows that the method presented here is consistent with respect to asymptotic approximation. Unfortunately, characterising such N_0 is extremely difficult, see also the discussion in [12, 13] about erorr bounds.

9 Related Work

As this is a very new direction of research there is, as yet, only a small amount of related work. Model checking (time homogeneous) Continuous Time Markov Chains (CTMC) against Continuous Stochastic Logics (CSL) specifications has a long tradition in computer science [5, 4, 45]. At the core of our approach to study time-bounded properties there are similarities to that developed in [5], because we consider a transient analysis of a Markov chain whose structure has been modified to reflect the formula under consideration. But the technical details of the transient analysis, and even the structural modification, differ to reflect the time-inhomogeneous nature of the process we are studying.

To the best of the authors' knowledge, there has been no previous proposal of an algorithm to model check CSL formulae on a ICTMC. Nevertheless model checking of ICTMCs has been considered with respect to other logics. Specifically, previous work includes model checking of HML and LTL logics on ICTMC.

In [30], Katoen and Mereacre propose a model checking algorithm for Hennessy-Milner Logic on ICTMC. Their work is based on the assumption of piecewise constant rates (with a finite number of pieces) within the ICTMC. The model checking algorithm is based on the computation of integrals and the solution of algebraic equations with exponentials (for which a bound on the number of zeros can be found).

LTL model checking for ICTMC, instead, has been proposed by Chen *et al.* in [18]. The approach works for time-unbounded formulae by constructing the product of the CTMC with a generalized Büchi automaton constructed from the LTL formula, and then reducing the model checking problem to computation of reachability of bottom strongly connected components in this larger (pseudo)-CTMC. The authors also propose an algorithm for solving time bounded reachability similar to the one considered in this paper (for time-constant sets).

Our work is underpinned by the notion of fast simulation, which has previously been applied in a number of different contexts [22]. One recent case is a study of policies to balance the load between servers in large-scale clusters of heterogeneous processors [24]. These ideas also underlie the work of Hayden *et al.* in [25]. Here the authors extend the consideration of transient characteristics as captured by the fluid approximation, to approximation of first passage times,

in the context of models generated from the stochastic process algebra PEPA. Their approach for passage times of individual components is closely related to the fast simulation result and the work presented in this paper. The main difference is that they consider just path properties, described by deterministic automata (formally treated in [26]), which they solve by integrating ODEs.

10 Conclusions

In this tutorial we presented a new method to approximately model check properties of individual agents in a large population, exploiting mean field theory. This theory predicts that in the limit of an infinite population individual agents will decouple, evolving as independent CTMC connected only through the mean state of the system, described by the fluid ODE. This independence frees us from the necessity of representing the whole state space of the population, and instead we need only represent the state space of the individual agent. However, since the behaviour of this agent depends on the mean state of the system, its transition rates are not constants, but instead vary with time. Thus, in order to check properties for this limit model, we need to deal with a time-inhomogeneous CTMC. In this chapter we have presented a method to model check CSL formulae against ICTMC, whose complexity stems from the time dependency of truth values of temporal sub-formulae.

Our objective here has been to introduce the main ideas in an informal manner, explaining them by means of a simple example of a peer-to-peer network epidemic, in order to give the reader an intuition of how the approach works. The reader interested in the formal details is invited to study the fuller account given in the recent CONCUR paper [12] or its extended version [13].

The development of a fluid approximation for model checking, albeit only currently for time-bounded properties of individual agents opens the possibility of carrying out model checking on a wide range of population models that were previously extremely computationally costly or even beyond the scope of existing tools. Moreover there is a lot of potential of expanding the reach of model checking still further. Currently, we are extending the approach in several directions, including:

- moving beyond CSL to consider more complex path properties, for instance those expressed by Deterministic Timed Automata [19] (DTA), obtaining a logic for individual properties similar to asCSL [6] and CSL-TA [23];
- the lifting of individual specifications to the collective level, similarly to [32]. In this paper, the authors consider atomic collective properties stating that the expected fraction of agents satisfying a local CSL property meets a given bound $\bowtie p$. Instead of the expectation, we are considering approximations of the probability that the fraction of agents satisfying a local CSL property meets a given bound $\bowtie p$, using higher order fluid approximations, like the functional central limit [35] or linear noise approximation [50].

References

1. GNU Octave.
2. G. Alefeld and G. Mayer. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121:421–464, 2000.
3. A. Andreychenko, P. Crouzen, and V. Wolf. On-the-fly uniformization of time-inhomogeneous infinite Markov population models. In *Proceedings Ninth Workshop on Quantitative Aspects of Programming Languages, QAPL 2011*, volume 57 of *EPTCS*, page 1, 2011.
4. A. Aziz, V. Singhal, F. Balarin, R. Brayton, and A. Sangiovanni-Vincentelli. Verifying continuous time Markov chains. In *Proceedings of CAV96*, 1996.
5. C. Baier, B. Haverkort, H. Hermanns, and J.P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Proceedings of Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372. 2000.
6. Christel Baier, Lucia Cloth, Boudewijn R. Haverkort, Matthias Kuntz, and Markus Siegle. Model checking Markov chains with actions and state labels. *IEEE Trans. Software Eng.*, 33(4):209–224, 2007.
7. R. Bakhshi, L. Cloth, W. Fokkink, and B.R. Haverkort. Mean-field analysis for the evaluation of gossip protocols. In *Proceedings of the Sixth International Conference on the Quantitative Evaluation of Systems, QEST 2009*, pages 247–256. IEEE Computer Society, 2009.
8. R. Bakhshi, L. Cloth, W. Fokkink, and B.R. Haverkort. Mean-field framework for performance evaluation of push-pull gossip protocols. *Perform. Eval.*, 68(2):157–179, 2011.
9. M. Benaïm and J. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 2008.
10. M. Benaïm and J.Y. Le Boudec. On mean field convergence and stationary regime. *CoRR*, abs/1111.5710, 2011.
11. Ludec Berec. Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis. *Ecological Modelling*, 150(1–2):55–81, 2002.
12. L. Bortolussi and J. Hillston. Fluid model checking. In *Proceedings of CONCUR 2012*, 2012.
13. L. Bortolussi and J. Hillston. Fluid model checking. *CoRR*, 1203.0920, 2012.
14. L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective systems behaviour: a tutorial. *Performance Evaluation*, 2013.
15. Luca Bortolussi. On the approximation of stochastic concurrent constraint programming by master equation. volume 220, pages 163–180, 2008.
16. Luca Bortolussi and Alberto Policriti. Dynamical systems and stochastic programming: To ordinary differential equations and back. In Corrado Priami, Ralph-Johan Back, and Ion Petre, editors, *Transactions on Computational Systems Biology XI*, volume 5750 of *Lecture Notes in Computer Science*, pages 216–267. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-04186-0_11.
17. R.L. Burden and J. D. Faires. *Numerical analysis*. Thomson Brooks/Cole, 2005.
18. T. Chen, T. Han, J.P. Katoen, and A. Mereacre. LTL model checking of time-inhomogeneous Markov chains. In *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis, ATVA 2009*, volume 5799 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2009.
19. T. Chen, T. Han, J.P. Katoen, and A. Mereacre. Model checking of continuous-time Markov chains against timed automata specifications. *Logical Methods in Computer Science*, 7(1), 2011.

20. E. Clarke, A. Peled, and A. Grunberg. *Model Checking*. MIT press, 1999.
21. R.W.R. Darling. Fluid limits of pure jump Markov processes: A practical guide. *arXiv. org*, 2002.
22. R.W.R. Darling and J.R. Norris. Differential equation approximations for Markov chains. *Probability Surveys*, 5, 2008.
23. Susanna Donatelli, Serge Haddad, and Jeremy Sproston. Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. Software Eng.*, 35(2):224–240, 2009.
24. N. Gast and B. Gaujal. A mean field model of work stealing in large-scale systems. In *Proceedings of ACM SIGMETRICS 2010*, pages 13–24, 2010.
25. R.A. Hayden, A. Stefanek, and J.T. Bradley. Fluid computation of passage-time distributions in large Markov models. *Theor. Comput. Sci.*, 413(1):106–141, 2012.
26. Richard A. Hayden, Jeremy T. Bradley, and Allan Clark. Performance specification and evaluation with unified stochastic probes and fluid analysis. *IEEE Trans. Software Eng.*, 39(1):97–118, 2013.
27. J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of SysTems, QEST 2005*, pages 33 – 42, sept. 2005.
28. A. Jensen. Markov chains as an aid in the study of Markov processes. *Skandinavisk Aktuarietidskriff*, 36, 1953.
29. S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian approach to model checking biological systems. In *Proceedings of the 7th International Conference on Computational Methods in Systems Biology, CMSB 2009*, volume 5688 of *Lecture Notes in Computer Science*, pages 218–234, 2009.
30. J.-P. Katoen and A. Mereacre. Model checking hml on piecewise-constant inhomogeneous Markov chains. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2008*, volume 5215 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2008.
31. A. Kolesnichenko, A. Remke, P.T. de Boer, and B.R. Haverkort. Comparison of the mean-field approach and simulation in a peer-to-peer botnet case study. In *Proceedings of 8th European Performance Engineering Workshop, EPEW 2011*, volume 6977 of *LNCS*, pages 133–147. Springer, 2011.
32. Anna Kolesnichenko, Anne Remke, Pieter-Tjerk Boer de, and Boudewijn R. Haverkort. A logic for model-checking of mean-field models. In *Proceedings of the 43rd International Conference on Dependable Systems and Networks, DSN 2013*, 2013.
33. S. Krantz and P.R. Harold. *A Primer of Real Analytic Functions (Second ed.)*. Birkhäuser, 2002.
34. T. G. Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, 7:49–58, 1970.
35. T.G. Kurtz. *Approximation of population processes*. SIAM, 1981.
36. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, September 2004.
37. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.
38. Jean-Yves Le Boudec. *Performance Evaluation of Computer and Communication Systems*. EPFL Press, Lausanne, Switzerland, 2010.
39. M. Massink, D. Latella, A. Bracciali, M. Harrison, and J. Hillston. Scalable context-dependent analysis of emergency egress models. *Formal Aspects of Computing*, pages 1–36, in print.

40. MATLAB. *v. 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.
41. A. Neumaier. *Interval Methods for Systems of Equations*. University Press, Cambridge, 1990.
42. J. R. Norris. *Markov Chains*. Cambridge University Press, 1997.
43. H. Qian and E.L. Elson. Single-molecule enzymology: stochastic michaelis?menten kinetics. *Biophysical Chemistry*, 101:565–576, 2002.
44. D. Richardson. Zero tests for constants in simple scientific computation. *Mathematics in Computer Science*, 1(1):21–37, 2007.
45. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.
46. D.T.J. Sumpter. *From Bee to Society: An Agent-based Investigation of Honey Bee Colonies*. PhD thesis, University of Manchester, 2000.
47. Z. Szallasi, J. Stelling, and V. Periwal, editors. *System Modeling in Cellular Biology, From Concepts to Nuts and Bolts*. MIT Press, 2012.
48. Mirco Tribastone, Jie Ding, Stephen Gilmore, and Jane Hillston. Fluid rewards for a stochastic process algebra. *IEEE Trans. Software Eng.*, 38(4):861–874, 2012.
49. Mirco Tribastone, Stephen Gilmore, and Jane Hillston. Scalable differential analysis of process algebra models. *IEEE Trans. Software Eng.*, 38(1):205–219, 2012.
50. N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier, 1992.
51. A. P. A. van Moorsel and K. Wolter. Numerical solution of non-homogeneous Markov processes through uniformization. In *Proceedings of the 12th European Simulation Multiconference - Simulation- Past, Present and Future, ESM 1998*, pages 710–717. SCS Europe, 1998.

A Integrating the combined backward-forward Kolmogorov equation

In this appendix we will look more closely at the problem of numerically integrating the combined backward-forward Kolmogorov equation (8), needed to compute the time-dependent reachability probability for until formulae, see Section 6. Integrating this equation is necessary to check nested formulae. A result of this integration, for the running example, has been shown in Figure 7. We recall that the equation is

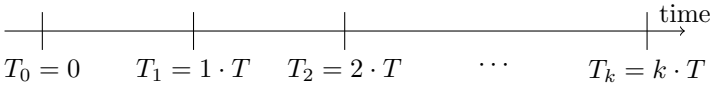
$$\frac{d\Pi(t, t+T)}{dt} = -Q(t)\Pi(t, t+T) + \Pi(t, t+T)Q(t+T),$$

which has to be solved from time t_0 to time t_1, with initial conditions $\Pi(t + 0, t_0 + T)$ computed using the forward equation.

In principle, this could be done by using one of the many ODE solvers available, e.g. those of Matlab$^{\text{TM}}$ [40] or Octave [1]. Practically, using one of those solvers, we have observed that in most of the cases, we obtain a plot like the one shown in Figure 9, in which the numerical error explodes. This is an indicator that equation (8) is, in general, very stiff [17], hence a stiff integration method has to be used. Unfortunately, this blow up phenomenon persisted even using the most accurate stiff integrators of Matlab$^{\text{TM}}$ or Octave, even with very high accuracy. We only obtained a reduction in the blow up speed.

In order to compute the trajectory in Figure 7, therefore, we need a different strategy. We present the idea in the following, showing how to compute the time-dependent reachability probability with time horizon T, without resorting to equation (8). The idea is to exploit the Chapman-Kolmogorov (CK) semigroup equations [42], $\Pi(t, t') = \Pi(t, t'')\Pi(t'', t')$, $t'' \in [t, t']$, in order to integrate the backward and the forward equations separately. The advantage of this is that the forward and the backward equations alone are, in general, quite stable.

For simplicity, we assume in the following that $t_0 = 0$ and $t_1 = k \cdot T$. The first operation is to split the time interval $[0, kT]$ into smaller time intervals, each of length T, as shown in the figure below:



Call T_j the time instant $j \cdot T$, fix $j \geq 1$, and pick $t \in [T_{j-1}, T_j]$. Applying the CK to times $t, T_j, t + T$, we get

$$\Pi(t, t+T) = \Pi(t, T_j)\Pi(T_j, t+T).$$

As T_j is a constant, in order to compute $\Pi(t, t+T)$ for $t \in [T_{j-1}, T_j]$, we can integrate separately the backward equation for $\Pi(t, T_j)$, $t \in [T_{j-1}, T_j]$, with initial value $\Pi(T_{j-1}, T_j)$, and the forward equation for $\Pi(T_j, t')$, $t' \in [T_j, T_{j+1}]$, with

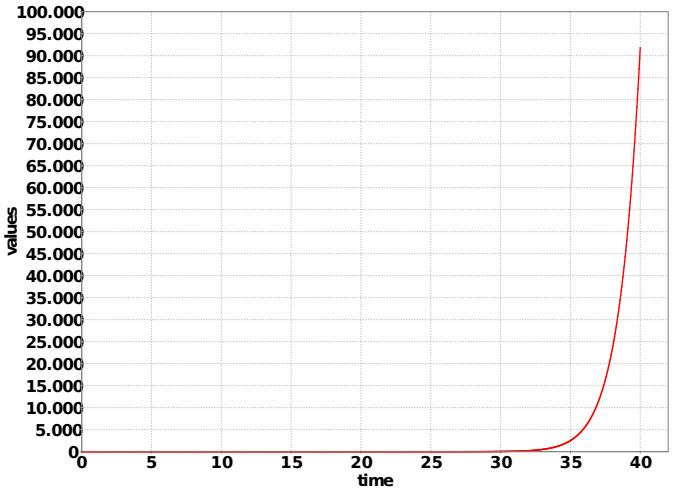


Fig. 9. Integration with a ODE numerical solver of the equation (8) for the formula $G^{[0,T_2]}\neg a_{infected}$, for $T_2 = 10$. Due to the high degree of stiffness of the equation and numerical instabilities, the error blows up.

initial value the identity matrix. These two equations can be solved simultaneously. Then, we can take the product of the so obtained matrices to compute $\Pi(t, t+T)$.

The full algorithm is a simple loop over j, observing that the initial conditions needed to integrate the backward equation are the last point computed by the forward equation in the previous iteration.

Practically, if we want just to visualize the result, we need to compute the product of $\Pi(t, T_j)$ and $\Pi(T_j, t+T)$ only at the sampled points of the function, generally a fixed grid of stepsize h. If, instead, we want to solve the equation $P(t) - p = 0$, in order to obtain the truth value, we need to take the product of the two matrices every time the root finding function (usually embedded in the ODE solver) needs to know the value of the function $P(t) - p$.