Christos G. Cassandras and John Lygeros

Stochastic Hybrid Systems: Recent Developments and Research Trends

CRC PRESS
Boca Raton London New York Washington, D.C.

CONTRIBUTORS

Henk A.P. Blom
NLR, The Netherlands

Christos G. Cassandras
Department of Manufacturing Engineering
Boston University, Boston, MA, U.S.A.

Joao P. Hespanha
Electrical & Computer Engineering Department
University of California, Santa Barbara, CA,
U.S.A.

John Lygeros
Department of Electrical & Computer
Engineering
University of Patras, Rio, Patras, Greece

Maria Prandini
Departimento di Eletronica e Informazione
Politecnico di Milano, Milano, Italy

Contents

0-8493-0052-5/00/$0.00+$.50

Chapter 1

Stochastic model checking

Joost-Pieter Katoen
RWTH Aachen

1.1 Introduction

When a program is run on a computer one of the most important considerations is whether it will work correctly. The fundamental question "when and why does software not work as expected?" has been the subject of intense research in computer science for decades [1]. The origins of a sound mathematical approach towards program correctness can be traced back to Turing in 1949 [59]. Early attempts to assess the correctness of computer programs were based on mathematical proof rules [37, 51, 3]. Proofs of realistic programs, though, are rather lengthy for programs of realistic size and require a large amount of human ingenuity.

In the early 1980s an alternative to using proof rules was proposed, independently by researchers in Europe [53] and the US [19], that given a (finite) model of a program systematically checks whether it satisfies a given property. This breakthrough was the first step towards the *automated verification* of concurrent programs. Typical questions treated by "*model checking*" are:

- safety: e.g., does a given mutual exclusion algorithm guarantee exclusive access to the shared resource?
- liveness: e.g., will a transmitted packet eventually arrive at the destination?

[1] According to Hoare, one of the pioneers in program verification and currently at Microsoft's laboratory, up to three-quarters of the 400 billion US dollar spent annually employing computer programmers in the US goes on debugging.

0-8493-0052-5/00/$0.00+$.50

- fairness: e.g., will a repetitive attempt to carry out a transaction be eventually granted?

Over the last two decades, model checking has received a lot of attention and is subject of study of a rapidly growing research community [21, 20].

How does model checking work? Given a model of the system (the "possible behavior") and a specification of the property to be considered (the "desirable behavior"), model checking is a technique that systematically checks the validity of the property in the model. Models are typically nondeterministic finite-state automata, consisting of a finite set of states and a set of transitions that describe how the system evolves from one state into another. These automata are usually composed of concurrent entities and are often generated from a high-level description language such as Petri nets, process algebra [50], PROMELA [38] or Statecharts [30]. Properties are typically specified in temporal logic such as CTL (Computation Tree Logic) [19], an extension of propositional logic that allows one to express properties that refer to the relative order of events. Statements can either be made about states or about paths, i.e., sequences of states that model an evolution of the system.

The basis of model checking is a systematic, usually exhaustive, state-space exploration to check whether the property is satisfied in each state of the model, thereby using effective methods (such as symbolic data structures, partial-order reduction or clever hashing techniques) to combat the state-space explosion problem. Due to unremitting improvements of underlying algorithms and data structures together with hardware technology improvements, model-checking techniques that a decade ago only worked for simple examples, are nowadays applicable to more realistic designs. State-of-the-art model checkers can handle state spaces of about 10^9 states using off-the-shelf technology. Using clever algorithms and tailored data structures, larger state spaces (up to 10^{476} states [55]) can be handled for specific problems and restricted types of correctness properties, namely so-called reachability properties.

1.1.1 Stochastic model checking

Whereas model checking focuses on the absolute correctness, in practice such rigid notions are hard, or even impossible, to guarantee. Instead, systems are subject to various phenomena of stochastic nature, such as message loss or garbling, unpredictable environments, faults, and delays. Correctness thus is of a less absolute nature. Accordingly, instead of checking whether system failures are impossible a more realistic aim is to establish, for instance, whether "the chance of shutdown occurring is at most 0.01%". Similarly, the question whether a distributed computation terminates becomes "does it eventually terminate with probability 1?". These queries can be checked using *stochastic* model checking, an automated technique for stochastic models in which state transitions encode the probability of making a transition between states rather than just the existence of such transition.

Stochastic model checking is based on conventional model checking, since it relies on reachability analysis of the underlying transition system, but must also entail the calculation of the actual likelihoods through appropriate numerical methods. In

addition to the qualitative statements made by conventional model checking, this provides the possibility to make quantitative statements about the system. Stochastic model checking uses extensions of temporal logics with probabilistic operators, affording the expression of these quantitative statements. Prominent examples of such extensions for CTL are PCTL [29] and CSL [4, 8].

Stochastic model checking is typically based on discrete-time and continuous-time Markov chains (DTMCs and CTMCs, respectively), or Markov decision processes (MDPs). Whereas Markov chains are fully stochastic, MDPs allow for nondeterminism. The former models are intensively used in performance and dependability analysis, whereas MDPs are of major importance in stochastic operations research [57, 52] and automated planning in AI [15]. Extensions of model checking to stochastic models originate from the mid 1980s [31, 60], first focusing on 0-1 probabilities, but later also considering quantitative properties [23]. During the last decade, these methods have been extended, refined and improved, and – most importantly – been supported by software tools [48, 35]. Currently, model checkers for DTMCs, CTMCs and MDPs do exist, and have been applied to several case studies ranging from randomized distributed algorithms [47] to dependability analysis of workstation clusters [33, 18]. With the currently available technology, models of 10^7 – 10^8 states can be successfully checked [18, 44]. This number can be increased significantly by applying aggressive abstraction techniques such as symmetry reduction, (bi)simulation relations or three-valued logics.

1.1.2 Topic of this survey

This chapter surveys the state-of-the-art in model checking of fully probabilistic models, i.e., stochastic models without nondeterminism. Discrete- and continuous-time models are covered as well as their extensions with costs. Syntax and semantics of (core fragments of) temporal logics are provided and the algorithms for checking (conditional) reachability properties are considered. This encompasses simple probabilistic reachability, i.e., what is the probability to reach a set of goal states, as well as time-bounded probabilistic reachability where in addition the goal state should be reached within a given deadline (which can be either discrete or continuous). For cost-extended models, we consider cost- and time-bounded reachability. For these models the following question is central: what is the probability to reach a given set of goal states within a given time-bound and a given bound on the cost? Bisimulation and simulation relations are defined and their relationship to the temporal logics covered in this chapter is established.

1.2 The discrete-time setting

We first consider discrete-time Markov chains and equip these with costs.

1.2.1 Discrete-time Markov chains

DTMCs. Let AP be a fixed, finite set of *atomic propositions*. The atomic propositions will be used to label states in a Markov chain, and are used to express the most elementary properties of a state. Such properties could be, e.g., "the discrete variable x lies between 0 and 201", or "the number of active processes in this state equals 7". It is assumed that the validity of an atomic proposition in a state can easily be determined (e.g., by inspection).

A (labelled) DTMC $\mathscr{D}$ is a tuple $(S, \mathbf{P}, L)$ where S is a finite set of *states*, $\mathbf{P} : S \times S \to [0,1]$ is a *probability matrix* such that $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ for all $s \in S$, and $L : S \to 2^{AP}$ is a *labelling* function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are valid in s. A path through a DTMC is a sequence [2] of states $\sigma = s_0 s_1 s_2 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all i. Let $Path^{\mathscr{D}}$ denote the set of all paths in DTMC $\mathscr{D}$. $\sigma[i]$ denotes the $(i+1)$th state of σ, i.e. $\sigma[i] = s_i$. Let $\Pr_s$ denote the unique probability measure on sets of paths that start in state s. This probability measure is defined in the standard way, see e.g., [46].

PCTL. Properties over DTMCs are expressed using an extension of temporal logic. Let $a \in AP$, probability $p \in [0,1]$, k be a natural number (or ∞) and $\bowtie$ a binary comparison operator in $\{\leq, \geq\}$. The syntax of Probabilistic CTL (PCTL) is defined by the grammar:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg \Phi \;\Big|\; \mathscr{P}_{\bowtie p}(\Phi \; \mathscr{U}^{\leq k} \; \Phi)$$

Thus, a formula in PCTL is built up from the basic formulas tt (true) and atomic proposition a, can be obtained by combining two PCTL-formulae by $\wedge$ (conjunction), prefixing a PCTL-formula with $\neg$ (negation), or by a so-called until-formula (denoted $\mathscr{U}$) that is contained in a $\mathscr{P}$-context which has as parameters a probability and a binary comparison operator. The other usual boolean connectives such as disjunction, implication and equivalence are derived in the usual way, e.g., $\Phi \vee \Psi = \neg(\neg\Phi \wedge \neg\Psi)$. The formula $\Phi \; \mathscr{U}^{\leq k} \; \Psi$ states a property over paths; a path $s_0 s_1 s_2 \ldots$ satisfies this formula if within k steps a Ψ-state is reached, and when all preceding states satisfy Φ. That is, when $\sigma[j]$ satisfies Ψ with $j \leq k$, and $\sigma[i]$ satisfies Ψ, for all indices i such that $i < j$. The unbounded until formula that is standard in temporal logics is obtained by taking k equal to ∞, i.e., $\Phi \mathscr{U} \Psi = \Phi \; \mathscr{U}^{\leq \infty} \; \Psi$. For the sake of simplicity, we do not consider the next operator.

The semantics of PCTL is defined by [29] a binary relation, denoted $\models$, between states of the DTMC and PCTL-formulae. The fact that $(s, \Phi) \in \models$ is denoted as $s \models \Phi$, and denotes that the PCTL-formula Φ holds in state s. The relation $\models$ is defined by structural induction on the formula Ψ in the following way:

$$\begin{array}{ll} s \models \mathrm{tt} & \text{for all } s \in S \\ s \models a & \text{iff } a \in L(s) \\ s \models \neg\Phi & \text{iff } s \not\models \Phi \end{array} \qquad \begin{array}{ll} s \models \Phi \wedge \Psi & \text{iff } s \models \Phi \wedge s \models \Psi \\ s \models \mathscr{P}_{\bowtie p}(\Phi \; \mathscr{U}^{\leq k} \; \Psi) & \text{iff } Prob^{\mathscr{D}}(s, \Phi \; \mathscr{U}^{\leq k} \; \Psi) \bowtie p \end{array}$$

[2] In this chapter, we do not dwell upon distinguishing finite and infinite paths.

$\mathscr{P}_{\bowtie p}(\Phi\ \mathscr{U}^{\leq k}\ \Psi)$ asserts that the probability measure of the paths that start in s and that satisfy $\Phi\ \mathscr{U}^{\leq k}\ \Psi$ meets the bound $\bowtie p$. Here,

$$\mathit{Prob}^{\mathscr{D}}(s, \Phi\ \mathscr{U}^{\leq k}\ \Psi) = \Pr_s\{\,\sigma \in \mathit{Path}^{\mathscr{D}} \mid \sigma \models \Phi\ \mathscr{U}^{\leq k}\ \Psi\,\} \quad .$$

Formula $\Phi\ \mathscr{U}^{\leq k}\ \Psi$ asserts that Ψ will be satisfied within k steps and that all preceding states satisfy Φ, i.e.:

$$\sigma \models \Phi\ \mathscr{U}^{\leq k}\ \Psi \ \text{ iff } \exists j \leq k.\ (\sigma[j] \models \Psi\ \wedge\ \forall i < j.\, \sigma[i] \models \Phi) \quad .$$

The hop-constraint $\leq k$ can easily be generalised towards arbitrary intervals. The same applies to cost- and real-time constraints that are considered later. For the sake of brevity, we refrain from going into the details of these generalisations.

Let us illustrate the expressiveness of PCTL by means of an abstract example. Suppose that the states in the DTMC under consideration are forbidden (= *illegal*) states, *goal* states and others. Assume that *illegal* and *goal* are atomic propositions. The logic PCTL allows to express e.g.,

- with probability ≥ 0.92, a goal state is reached via legal states only:

$$\mathscr{P}_{\geq 0.92}\,((\neg\,\mathit{illegal})\ \mathscr{U}\ \mathit{goal})$$

- ... in maximally 137 steps: $\mathscr{P}_{\geq 0.92}\,\big((\neg\,\mathit{illegal})\ \mathscr{U}^{\leq 137}\ \mathit{goal}\big)$
- ... once there, remain almost always for the next 31 steps:

$$\mathscr{P}_{\geq 0.92}\,\big((\neg\,\mathit{illegal})\ \mathscr{U}^{\leq 137}\ \mathscr{P}_{\geq 0.9999}\,\big(\Box^{\leq 31}\ \mathit{goal}\big)\big)$$

where $\mathscr{P}_{\geq p}(\Box^{\leq k}\Phi) = \mathscr{P}_{\leq 1-p}(\Diamond^{\leq k}\neg\Phi)$ and $\Diamond^{\leq k}\Phi$ stands for tt $\mathscr{U}^{\leq k}\ \Phi$. The formula $\Diamond^{\leq k}\Phi$ thus denotes that eventually a Φ-state will be reached within k steps and $\Box^{\leq k}\Phi$ asserts that the next k states all satisfy Φ.

Verifying hop-constrained probabilistic reachability. PCTL model checking [29] is carried out in the same way as verifying CTL [21] by recursively computing the set $\mathit{Sat}(\Phi) = \{\,s \in S \mid s \models \Phi\,\}$. This is done by means of a bottom-up recursive algorithm over the parse tree of Φ. Checking bounded until-formulas amounts to computing the least solution[3] of the following set of equations: $\mathit{Prob}^{\mathscr{D}}(s, \Phi\ \mathscr{U}^{\leq k}\ \Psi)$ equals 1 if $s \in \mathit{Sat}(\Psi)$,

$$\mathit{Prob}^{\mathscr{D}}(s, \Phi\ \mathscr{U}^{\leq k}\ \Psi) = \sum_{s' \in S} \mathbf{P}(s,s') \cdot \mathit{Prob}^{\mathscr{D}}(s', \Phi\ \mathscr{U}^{\leq k-1}\ \Psi) \qquad (1.1)$$

if $s \in \mathit{Sat}(\Phi \wedge \neg\Psi)$ and $k > 0$, and equals 0 otherwise. This probability can be computed as the solution of a regular system of linear equations by standard means such

[3] Strictly speaking, the function $s \mapsto \mathit{Prob}^{\mathscr{D}}(s, \Phi\,\mathscr{U}\,\Psi)$ is the least fixpoint of a higher-order function on $(S \to [0,1]) \to (S \to [0,1])$ where the underlying partial order on $S \to [0,1]$ is defined for $F_1, F_2 : S \to [0,1]$ by $F_1 \leq F_2$ iff $F_1(s) \leq F_2(s)$ for all $s \in S$.

as Gaussian elimination or can be approximated by an iterative approach (fixed point computation). The following alternative recipe is of interest to the stochastic models treated later on in this chapter.

For DTMC $\mathscr{D} = (S, \mathbf{P}, L)$ and PCTL formula Φ, let DTMC $\mathscr{D}[\Phi] = (S, \mathbf{P}', L)$ where if $s \not\models \Phi$, then $\mathbf{P}'(s,s') = \mathbf{P}(s,s')$ for all $s' \in S$, and if $s \models \Phi$, then $\mathbf{P}'(s,s) = 1$ and $\mathbf{P}'(s,s') = 0$ for all $s' \neq s$. We have $\mathscr{D}[\Phi][\Psi] = \mathscr{D}[\Phi \vee \Psi]$. Let $\pi^{\mathscr{D}}(s,k)(s')$ denote the probability of being in state s' after exactly k steps in DTMC $\mathscr{D}$ when starting in s, i.e. $\pi^{\mathscr{D}}(s,k)(s') = \Pr\{\, \sigma \in \mathit{Path}^{\mathscr{D}} \mid \sigma[k] = s' \wedge \sigma[0] = s \,\}$. This is known as the transient probability of state s' after k steps and can be obtained by $(\alpha_s \cdot \mathbf{P}^k)(s')$ where α_s is a probability vector that is one in state s, and 0 otherwise. It now follows that for any DTMC $\mathscr{D}$:

$$\mathit{Prob}^{\mathscr{D}}(s, \Phi\ \mathscr{U}^{\leq k}\ \Psi) = \sum_{s' \models \Psi} \pi^{\mathscr{D}[\neg\Phi \vee \Psi]}(s,k)(s') \tag{1.2}$$

Note that $\mathscr{D}[\neg\Phi \vee \Psi] = \mathscr{D}[\neg(\Phi \wedge \Psi)][\Psi]$, i.e. all $\neg(\Phi \vee \Psi)$-states and all Ψ-states in $\mathscr{D}$ are made absorbing. That is, the only transitions available in these states are self-loops with probability one. The former is correct since $\Phi\ \mathscr{U}^{\leq k}\ \Psi$ is violated as soon as some state is visited that neither satisfies Φ nor Ψ. The latter is correct since, once a Ψ-state in $\mathscr{D}$ has been reached (along a Φ-path) in at most k steps, then $\Phi\ \mathscr{U}^{\leq k}\ \Psi$ holds, regardless of which states will be visited later on. This modification of the system dynamics is closely related to the one in Chapter **??** of this volume for the numerical computation of reachability probabilisties in a more general class of stochastic hybrid systems.

Determining the set of states that satisfy $\Phi\ \mathscr{U}^{\leq k}\ \Psi$ thus amounts to computing $(\mathbf{P}^{\mathscr{D}[\neg\Phi\vee\Psi]})^k \cdot \underline{\iota}_\Psi$, where $\underline{\iota}_\Psi$ characterises $\mathit{Sat}(\Psi)$, i.e. $\iota_\Psi(s) = 1$ if $s \models \Psi$, and 0 otherwise. As iterative squaring is not attractive for stochastic matrices due to fill in [56], the product is typically computed in an iterative fashion: $\mathbf{P} \cdot (\ldots(\mathbf{P} \cdot \underline{\iota}_\Psi)))$.

1.2.2 Rewards

DMRM. A discrete-time Markov reward model (DMRM) $\mathscr{D}_r$ is a tuple $(\mathscr{D}, r)$ where $\mathscr{D}$ is a DTMC and $r : S \to \mathbb{R}_{\geq 0}$ is a *reward* assignment function. The quantity $r(s)$ indicates the reward that is earned on leaving state s. Note that rewards could also be attached to edges in a DTMC, but this does not increase expressivity. A path through a DMRM is a path through its DTMC, i.e., sequence of states $\sigma = s_0 s_1 s_2 \ldots$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for all i. The probability measure on sets of paths is defined as for DTMCs.

PRCTL. Let $r \in \mathbb{R}_{\geq 0}$ be a nonnegative reward bound, k a natural number, $p \in [0,1]$ and $a \in AP$. The syntax of Probabilistic Reward CTL (PRCTL) is defined by the following grammar:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathscr{P}_{\bowtie p}(\Phi\ \mathscr{U}^{\leq k}_{\leq r}\ \Phi) \;\Big|\; \mathscr{E}^{=k}_{\leq r}(\Phi)$$

Note that the binary until-operator is now equipped with two bounds: one on the number (k) of allowed hops to reach the goal states, and one on the allowed cumulated reward (r) before reaching the goal states. Formula $\mathscr{E}^{=k}_{\leq r}(\Phi)$ asserts that the

expected cumulated reward in Φ-states until the k-th transition is at most r. Thus, in order to check the validity of this formula for a given path, all visits to Φ-state are considered in the first k steps and the total reward that is obtained in these states; the reward earned in other states is not relevant, this also applies to Φ-states that are visited after having visited k states in the path. Whenever the expected value of this quantity over all paths that start in state s is at most r, state s is said to satisfy $\mathscr{E}^{=k}_{\leq r}(\Phi)$. The formal definition follows below. Other operators involving rewards that could be considered can be found in [2].

The semantics of the state-formulas of PRCTL that are common with PCTL is identical to the semantics for PCTL as presented above. Formula $\Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi$ asserts that Ψ will be satisfied within k steps, that all preceding states satisfy Φ, and that the cumulated reward until reaching the Ψ-state is at most r. Thus, for path σ we have:

$$\sigma \models \Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi \ \text{ iff } \exists j \leq k.\ \Big(\sigma[j] \models \Psi \wedge (\forall i < j.\, \sigma[i] \models \Phi) \ \wedge\ \textstyle\sum_{i=0}^{j-1} r(\sigma[i]) \leq r\Big)\ .$$

Similar as for PCTL:

$$s \models \mathscr{P}_{\bowtie p}(\Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi) \text{ if and only if } \mathit{Prob}^{\mathscr{D}_r}(s, \Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi) \bowtie p$$

where $\mathit{Prob}^{\mathscr{D}_r} = \mathit{Prob}^{\mathscr{D}}$. The semantics of the expected cumulated reward operator is defined by:

$$s \models \mathscr{E}^{=k}_{\leq r}(\Phi) \text{ if and only if } \sum_{i=0}^{k-1} \sum_{s' \models \Phi} \pi(s,i)(s') \cdot r(s') \ \leq\ r \quad .$$

Note that Φ plays the role of a state selector: only in states that satisfy Φ, the reward is considered. Rewards in the other states are ignored.

Multiple rewards. The logic PRCTL can easily be enhanced such that properties over models equipped with multiple reward structures can be treated. Suppose $\mathscr{C} = (\mathscr{D}, r_1, \ldots, r_k)$ is a DMRM with $k > 0$ reward assignment functions, and let $0 < j \leq k$. The reward operators of PRCTL can be generalised in a straightforward manner such that constraints on all k reward structures can be expressed in a single formula. For instance, the formula $\mathscr{E}^{=k}_{\leq r_1 \ldots \leq r_k}(\Phi)$ expresses that the expected cumulative reward in Φ-states until the k-th transition meets the upper bounds r_i of the i-th reward (for $0 < i \leq k$). The bounded-until operator can be generalised in a similar manner. Note that the hop-constraint (k) can also be considered as a reward in this setting.

Verifying hop- and reward-bounded probabilistic reachability. Checking the bounded until-operator in PRCTL amounts to computing the least solution of the following set of linear equations: $\mathit{Prob}^{\mathscr{D}_r}(s, \Phi\ \mathscr{U}^{\leq k}_{\leq r}\ \Psi)$ equals 1 if $s \in \mathit{Sat}(\Psi)$,

$$\mathit{Prob}^{\mathscr{D}_r}(s, \Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi) = \textstyle\sum_{s' \in S} \mathbf{P}(s,s') \cdot \mathit{Prob}^{\mathscr{D}_r}(s', \Phi\ \mathscr{U}^{\leq k-1}_{\leq r-r(s)}\ \Psi) \qquad (1.3)$$

if $s \in \mathit{Sat}(\Phi \wedge \neg\Psi)$, $k > 0$, and $r(s) \geq r$, and equals 0 otherwise.

Let $\pi^{\mathscr{D}_r}_r(s,k)(s') = \Pr_s\{\, \sigma \in \mathit{Path}^{\mathscr{D}} \mid \sigma[k] = s' \wedge \sum_{i=0}^{k-1} r(\sigma[i]) \leq r \,\}$. Then for any DMRM $\mathscr{D}_r$:

$$\mathit{Prob}^{\mathscr{D}_r}(s, \Phi\ \mathscr{U}^{\leq k}_{\leq r}\Psi) = \sum_{s' \models \Psi} \pi^{\mathscr{D}_r[\neg\Phi \vee \Psi]}_r(s,k)(s') \qquad (1.4)$$

where for formula Φ, the DMRM $\mathscr{D}_r[\Phi]$ is defined as $(\mathscr{D}[\Phi], r')$ with DTMC $\mathscr{D}[\Phi]$ as before and $r'(s) = r(s)$ if $s \not\models \Phi$ and 0 otherwise. That is, all states that are made absorbing obtain a zero reward.

The mathematical characterization (1.4) has a strong resemblance with the result for DTMCs, cf. equation (1.2). Nevertheless, computing the transient reward probabilities $\pi_r^{\mathscr{D}_r}(s,k)(s')$ is a bit more involved than computing transient probabilities $\pi^{\mathscr{D}}$. We sketch two algorithms. The first algorithm is based on the following recursion scheme [58]. Assume the rewards are either natural or rational numbers. Let $p_r(s,k)$ be the probability to be in state s at the k-th step while having incurred an accumulated cost exactly r. Then:

$$\pi_r^{\mathscr{D}_r}(s,k)(s') = \sum_{i=1}^{r} p_i(s',k) \quad .$$

Let $p_i(s',1) = 1$ if $s' = s$ and $i = r(s)$ and 0 otherwise. Then for $k \geq 0$:

$$p_i(s,k+1) = \sum_{s' \in S} p_{i-r(s')}(s',k) \cdot \mathbf{P}(s',s) \quad .$$

Alternatively, we exploit the following adaptation of the path graph generation algorithm [54]. The basic idea is to unfold the DMRM while keeping track of the cumulative reward so far. In the i-th step, only Φ-successors of state s are "unfolded" if $i < k-1$, and if $i = k-1$, only Ψ-successors of state s are considered. Vertices that have the same cumulated reward are grouped. The groups have the form $(R, \{(s_1,p_1),\ldots,(s_m,p_m)\}) \in V_h$ where h is the unfolding depth, the root vertex is $(0,\{s,1\})$ (only element in V_0), $\sum_i p_i$ is the probability to gain R reward in h transitions, and p_i is the probability to reach s_i when starting from s. The unfolding is stopped on reaching depth k. On termination, the total probability of reward r can easily be obtained from the groups of vertices.

Checking the operator on expected cumulated reward amounts to solving a system of linear equations. The quantity $\sum_{i=0}^{k-1} \sum_{s \models \Phi} \pi(s,i)(s') \cdot r(s')$ can be characterized as the smallest solution of the following system of linear equations:

$$H(s,k) = \begin{cases} 0 & \text{if } n = 0 \\ r(s) + \sum_{s' \in S} \mathbf{P}(s,s') \cdot H(s',k-1) & \text{if } s \in \mathit{Sat}(\Phi) \ \wedge \ k > 0 \\ \sum_{s' \in S} \mathbf{P}(s,s') \cdot H(s',k-1) & \text{if } s \notin \mathit{Sat}(\Phi) \ \wedge \ k > 0 \end{cases}$$

1.3 The continuous-time setting

1.3.1 Continuous-time Markov chains

CTMCs. A (labelled) CTMC $\mathscr{C}$ is a tuple $(S, \mathbf{R}, L)$ where S and L are as for DTMCs, and $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ is the *rate matrix*. The exit rate $E(s) = \sum_{s' \in S} \mathbf{R}(s,s')$

denotes that the probability of taking a transition from s within t time units equals $1-e^{-E(s)\cdot t}$. If $\mathbf{R}(s,s')>0$ for more than one state s', a race between the outgoing transitions from s exists. That is, the probability $\mathbf{P}(s,s')$ of moving from s to s' in a single step equals the probability that the delay of going from s to s' "finishes before" the delays of any other outgoing transition from s. The probability of moving from state s to a state s' is $\mathbf{P}(s,s') = \frac{\mathbf{R}(s,s')}{E(s)}$. The probability of making a transition from state s to s' within time t is given by:

$$\frac{\mathbf{R}(s,s')}{E(s)} \cdot (1-e^{-E(s)\cdot t})$$

The time-abstract behaviour of a CTMC is described by its embedded DTMC. For CTMC $\mathscr{C} = (S,\mathbf{R},L)$, the *embedded* DTMC is given by $emb(\mathscr{C}) = (S,\mathbf{P},L)$, where $\mathbf{P}(s,s') = \mathbf{R}(s,s')/E(s)$ if $E(s)>0$, and $\mathbf{P}(s,s)=1$ and $\mathbf{P}(s,s')=0$ for $s\neq s'$ if $E(s)=0$.

A path in a CTMC is an alternating sequence $\sigma = s_0\,t_0\,s_1\,t_1\,s_2\ldots$ with $\mathbf{R}(s_i,s_{i+1})>0$ and $t_i \in \mathbb{R}_{>0}$ for all i. The time stamps t_i denote the amount of time spent in state s_i. Let $Path^{\mathscr{C}}$ denote the set of paths through $\mathscr{C}$. $\sigma@t$ denotes the state of σ occupied at time t, i.e. $\sigma@t = \sigma[i]$ with i the smallest index such that $t \leq \sum_{j=0}^{i} t_j$. Let $\Pr_s$ denote the unique probability measure on sets of paths that start in s [10].

CSL. Let a, p and $\bowtie$ be as before and $t \in \mathbb{R}_{\geq 0}$ (or ∞). The syntax of Continuous Stochastic Logic (CSL [4, 10]) is:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathscr{P}_{\bowtie p}(\Phi\ \mathscr{U}^{\leq t}\ \Phi)$$

The semantics of CSL for the boolean operators is identical to that for PCTL. For the time-bounded until-formula:

$$s \models \mathscr{P}_{\bowtie p}(\Phi\ \mathscr{U}^{\leq t}\ \Psi) \text{ if and only if } Prob^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi) \bowtie p$$

$Prob^{\mathscr{C}}(\cdot)$ is defined in a similar way as for DTMCs:

$$Prob^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi) = \Pr_s\{\,\sigma \in Path^{\mathscr{C}} \mid \sigma \models \Phi\ \mathscr{U}^{\leq t}\ \Psi\,\}.$$

It is not difficult to establish that the set indicated on the right-hand side is measurable. The operator $\mathscr{U}^{\leq t}$ is the real-time variant of the PCTL operator $\mathscr{U}^{\leq k}$; $\Phi\ \mathscr{U}^{\leq t}\ \Psi$ asserts that Ψ will be satisfied at some time instant in the interval $[0,t]$ and that at all preceding time instants Φ holds:

$$\sigma \models \Phi\ \mathscr{U}^{\leq t}\ \Psi \text{ iff } \exists x \leq t.\ (\sigma@x \models \Psi \wedge \forall y < x.\ \sigma@y \models \Phi)\ .$$

Note that the standard until operator is obtained by taking t equal to ∞.

Model checking time-bounded until properties. CSL model checking [10, 13] is performed in the same way as for CTL [21] and PCTL [29], by recursively computing the set $Sat(\Phi)$. For the boolean operators this is exactly as for CTL and for unbounded until (i.e., $t{=}\infty$) this is exactly as for PCTL. Checking time-bounded until

formulas is based on determining the least solution of the following set of Volterra integral equations: $\mathit{Prob}^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi)$ equals 1 if $s \in \mathit{Sat}(\Psi)$,

$$\mathit{Prob}^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi) = \int_0^t \sum_{s' \in S} \mathbf{P}(s,s') \cdot E(s) \cdot e^{-E(s)\cdot x} \cdot \mathit{Prob}^{\mathscr{C}}(s',\Phi\ \mathscr{U}^{\leq t-x}\ \Psi)\, dx$$

if $s \in \mathit{Sat}(\Phi \wedge \neg\Psi)$, and equals 0 otherwise. Here, $E(s)\cdot e^{-E(s)\cdot x}$ denotes the probability density of taking some outgoing transition from s at time x. Note the resemblance with equation (1.1) for the PCTL bounded until operator. For CTMC $\mathscr{C} = (S,\mathbf{R},L)$ and CSL formula Φ let CTMC $\mathscr{C}[\Phi] = (S,\mathbf{R}',L)$ with $\mathbf{R}'(s,s') = \mathbf{R}(s,s')$ if $s \not\models \Phi$ and 0 otherwise. Note that $\mathit{emb}(\mathscr{C}[\Phi]) = \mathit{emb}(\mathscr{C})[\Phi]$. It has been shown in [13] that for a given CTMC $\mathscr{C}$ and state s in $\mathscr{C}$, the measure $\mathit{Prob}^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi)$ can be calculated by means of a transient analysis of the CTMC $\mathscr{C}'$, which can easily be derived from $\mathscr{C}$ using the $[\cdot]$ operator. Let $\pi^{\mathscr{C}}(s,t)(s')$ denote the probability of being in state s' at time t given that the system started in state s, i.e. $\pi^{\mathscr{C}}(s,t)(s') = \Pr_s\{\,\sigma \in \mathit{Path}^{\mathscr{C}} \mid \sigma@t = s'\,\}$. It follows that for any CTMC $\mathscr{C}$:

$$\mathit{Prob}^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi) = \sum_{s' \models \Psi} \pi^{\mathscr{C}[\neg\Phi\vee\Psi]}(s,t)(s'). \tag{1.5}$$

Note that this is just the generalization of equation (1.2) to the continuous setting. Verifying time-bounded until-properties in a CTMC thus amounts to computing transient state probabilities in a derived CTMC. These probabilities are obtained by solving a linear differential equation that can efficiently (and numerically stable) be determined by uniformisation [28]. Uniformisation is a transformation of a CTMC into a DTMC: For CTMC $\mathscr{C} = (S,\mathbf{R},L)$ the *uniformised* DTMC is given by $\mathit{unif}(\mathscr{C}) = (S,\mathbf{P},L)$ where $\mathbf{P} = \mathbf{I} + \mathbf{Q}/q$ for $q \geq \max\{E(s) \mid s \in S\}$ and $\mathbf{Q} = \mathbf{R} - \mathit{diag}(E)$. Here, $\mathbf{I}$ denotes the identity matrix. The *uniformisation rate* q is determined by the state with the shortest mean residence time. All (exponential) delays in the CTMC $\mathscr{C}$ are normalised with respect to q. That is, for each state $s \in S$ with $E(s) = q$, one epoch in $\mathit{unif}(\mathscr{C})$ corresponds to a single exponentially distributed delay with rate q, after which one of its successor states is selected probabilistically. As a result, such states have no self-loop in the DTMC. If $E(s) < q$—this state has on average a longer state residence time than $\frac{1}{q}$—one epoch in $\mathit{unif}(\mathscr{C})$ might not be "long enough". Hence, in the next epoch these states might be revisited and, accordingly, are equipped with a self-loop with probability $1 - \frac{E(s)}{q}$. Note the difference between the embedded DTMC $\mathit{emb}(\mathscr{C})$ and the uniformised DTMC $\mathit{unif}(\mathscr{C})$: whereas the epochs in $\mathscr{C}$ and $\mathit{emb}(\mathscr{C})$ coincide and $\mathit{emb}(\mathscr{C})$ can be considered as the time-abstract variant of $\mathscr{C}$, a single epoch in $\mathit{unif}(\mathscr{C})$ corresponds to a single exponentially distributed delay with rate q in $\mathscr{C}$. It now follows that for any CTMC $\mathscr{C}$:

$$\mathit{Prob}^{\mathscr{C}}(s,\Phi\ \mathscr{U}^{\leq t}\ \Psi) = \sum_{k=0}^{\infty} \gamma(k,q\cdot t) \cdot \mathit{Prob}^{\mathit{unif}(\mathscr{C})}(s,\Phi\ \mathscr{U}^{\leq k}\ \Psi) \tag{1.6}$$

where $\gamma(k,q\cdot t)$ denotes the Poisson probability of taking k jumps in the DTMC $\mathit{unif}(\mathscr{C})$ in the interval $[0,t)$, i.e., $\gamma(k,q\cdot t) = e^{-q\cdot t}\cdot(q\cdot t)^k/k!$.

Example. Consider two clusters of workstations that are connected via a backbone connection. Each cluster consists of N workstations, connected in a star topology with a central switch that provides the interface to the backbone. Each of the components of the system (workstations, switches, and backbone) can break down. There is single repair unit that takes care of repairing failed components. The computing power of the cluster is over-dimensioned, in order to be able to accommodate varying levels of traffic volume, as well as to cope with component failures. The system operation is subject to the following informal constraints:

- In order to provide *minimum* quality of service (QoS), at least k ($k < N$) workstations have to be operational, and these workstations have to be connected.

- *Premium* quality of service requires at least N operational workstations, with the same connectivity constraints as mentioned above.

Figure 1.1 indicates the verification times (in seconds) for varying sizes of N (indicated by the absolute number of states of the CTMC). The property that has been checked is: $\mathscr{P}_{\geq .99}(\mathit{Minimum}\,\mathscr{U}^{\leq t}\,\mathit{Premium})$ for various t. The experiments were conducted on a computer with an Intel P4 3 GHz. processor, 2 GB of RAM running SuSe Linux ver. 9.1. Note that for large values of t, the CTMC may have already reached an equilibrium. This information can be used during the model checking in order to speed up the verification process [45].

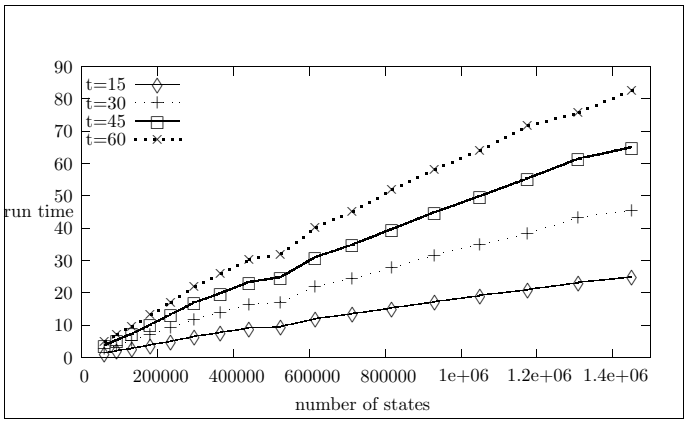

FIGURE 1.1: Verification times for time-bounded until versus the CTMC state space size for various time bounds

1.3.2 Rewards

Costs can be attached to CTMCs to states and to transitions. Cost rates associated with states indicate the cost per unit of time the system stays in that state; rewards associated to edges—these rewards are also called impulse rewards—are fixed and independent of time. For the sake of simplicity, we just consider reward rates. All results and definitions can, however, easily be extended to incorporate impulse rewards as well (see e.g., [22]).

CMRM. A continuous-time Markov reward model (CMRM) $\mathscr{C}_r$ is a tuple $(\mathscr{C}, r)$ where $\mathscr{C}$ is a CTMC and $r : S \to \mathbb{R}_{\geq 0}$ is a reward assignment function (as before). The state reward structure is a function r that assigns to each state $s \in S$ a reward rate $r(s)$ such that if t time-units are spent in state s, a reward $r(s) \cdot t$ is acquired. A path through a CMRM is a path through its underlying CTMC. Let $\sigma = s_0\, t_0\, s_1\, t_1 \ldots$ be a path. For $t = \sum_{j=0}^{k-1} t_j + t'$ with $t' \leq t_k$ we define $r(\sigma, t) = \sum_{j=0}^{k-1} t_j \cdot r(s_j) + t' \cdot r(s_k)$, the cumulative reward along σ up to time t.

CSRL. Let a, p and $\bowtie$ be as before and $t, r \in \mathbb{R}_{\geq 0}$ (or ∞). The syntax of Continuous Stochastic Reward Logic (CSRL [13]) is:

$$\Phi ::= \mathrm{tt} \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathscr{P}_{\bowtie p}(\Phi \;\mathscr{U}^{\leq t}_{\leq r}\; \Phi)$$

The semantics of the time- and reward-bounded until-operator is given by:

$$\sigma \models \Phi \;\mathscr{U}^{\leq t}_{\leq r}\; \Psi \text{ iff } \exists x \leq t.\ (\sigma@x \models \Psi \wedge \forall y < x.\ \sigma@y \models \Phi \wedge r(\sigma, x) \leq r)\ .$$

Note that the standard until operator is obtained by taking t equal to ∞.

Before continuing with presenting the algorithmic approach for checking cost- and time-bounded until-formulas the following intermezzo is of relevance.

Duality of time and rewards. In the discrete-time setting we have seen that a hop constraint can just be considered as an additional reward constraint. In the continuous-time setting there is also a strong relationship between rewards and time. This is referred to as duality. The basic idea behind this duality, inspired by [14], is that the progress of time can be regarded as the earning of reward and vice versa. First we obtain a duality result for CMRMs where all states have a positive reward. After that we consider the (restricted) applicability of the duality result to CMRMs with zero rewards. Let $\mathscr{C} = ((S, \mathbf{R}, L), r)$ be a CMRM that satisfies $r(s) > 0$ for any state s. Define CMRM $\mathscr{C}^{-1} = ((S, \mathbf{R}', L), r')$ that results from $\mathscr{C}$ by: (i) rescaling the transition rates by the reward of their originating state (as originally proposed in [14]), i.e., $\mathbf{R}'(s, s') = \mathbf{R}(s, s')/r(s)$ and, (ii) inverting the reward structure, i.e., $r'(s) = 1/r(s)$. Intuitively, the transformation of $\mathscr{C}$ into $\mathscr{C}^{-1}$ stretches the residence time in state s with a factor that is proportional to the reciprocal of its reward $r(s)$ if $r(s) > 1$, and it compresses the residence time by the same factor if $0 < r(s) < 1$. The reward structure is changed similarly. Note that $\mathscr{C} = (\mathscr{C}^{-1})^{-1}$.

One might interpret the residence of t time units in $\mathscr{C}^{-1}$ as the earning of t reward in state s in $\mathscr{C}$, or (reversely) an earning of a reward r in state s in $\mathscr{C}$ corresponds to a residence of r in $\mathscr{C}^{-1}$. Thus, the notions of time and reward in $\mathscr{C}$ are reversed in $\mathscr{C}^{-1}$. Accordingly [13], for any CMRM $\mathscr{C} = ((S, \mathbf{R}, L), r)$ with $r(s) > 0$ for all $s \in S$ and CSRL state-formula Φ:

$$Sat^{\mathscr{C}}(\Phi) \;=\; Sat^{\mathscr{C}^{-1}}(\Phi^{-1}) \tag{1.7}$$

(Recall that $Sat(\Phi) = \{\, s \in S \mid s \models \Phi \,\}$). Thus, verifying cost-bounded until-formulas on CMRMs with only non-zero rewards can be done in the same way as checking time-bounded until-formulas on CTMCs. If CMRM $\mathscr{C}$ contains states equipped with a zero reward, this duality result does not hold, as the reverse of earning a zero reward in $\mathscr{C}$ when considering Φ should correspond to a residence of 0 time units in $\mathscr{C}^{-1}$ for Φ^{-1}, which — as the advance of time in a state cannot be halted — is in general not possible. However, if for each sub-formula of the form $\Phi\,\mathscr{U}^{\leq t}_{\leq r}\Psi$ we have $Sat^{\mathscr{C}}(\Phi) \subseteq \{\, s \in S \mid r(s) > 0 \,\}$, i.e., all Φ-states are positively rewarded then equation (1.7) applies. Here, $\mathscr{C}^{-1}$ is defined by setting $\mathbf{R}'(s,s') = \mathbf{R}(s,s')$ and $r'(s) = 0$ in case $r(s) = 0$ and as defined above otherwise.

Verifying time- and cost-bounded until properties. Checking time- and cost-bounded until formulas is based on determining the least solution of the following set of Volterra integral equations: $Prob^{\mathscr{C}}(s, \Phi\,\mathscr{U}^{\leq t}_{\leq k}\Psi)$ equals 1 if $s \in Sat(\Psi)$,

$$Prob^{\mathscr{C}}(s, \Phi\,\mathscr{U}^{\leq t}_{\leq k}\Psi) = \int_{K(s)} \sum_{s' \in S} \mathbf{P}(s,s') \cdot E(s) \cdot e^{-E(s)\cdot x} \cdot Prob^{\mathscr{C}}(s', \Phi\,\mathscr{U}^{\leq t-x}_{\leq r-r(s)\cdot x}\Psi)\, dx$$

if $s \in Sat(\Phi \wedge \neg\Psi)$, and equals 0 otherwise. Here $K(s) = \{\, x \leq t \mid r(s)\cdot x \leq r \,\}$ is the subset of $[0,t]$ whose reward lies in $[0,r]$. It is not difficult to see that for $r = \infty$, the above integral equation is exactly the one obtained for time-bounded until properties in CTMCs.

Let now $\pi_r^{\mathscr{C}_r}(s,t)(s') = \Pr_s\{\, \sigma \in Path^{\mathscr{C}} \mid \sigma@t = s' \wedge r(\sigma,t) \leq r \,\}$. Then for any CMRM $\mathscr{C}_r$:

$$Prob^{\mathscr{C}_r}(s, \Phi\,\mathscr{U}^{\leq t}_{\leq r}\Psi) = \sum_{s' \models \Psi} \pi_r^{\mathscr{C}_r[\neg\Phi\vee\Psi]}(s,t)(s') \tag{1.8}$$

where for formula Φ, the CMRM $\mathscr{C}_r[\Phi]$ is defined as $(\mathscr{C}[\Phi], r')$ with CTMC $\mathscr{C}[\Phi]$ as before and $r'(s) = r(s)$ if $s \not\models \Phi$ and 0 otherwise. That is, all states that are made absorbing obtain a zero reward (like in the discrete case). The remaining problem, however, is to compute the transient reward probabilities $\pi_r^{\mathscr{C}_r}(s,t)(s')$. We describe an algorithm that is heavily based on that for DMRMs: discretization together with a recursive scheme. A generalisation of the path graph generation algorithm for CMRMs can be found in [22].

A discretization approach. This method is based on the algorithm by Tijms and Veldman [58] that discretizes both the time interval and the accumulated reward as multiples of the same step size $d > 0$. d is chosen such that the probability of more than one transition in the CMRM in an interval of length d is negligible. Using this discretization, the probability of accumulating at most r reward at time t is given by:

$$\sum_{i=1}^{i=R} p'_i(s,T) \cdot d \text{ where } R = \frac{r}{d} \text{ and } T = \frac{t}{d}$$

$p'_i(s,T)$ is the probability of being in state s at discretized time T with cumulated discretized reward i. $p'_i(s,T)$ is defined in a similar way as for the discrete setting,

with the exception that the transition probabilities are determined differently. The recursive equation now becomes for $i \geq 0$:

$$\begin{aligned} p'_i(s,k{+}1) = \quad & p'_{i-r(s)}(s,k)\cdot(1-E(s)\cdot d) \\ & +\textstyle\sum_{s'} p'_{i-r(s')}(s',k)\cdot\mathbf{R}(s',s)\cdot d \end{aligned}$$

For the CMRM to be in state s at the $(k{+}1)$-st time-instant either the CMRM was in state s in the k-th time-instant and remained there for d time-units without traversing a self-loop (the first summand) or it was in state s' and has moved to state s in that period (the second summand). Given that the cumulative reward at the $(k{+}1)$-st time-instant is i, the cumulative reward in the k-th time-instant is approximated by $i-r(s)$ in the first summand and $i-r(s')$ in the second summand. If this recursive method is implemented by using matrices to store $p(k{+}1)$ and $p(k)$, then it is necessary to have integer rewards only.

Example. The time complexity of the discretization algorithm is cubic in the number of states in the CMRM and proportional to d^{-2}. To illustrate the complexity of this algorithm on a realistic example, Figure 1.2 depicts the verification times for a CMRM with 276 states. The rewards have been used to model power consumption, and the model has been obtained from a dynamic power management strategy in mobile phones. The property that has been checked is $\mathscr{P}_{>0.5}(\Diamond^{\leq 2000}_{\leq 200}\, \mathit{done})$. This CSRL-formula asserts that the probability to eventually reach a *done*-state within 2000 ms such that at most 200 mJ is spent, is at least $\frac{1}{2}$. Further details on the case study can be found in [22, 1]. Figure 1.3 plots the verification time for an increasing state space size. For the error bound 10^{-3}, time increase is negligible, cf. the plot close to the zero y-axis. Note the significant difference in state space size with the continuous-time setting without rewards. (Expected rewards can be checked in a much faster way as just a system of linear equations needs to be solved where the number of variables is linear in the size of the state space.)

1.3.3 Time-inhomogenity

ICMRM. A time-inhomogeneous CMRM (ICMRM, for short) is a triple $(S,\mathbf{R},r)$ where: S is a finite set of states, $\mathbf{R} : S\times S\times\mathbb{R}_{\geq 0}\to\mathbb{R}_{\geq 0}$ is a *time-indexed* rate matrix, and $r : S\times\mathbb{R}_{\geq 0}\to\mathbb{R}_{\geq 0}$ is a *time-indexed* reward assignment function. Main difference with the continuous-time models treated so far in this survey is that both the transition rates and the rewards are time-dependent. The exit rate $E(s,d) = \sum_{s'\in S}\mathbf{R}(s,s',d)$ denotes that the probability of taking a transition from state s within t time units at time d equals $1-e^{-E(s,d)\cdot t}$. The probability of moving from state s to a state s' at time d is given by $\mathbf{P}(s,s',d) = \frac{\mathbf{R}(s,s',d)}{E(s,d)}$. The probability of making a transition from state s to s' at time d within the next t time units is defined by:

$$\frac{\mathbf{R}(s,s',d)}{E(s,d)}\cdot(1-e^{-E(s,d)\cdot t})$$

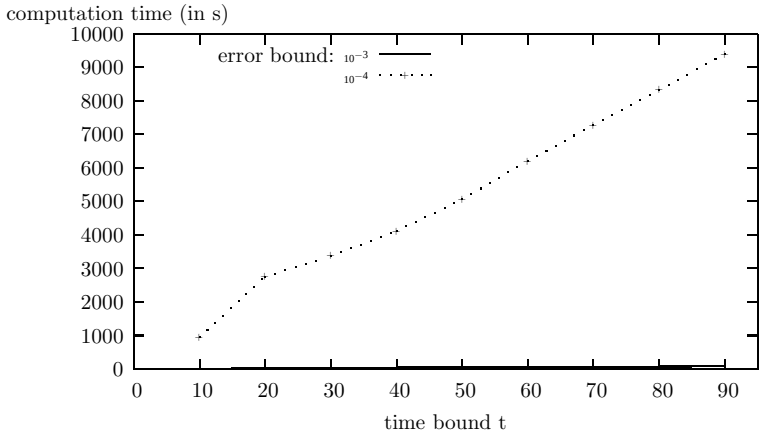

FIGURE 1.2: Computation times versus time bound

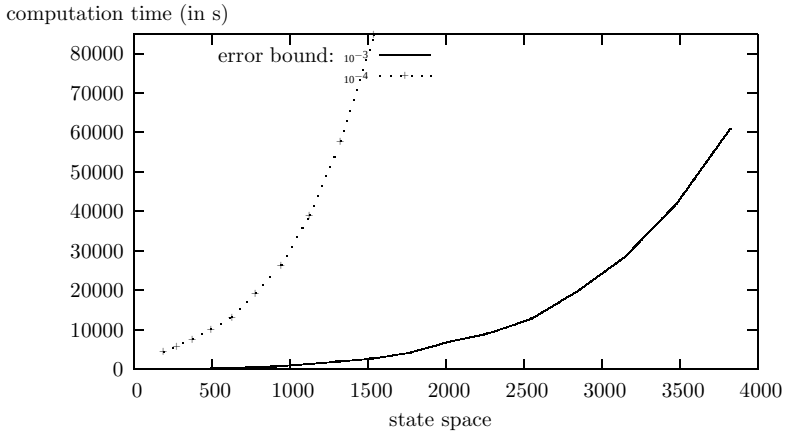

FIGURE 1.3: Computation times versus state space size of CMRM

The earned reward when staying for d time units in state s is given by:

$$d \cdot \int_0^d r(s,u)\, du$$

Note that in case the rate matrix $\mathbf{R}$ and the reward-assignment function r are effectively independent from t, we obtain a time-homogeneous CMRM.

Verifying time- and cost-bounded reachability. Properties of time-inhomogeneous CMRM models can be stated in the logic CSRL. We have seen that the time- and cost-bounded until-operator is one of the key ingredients of this logic. The verification of time- and cost-bounded reachability properties boils down to the compu-

tation of transient rewards rates in ICMRM. This can be done by generalizing the Tijms-Veldman algorithm for homogeneous CMRMs in the following way [34]. The recursive equation now becomes for $i \geq 0$:

$$\begin{aligned} p'_i(s,k{+}1) = \quad & p'_{i-r(s,k\cdot d)}(s,k)\cdot(1-E(s,k\cdot d)\cdot d) \\ & + \sum_{s'} p'_{i-r(s',k\cdot d)}(s',k)\cdot\mathbf{R}(s',s,k\cdot d)\cdot d \end{aligned}$$

This equation is simply obtained from the recursive scheme for homogeneous CMRMs by replacing transition rates, exit rates, and rewards by their time-dependent counterparts. As time is discretised, at discrete time $k{+}1$, the transition rate, exit rate and reward rate at the time instant k are relevant. As transition and reward-rates depend on a continuous-time parameter (and not the discretised notion of time), the real time-instant equals $k{\cdot}d$. Note that it is straightforward to simplify the above equation in case just the transition rates are time dependent and the reward rates are not, or vice versa.

1.4 Bisimulation and simulation relations

The behaviour of Markov chains can be compared by means of equivalence and pre-order relations. Based on the concepts of bisimulation and simulation relations for labeled transitions systems (see e.g., [50]), probabilistic variants thereof have been defined for Markov chains. Three of these notions are treated in more detail in this section, as well as their relationship to the logics CSL and PCTL. For the sake of simplicity, rewards are not considered. The results and definitions in this section can, however, be easily extended towards Markov chains with state rewards.

1.4.1 Strong bisimulation

One of the most elementary equivalence relations on discrete-time probabilistic systems is probabilistic bisimulation [49]. This variant of strong bisimulation considers two states to be equivalent if the cumulative probability to move to any of the equivalence classes that this relation induces is the same. We consider a slight variant of the original notion in which we require in addition that equivalent states are equally labeled. This is exploited later to establish logical characterizations. For $C \subseteq S$, $\mathbf{P}(s,C) = \sum_{s'\in C} \mathbf{P}(s,s')$ denotes the probability for s to move to a state in C. Let $\mathscr{D} = (S,\mathbf{P},L)$ be a DTMC and R an equivalence relation on S. The quotient of S under R is denoted S/R. R is a *strong bisimulation* on $\mathscr{D}$ if for $s_1 R s_2$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \mathbf{P}(s_1,C) = \mathbf{P}(s_2,C) \text{ for all } C \text{ in } S/R.$$

s_1 and s_2 in $\mathscr{D}$ are strongly bisimilar, denoted $s_1 \sim_d s_2$, if there exists a strong bisimulation R on $\mathscr{D}$ with $s_1 R s_2$.

Strong bisimulation [17, 36] for CTMCs, also known as ordinary lumpability, is a mild variant of the notion for the discrete-time probabilistic setting where it is required that the cumulative rate (instead of the discrete probability) for two equivalent states to move to any of the induced equivalence classes is equal. Let $\mathscr{C} = (S, \mathbf{R}, L)$ be a CTMC and R an equivalence relation on S. As in the discrete case, for $C \subseteq S$, $\mathbf{R}(s,C) = \sum_{s' \in C} \mathbf{R}(s,s')$ denotes the rate of moving from state s to a state in C via a single transition. Note that $E(s) = \mathbf{R}(s,S)$. R is a *strong bisimulation* on $\mathscr{C}$ if for $s_1 R s_2$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \mathbf{R}(s_1,C) = \mathbf{R}(s_2,C) \text{ for all } C \text{ in } S/R.$$

s_1 and s_2 in $\mathscr{C}$ are strongly bisimilar, denoted $s_1 \sim_c s_2$, if there exists a strong bisimulation R on $\mathscr{C}$ with $s_1 R s_2$.

Concerning the relationship between the strong bisimulation notions on discrete-time and continuous-time Markov chains it holds:

$$s \sim_c s' \text{ in CTMC } \mathscr{C} \text{ implies } s \sim_d s' \text{ in DTMC } \mathit{emb}(\mathscr{C}) \quad .$$

The reverse does not hold in general, but holds if all states in $\mathscr{C}$ have identical exit rates. A similar strong relationship holds between bisimulation on CTMCs and on their uniformised DTMCs:

$$s \sim_c s' \text{ in CTMC } \mathscr{C} \text{ implies } s \sim_d s' \text{ in DTMC } \mathit{unif}(\mathscr{C}) \quad .$$

1.4.2 Weak bisimulation

Whereas strong bisimulation relates states that mutually mimic all individual steps, *weak bisimulation* only requires this for certain ("observable") transitions and not for other ("silent") transitions. Let $\mathscr{D} = (S, \mathbf{P}, L)$ be a DTMC and $R \subseteq S \times S$ an equivalence relation. Any transition from s to s' (i.e., $\mathbf{P}(s,s') > 0$) where s and s' are R-equivalent is considered an *R-silent* move. Let Silent_R denote the set of states $s \in S$ for which $\mathbf{P}(s,[s]_R) = 1$, i.e., all stochastic states that do not have a successor state outside their R-equivalence class. These states can only perform R-silent moves. Stochastic states outside Silent_R thus may leave their R-equivalence class with positive probability by a single transition. For any state $s \notin \mathit{Silent}_R$, $C \subseteq S$ with $C \cap [s]_R = \emptyset$:

$$\frac{\mathbf{P}(s,C)}{1 - \mathbf{P}(s,[s]_R)}$$

denotes the conditional probability to move from s to some state in C (which is outside $[s]_R$) via a single transition under the condition that from s *no* transition inside $[s]_R$ is taken. Equivalence R on S is a *weak bisimulation* on $\mathscr{D}$ if for all $s_1 R s_2$ all following conditions hold:

1. $L(s_1) = L(s_2)$
2. If $\mathbf{P}(s_i,[s_i]_R) < 1$ for i=1,2 then for all $C \in S/R$, $C \neq [s_1]_R = [s_2]_R$:

$$\frac{\mathbf{P}(s_1,C)}{1 - \mathbf{P}(s_1,[s_1]_R)} = \frac{\mathbf{P}(s_2,C)}{1 - \mathbf{P}(s_2,[s_2]_R)}$$

3. s_1 can reach a state outside $[s_1]_R$ iff s_2 can reach a state outside $[s_2]_R$.

s_1 and s_2 in $\mathscr{D}$ are weakly bisimilar, denoted $s_1 \approx_d s_2$, iff there exists a weak bisimulation R on $\mathscr{D}$ such that $s_1 R s_2$.

Weakly bisimilar states are equally labeled and their conditional probability to move to another equivalence class (given that they do not stay in their own equivalence class) coincides. Furthermore, by the third condition, for any R-equivalence class C, either all states in C are R-silent (i.e., $\mathbf{P}(s,C) = 1$ for all $s \in C$) or for all $s \in C$ there is a sequence of states $s = s_0, s_1, \ldots, s_n$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ that ends in an equivalence class that differs from C (i.e., $s_n \notin C$).

The intuition behind weak bisimulation on CTMCs is that the time-abstract behaviour of equivalent states is weakly bisimilar (in the sense of the first two conditions of $\approx_d$), and that the "relative speed" of these states to move to another equivalence class is equal. The following result shows that this formulation can be simplified considerably. Let $\mathscr{C} = (S, \mathbf{R}, L)$ be a CTMC and R an equivalence relation on S with $s_1 R s_2$. The following statements are equivalent:

1. If $s_1, s_2 \notin \mathit{Silent}_R$ then for all $C \in S/R$, $C \neq [s_1]_R = [s_2]_R$:

$$\frac{\mathbf{P}(s_1,C)}{1-\mathbf{P}(s_1,[s_1]_R)} = \frac{\mathbf{P}(s_2,C)}{1-\mathbf{P}(s_2,[s_2]_R)} \quad \text{and} \quad \mathbf{R}(s_1, S \setminus [s_1]_R) = \mathbf{R}(s_2, S \setminus [s_2]_R)$$

2. $\mathbf{R}(s_1,C) = \mathbf{R}(s_2,C)$ for all $C \in S/R$ with $C \neq [s_1]_R = [s_2]_R$.

This result justifies the following definition of weak bisimulation on CTMCs [16]. Let $\mathscr{C} = (S, \mathbf{R}, L)$ be a CTMC and R an equivalence relation on S. R is a *weak bisimulation* on $\mathscr{C}$ if for all $s_1 R s_2$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \mathbf{R}(s_1,C) = \mathbf{R}(s_2,C) \text{ for all } C \text{ in } S/R \text{ with } C \neq [s_1]_R.$$

s_1 and s_2 in $\mathscr{C}$ are weakly bisimilar, denoted $s_1 \approx_c s_2$, iff there exists a weak bisimulation R on $\mathscr{C}$ such that $s_1 R s_2$. Evidently, any strongly bisimilar pair of states is also weak bisimilar. The reverse, however, does not hold. Thus:

$$\sim_c \subseteq \approx_c \quad \text{and} \quad \sim_d \subseteq \approx_d \quad .$$

Concerning the relationship between the weak bisimulation notions on discrete-time and continuous-time Markov chains we obtain similar results as for strong bisimulation:

$$s \approx_c s' \text{ in CTMC } \mathscr{C} \text{ implies } s \approx_d s' \text{ in DTMC } \mathit{emb}(\mathscr{C}) \text{ and } s \approx_d s' \text{ in DTMC } \mathit{unif}(\mathscr{C}) \quad .$$

For a CTMC in which all states have the same exit rate, i.e., $E(s)$ equals some constant E for any state s, weak bisimulation $\approx_c$ and strong bisimulation $\sim_c$ coincide.

1.4.3 Strong simulation

Bisimulation relations are equivalences requiring two bisimilar states to exhibit identical stepwise behaviour. On the contrary, simulation relations are preorders on

the state space requiring that whenever $s \prec s'$ ("s' simulates s") state s' can mimic all stepwise behaviour of s; the converse, i.e., $s' \prec s$ is not guaranteed, so state s' may perform steps that cannot be matched by s. Thus, if s' simulates s then every successor of s has a corresponding, i.e., related successor of s', but the reverse does not necessarily hold. The use of simulation relies on the preservation of certain classes of formulas, not of all formulas (such as for $\sim$).

For labeled transition systems, state s' simulates state s if for each successor state t of s there is a one-step successor state t' of s' that simulates t. Simulation of two states is thus defined in terms of simulation of their successor states. (It is therefore sometimes called forward simulation.) In the probabilistic setting, the target of a transition is in fact a probability distribution, and thus, the simulation relation $\prec$ needs to be lifted from states to distributions. In fact, strong bisimulation on FPSs was defined as an equivalence on S such that all R-equivalent states s_1 and s_2 are equally labeled and

$$\mathbf{P}(s_1,\cdot) \;\equiv_R\; \mathbf{P}(s_2,\cdot)$$

where $\equiv_R$ denotes the lifting of R on $\mathit{Distr}(S)$ defined as:

$$\mu \equiv_R \mu' \quad \text{iff} \quad \mu(C) = \mu'(C) \text{ for all } C \in S/R.$$

(It is easy to see that $\equiv_R$ is an equivalence.) The rough idea behind the definition of simulation relations is to replace the equivalence $\equiv_R$ by a non-symmetric relation $\sqsubseteq_R$ which is obtained using the concept of weight functions [41, 42].

Let S be a set, $R \subseteq S \times S$, and $\mu, \mu' \in \mathit{Distr}(S)$. A *weight function* for μ and μ' with respect to R is a function $\Delta : S \times S \to [0,1]$ such that:

1. $\Delta(s,s') > 0$ implies $s\,R\,s'$
2. $\mu(s) = \sum_{s' \in S} \Delta(s,s')$ for any $s \in S$
3. $\mu'(s') = \sum_{s \in S} \Delta(s,s')$ for any $s' \in S$

We write $\mu \sqsubseteq_R \mu'$ (or simply $\sqsubseteq$, if R is clear from the context) iff there exists a weight function for μ and μ' with respect to R. $\sqsubseteq_R$ is the lift of R to distributions.

Intuitively, Δ distributes a probability distribution over a set X to a distribution over a set Y such that the total probability assigned by Δ to $y \in Y$ equals the original probability $\mu'(y)$ on Y. In a similar way, the total probability mass of $x \in X$ that is assigned by Δ must coincide with the probability $\mu(x)$ on X. Δ is a probability distribution on $X \times Y$ such that the probability to select (x,y) with xRy is one. In addition, the probability to select an element in R whose first component is x equals $\mu(x)$, and the probability to select an element in R whose second component is y equals $\mu'(y)$.

In the discrete-time setting, simulating states need to be equally labeled, and a weight function must exist that relates their one-step probabilities [42]. Let $\mathscr{D} = (S, \mathbf{P}, L)$ be a DTMC and $R \subseteq S \times S$. R is a *strong simulation* on $\mathscr{D}$ if for all $s_1\,R\,s_2$:

$$L(s_1) = L(s_2) \quad \text{and} \quad \mathbf{P}(s_1,\cdot) \sqsubseteq_R \mathbf{P}(s_2,\cdot).$$

s_2 strongly simulates s_1 in $\mathscr{D}$, denoted $s_1 \prec_d s_2$, iff there exists a strong simulation R on $\mathscr{D}$ such that $s_1 R s_2$. For any DTMC $\mathscr{D}$ it holds:

$$\sim_d \text{ coincides with } \prec_d \cap \prec_d^{-1}$$

where $\prec_d^{-1}$ denotes the inverse of the relation $\prec_d$, i.e., $s' \prec_d^{-1} s$ if and only if $s \prec_d s'$.

The intention of a simulation preorder on CTMCs is to ensure that state s_2 simulates s_1 if and only if (i) s_2 is "faster than" s_1 and (ii) the time-abstract behavior of s_2 simulates that of s_1. Note that compared to the discrete-time setting, the only extra requirement is the "faster than" constraint, the other constraints are identical. It therefore directly follows that this notion is a pre-order. Let $\mathscr{C} = (S, \mathbf{R}, L)$ be a CTMC and $R \subseteq S \times S$. R is a *strong simulation* on $\mathscr{C}$ if for all $s_1 R s_2$:

$$L(s_1) = L(s_2), \quad \mathbf{P}(s_1, \cdot) \sqsubseteq_R \mathbf{P}(s_2, \cdot) \quad \text{and} \quad E(s_1) \leq E(s_2).$$

s_2 strongly simulates s_1 in $\mathscr{C}$, denoted $s_1 \prec_c s_2$, iff there exists a strong simulation R on $\mathscr{C}$ such that $s_1 R s_2$.

Concerning the relationship between the simulation relations on discrete-time and continuous-time Markov chains it holds:

$$s \prec_c s' \text{ in CTMC } \mathscr{C} \text{ implies } s \prec_d s' \text{ in DTMC } \mathit{emb}(\mathscr{C}) \text{ and } s \prec_d s' \text{ in DTMC } \mathit{unif}(\mathscr{C}) \quad .$$

The reverse does not hold in general, but holds for the particular case in which all states have the same exit rate; these CTMCs are sometimes refered to as *uniform*.

Similar to the notion of weak bisimulation, weak versions of the simulation preorder relations can be defined that only require that s' mimick the visible steps of s (rather than all possible steps). For the definition of weak simulation relations on DTMCs and CTMCs we refer to [12].

1.4.4 Logical characterization

Bisimulation. In both the discrete and the continuous setting, strong bisimulation ($\sim_d$ and $\sim_c$) coincide with logical equivalence (for the logics PCTL and CSL, respectively). The latter are denoted $\equiv_{\text{PCTL}}$ and $\equiv_{\text{CSL}}$, respectively. That is, $s_1 \equiv_{\text{PCTL}} s_2$ if and only if s_1 and s_2 satisfy exactly the same PCTL formulae. Similarly, $s_1 \equiv_{\text{CSL}} s_2$ if and only if s_1 and s_2 satisfy exactly the same CSL formulae.

- For any DTMC [5]: $\sim_d$ coincides with $\equiv_{\text{PCTL}}$.
- For any CTMC [8, 27]: $\sim_c$ coincides with $\equiv_{\text{CSL}}$.

Desharnais *et al.* [27] have shown that $\sim_c$ and $\equiv_{\text{CSL}}$ not only coincide for CTMCs with a countable state space but also for continuous-state processes. These results mean that any two bisimilar Markov chains cannot be distinguished by any PCTL (or CSL)-formula, since they satisfy exactly the same formulae. Using efficient algorithms to construct the quotient space under bisimilarity [25], a Markov chain can be lumped prior to the model checking while preserving the results. (The worst

case time complexity is logarithmic in the number of states and linear in the number of transition probabilisties.) Another consequence of this result is that in order to disprove that two Markov chains are bisimilar, it suffices to provide a single logical formula that holds in one but not in the other chain. The above definitions and results can easily be lifted to models with rewards, by requiring for $s \sim s'$ that $r(s) = r(s')$. Quotienting reward models with respect to bisimulation has the same time complexity as for DTMCs (and CTMCs).

For weak bisimulation we obtain:

- For any DTMC [5]: $\approx_d$ coincides with $\equiv_{\text{PCTL}}$.
- For any CTMC [12]: $\approx_c$ coincides with $\equiv_{\text{CSL}}$.

A few remarks are in order here as it might be surprising that both strong and weak bisimulation—that have distinguishing expressiveness—coincide with logical equivalence. The main reason for this is that in this chapter we consider the next-less fragment of PCTL (and CSL), i.e., we do not consider the next operator. As the next operator refers to the direct successors of a state, and as these states might be abstracted from in weak (but not in strong) bisimulation, the validity of this operator is not preserved under weak bisimulation whereas it is preserved under strong bisimulation. In absence of the next operator, indeed both weak and strong bisimulation cannot be distinguished from the logical perspective.

Simulation. To consider the relation between simulation relations and the logics PCTL and CSL, we consider the so-called safe fragments of these logics. The syntax of safe PCTL is defined by the grammar:

$$\Phi ::= \text{tt} \;\Big|\; a \;\Big|\; \neg a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \Phi \vee \Phi \;\Big|\; \mathscr{P}_{\geq p}(\Phi \;\mathscr{U}^{\leq k}\; \Phi) \;\Big|\; \mathscr{P}_{\geq p}(\Phi \,\mathscr{W}^{\leq k} \Phi)$$

where the weak until operator $\mathscr{W}$ (sometimes referred to as unless) is defined by:

$$\sigma \models \Phi \,\mathscr{W}^{\leq k} \Psi \text{ iff } \sigma \models \Phi \;\mathscr{U}^{\leq k}\; \Psi \text{ or } \sigma[i] \models \Phi \text{ for all } i \leq k$$

In contrast to the (strong) until-operator, the weak until-operator does not require Ψ to become valid. Note that $\Phi \,\mathscr{W}\, \text{ff}$ is identical to $\Box\Phi$. A typical safety property is $\mathscr{P}_{\geq 0.99}(\Box^{\leq 65} \mathit{illegal})$ asserting that with probability at least 0.99 the system will not visit an illegal state for the next 65 time units. It is important to realize that we only consider lowerbounds on probabilities in the $\mathscr{P}$-operator, and no upperbounds. In addition, negations only occur adjacent to atomic propositions. (Dually, a live fragment of the logic could be defined that only allows upperbounds. Then for each formula Φ in the safe fragment, there is a liveness formula that is equivalent to $\neg\Phi$.) The safe fragment of CSL is defined in a similar way. The relation between simulation pre-orders $\preceq_c$ and $\preceq_d$ and the safe fragments of CSL and PCTL, respectively, is as follows. Let $s \leq_{\text{safe PCTL}} s'$ if and only if for any formula Φ in safe PCTL it holds: $s' \models \Phi$ implies $s \models \Phi$. Similarly, the pre-order $\leq_{\text{safe CSL}}$ is defined. Then:

- For any DTMC [26]: $\preceq_d$ coincides with $\leq_{\text{safe PCTL}}$.
- For any CTMC [12]: $\preceq_c$ coincides with $\leq_{\text{safe CSL}}$.

The definitions and results for DTMCs and CTMCs can easily be adapted to reward extensions thereof by requiring for $s \prec s'$ that $r(s) = r(s')$.

Decision algorithms. For the sake of completeness, we briefly summarize the various decision algorithms that exist for the (bi)simulation relations considered here. Checking strong bisimulation on Markov chains can be done in time $\mathcal{O}(N \cdot \log K)$, where N is the number of states and K is the number of transitions [25]. This algorithm can also be employed for $\approx_c$. In the discrete-time case, checking $\sim_d$ takes $\mathcal{O}(K \cdot \log N)$ time [40], whereas $\approx_d$ take $\mathcal{O}(N^3)$ time [9]. The computation of $\prec_d$ can be reduced to a maximum flow problem [6] and has a worst case time complexity of $\mathcal{O}((K \cdot N^6 + K^2 \cdot N^3)/\log N)$. The same technique can be applied for computing $\prec_c$.

1.5 Epilogue

1.5.1 Summary of results

Table 1.1 summarizes the time-complexities for the various probabilistic reachability problems discussed in this survey. Note that the indicated complexity figures refer to solve the reachability problem for *all* states in the Markov model at hand. For the sake of completeness, details about the variants of DTMCs and CTMCs that exhibit nondeterminism are included in the table although these models are not further treated in this chapter. Due to the presence of nondeterminism, the problem in these models is to find the maximal (or, dually, the minimal) probability to reach a given goal (set of) state(s) within a given step- or time-bound. We emphasize that all complexity indications refer to model-checking algorithms that are approximate. Stated differently, these algorithms calculate probabilities up to a certain precision that is fixed a priori by the user, and based on these approximate probabilties decide on the validity of the formula under consideration. Interestingly enough, all the approximate algorithms have a polynomial complexity, in contrast to (exact) model-checking algorithms for timed automata that are exponential in, e.g., the number of clock variables.

Here, N denotes the state space size, i.e., $N = |S|$, k is the step bound (applicable for discrete-time models only), t is its continuous equivalent, r is the upperbound on the cumulative reward, d is the step size (in case of discretization), E is the largest exit rate in a CTMC, and M is the number of distinct actions in a state (relevant for nondeterministic models only).

Software tools. ETMCC was the first CTMC model checker [35]. It uses a sparse-matrix representation, is based on the algorithms explained in this note, and has a simple input format that enables its usage as a back-end to existing performance modeling tools. PRISM [48] is a model checker for (discrete-time and continuous-time) Markov chains as well as Markov decision processes. It also contains means

model class	*reachability problem*	*time complexity*	*note (if applicable)*
the discrete-time setting			
DTMC	step-bounded	$\mathcal{O}(k{\cdot}N^2)$	
MDP	step-bounded	$\mathcal{O}(poly(N,M))$	extremal probability
DMRM	step- and cost-bounded	$\mathcal{O}(k{\cdot}r{\cdot}N^3)$	recursive algorithm
the continuous-time setting			
CTMC	time-bounded	$\mathcal{O}(E{\cdot}t{\cdot}N^2)$	
CTMDP	time-bounded	$\mathcal{O}(E{\cdot}t{\cdot}N^2{\cdot}M)$	uniform exit rate E
CMRM	time- and cost-bounded	$\mathcal{O}(t{\cdot}r{\cdot}N^3{\cdot}d^{-2})$	discretization
ICMRM	time- and cost-bounded	$\mathcal{O}(t{\cdot}r{\cdot}N^3{\cdot}d^{-2})$	discretization

FIGURE 1.3: Overview of time complexities for verifying bounded probabilistic reachability problems

for checking expected cumulated reward properties. It uses a mixed representation: a binary-decision diagram for the probability (or rate) matrix and a sparse representation for the solution vector. This tool has been applied to several case studies from different application fields. This includes biological systems, randomized distributed protocols, and security protocols. MRMC [43] is based on the principles of ETMCC and supports besides CSL also the logic CSRL. This tool contains implementations of the discretization and path graph generation algorithms.

1.5.2 Further research topics

This chapter has surveyed the model-checking approach to discrete and continuous-time Markov (reward) models. We believe that the model-checking approach provides a useful technique for performance and dependability analysis. Logics are useful for specifying performance guarantees, and model-checking algorithms provide effective (and efficient) means for checking these guarantees. This is done in a fully automated way, and provides a *single* framework for checking performance measures as well as functional properties such as absence of deadlocks and reponsiveness. Note that (time-inhomogeneous) continuous-time Markov reward models can be considered as simple stochastic hybrid systems. Further work in this area is needed to consider more expressive models. An interesting topic for future work is to develop model-checking algorithms for (simple variants) of piecewise deterministic Markov processes [24] and to finding logical characterizations for bisimulations on PDMPs. The reader is referred to Chapter **??** in this volume on a compositional specification formalism for PDMPs.

References

[1] A. ACQUAVIVA, A. ALDINI, M. BERNARDO, A. BOGLIOLO, E. BONTA AND E. LATTANZI. Assessing the impact of dynamic power management on the functionality and the performance of battery-powered appliances. In *Dependable Systems and Networks (DSN 2004)*, IEEE CS Press, pp. 731–740, 2004.

[2] S. ANDOVA, H. HERMANNS, J.-P. KATOEN. Discrete-time rewards model-checked. In *Formal Methods for Timed Systems*, LNCS 2791: 88-104, 2003.

[3] K.R. APT, N. FRANCEZ AND W.-P. DE ROEVER. A proof system for communicating sequential processes. *ACM Transactions on Programming Languages and Systems*, **2**:359–385, 1980.

[4] A. AZIZ, K. SANWAL, V. SINGHAL AND R. BRAYTON. Model checking continuous time Markov chains. *ACM Transactions on Computational Logic*, **1**(1): 162–170, 2000.

[5] A. AZIZ, V. SINGHAL, F. BALARIN, R. BRAYTON AND A. SANGIOVANNI-VINCENTELLI. It usually works: the temporal logic of stochastic systems. In P. Wolper, editor, *Computer-Aided Verification*, LNCS 939: 155–165, 1995.

[6] C. BAIER, B. ENGELEN, AND M. MAJSTER-CEDERBAUM. Deciding bisimilarity and similarity for probabilistic processes. *J. of Comp. and System Sc.*, **60**(1):187–231, 2000.

[7] C. BAIER, B. HAVERKORT, H. HERMANNS AND J.-P. KATOEN. On the logical characterisation of performability properties. In: U. Montanari, J.D.P. Rolim, E. Welzl, editors, *Automata, Languages and Programming*, LNCS 1853:780–792, 2000.

[8] C. BAIER, B. HAVERKORT, H. HERMANNS AND J.-P. KATOEN. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, **29**(6): 524–541, 2003.

[9] C. BAIER AND H. HERMANNS. Weak bisimulation for fully probabilistic processes. *Computer-Aided Verification*, LNCS 1254, pp. 119-130, 1997.

[10] C. BAIER, J.-P. KATOEN AND H. HERMANNS. Approximate symbolic model checking of continuous-time Markov chains. In *Concurrency Theory*, LNCS 1664: 146–162, Springer, 1999.

[11] C. BAIER, H. HERMANNS, J.-P. KATOEN, B. HAVERKORT. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science*, vol. 345 no. 1, pages 2–26, 2005.

[12] C. BAIER, J.-P. KATOEN, H. HERMANNS, V. WOLF. Comparative branching-time semantics for Markov chains. *Information & Computation* **200**(2): 149–214, 2005.

[13] C. BAIER, B.R. HAVERKORT, H. HERMANNS AND J.-P. KATOEN. On the logical characterisation of performability properties. In *Automata, Languages, and Programming*, LNCS 1853: 780–792, Springer, 2000.

[14] M.D. BEAUDRY. Performance-related reliability measures for computing systems. *IEEE Trans. on Comp. Sys.*, **27**(6): 540–547, 1978.

[15] D. BERTSEKAS. *Dynamic Programming and Optimal Control, volumes 1 and 2.* Athena Scientific, 1995.

[16] M. BRAVETTI. Revisiting interactive Markov chains. In W. Vogler and K.G. Larsen (eds), *Models for Time-Critical Systems*, BRICS Notes Series NS-02-3, pp. 60–80, 2002.

[17] P. BUCHHOLZ. Exact and ordinary lumpability in finite Markov chains. *Journal of Applied Probability*, **31**: 59–75, 1994.

[18] P. BUCHHOLZ, J.-P. KATOEN, P. KEMPER AND C. TEPPER. Model-checking large structured Markov chains. *Journal of Logic and Algebraic Programming*, **56**(1-2): 69–97, 2003.

[19] E.M. CLARKE AND E.A. EMERSON. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logic of Programs*, LNCS 131: 52–71, 1981.

[20] E.M. CLARKE AND R. KURSHAN. Computer-aided verification. *IEEE Spectrum*, **33**(6):61–67, 1996.

[21] E.M. CLARKE, O. GRUMBERG AND D. PELED. *Model Checking*. MIT Press, 1999.

[22] L. CLOTH, J.-P. KATOEN, M. KHATTRI, R. PULUNGAN. Model checking Markov reward models with impulse rewards. *Dependable Systems and Networks (DSN 2005)*, pp. 722-731, IEEE CS Press, 2005.

[23] C. COURCOUBETIS AND M. YANNAKAKIS. Verifying temporal properties of finite-state probabilistic programs. In *Found. of Comp. Sc. (FOCS)*, pp. 338–345, 1988.

[24] M.H.A. DAVIS. Piecewise deterministic Markov processes: a general class of non-diffusion stochastic models. *J. Royal Statistical Soc. (B)*, **46**:353–388, 1984.

[25] S. DERISAVI, H. HERMANNS AND W.H. SANDERS. Optimal state-space lumping in Markov chains. *Information Processing Letters*, **87**(6):309–315, 2004.

[26] J. DESHARNAIS. Logical characterisation of simulation for Markov chains. *Workshop on Probabilistic Methods in Verification*, Tech. Rep. CSR-99-8, Univ. of Birmingham, pp. 33–48, 1999.

[27] J. DESHARNAIS AND P. PANANGADEN. Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. *Journal of Logic and Algebraic Programming*, **56**(1-2): 99-115, 2003.

[28] D. GROSS AND D.R. MILLER. The randomization technique as a modeling tool and solution procedure for transient Markov chains. *Op. Res.* **32**(2): 343–361, 1984.

[29] H.A. HANSSON AND B. JONSSON. A logic for reasoning about time and reliability. *Formal Aspects of Comp.* **6**(5), (1994) 512–535.

[30] D. HAREL. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, **8**(3):231–274, 1987.

[31] S. HART, M. SHARIR AND A. PNUELI. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, **5**(3): 356–380, 1983.

[32] B. HAVERKORT, L. CLOTH, H. HERMANNS, J.-P. KATOEN AND C. BAIER. Model-checking performability properties. In: *Dependable Systems and Networks*, pp. 103–113, 2002.

[33] B. HAVERKORT, H. HERMANNS, AND J.-P. KATOEN. On the use of model checking techniques for quantitative dependability evaluation. In *IEEE Sym. on Reliable Distributed Systems (SRDS)*, pp. 228–238, IEEE CS Press, 2000.

[34] B. HAVERKORT AND J.-P. KATOEN. The performability distribution for non-homogeneous Markov reward models. In *Performability Workshop*, pp. 32–34, 2005.

[35] H. HERMANNS, J.-P. KATOEN, J. MEYER-KAYSER AND M. SIEGLE. A Markov chain model checker. *J. on Software Tools and Technology Transfer*, **4**(2): 153–172, 2003.

[36] J. HILLSTON. *A Compositional Approach to Performance Modelling*. Cambridge Univ. Press, 1996.

[37] C.A.R. HOARE. An axiomatic basis for computer programming. *Communications of the ACM*, **12**:576–580, 583, 1969.

[38] G.J. HOLZMANN. *Design and Validation of Computer Protocols.* (Prentice-Hall, 1991).

[39] R.A. HOWARD. *Dynamic Probabilistic Systems; Vol. I, II.* John Wiley & Sons, 1971.

[40] T. HYUNH AND L. TIAN. On some equivalence relations for probabilistic processes. *Fundamentae Informatica*, **17**: 211–234, 1992.

[41] C. JONES AND G. PLOTKIN. A probabilistic powerdomain of evaluations. *IEEE Symp. on Logic in Comp. Sc.*, pp. 186–195, 1989.

[42] B. JONSSON AND K.G. LARSEN. Specification and refinement of probabilistic processes. *IEEE Symp. on Logic in Comp. Sc.*, pp. 266-277, 1991.

[43] J.-P. KATOEN, M. KHATTRI AND I.S. ZAPREEV. A Markov reward model checker. In: *Quantitative Evaluation of Systems (QEST)*. IEEE CS Press, 243–245, 2005.

[44] J.-P. KATOEN, M.Z. KWIATKOWSKA, G. NORMAN AND D. PARKER. Faster and symbolic CTMC model checking. In: L. de Alfaro and S. Gilmore, editors, *Process Algebra and Probabilistic Methods*, LNCS 2165:23–38, 2001.

[45] J.-P. KATOEN AND I.S. ZAPREEV. Safe on-the-fly steady-state detection for time-bounded reachability. In: *Quantitative Evaluation of Systems*, IEEE CS Press, 2006 (to appear).

[46] J.G. KEMENY AND J.L. SNELL. *Finite Markov Chains*. Van Nostrand, 1960.

[47] M.Z. KWIATKOWSKA, G. NORMAN AND R. SEGALA. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In: G. Berry et al, editors, *Computer-Aided Verification*, LNCS 2102: 194–206, 2001.

[48] M. KWIATKOWSKA, G. NORMAN AND D. PARKER. Probabilistic symbolic model checking using PRISM: a hybrid approach. *J. on Software Tools for Technology Transfer*, **6**(2): 128–142, 2004.

[49] K.G. LARSEN AND A. SKOU. Bisimulation through probabilistic testing. *Inf. and Comput.*, **94**(1): 1–28, 1991.

[50] R. MILNER. *Communication and Concurrency*. Prentice-Hall, 1989.

[51] S. OWICKI AND D. GRIES. An axiomatic proof technique for parallel programs. *Acta Informatica*, **6**:319–340, 1976.

[52] M.L. PUTERMAN. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[53] J.-P. QUEILLE AND J. SIFAKIS. Specification and verification of concurrent systems in CESAR. In: *Proceedings 5th International Symposium on Programming*, LNCS 137:337–351, 1982.

[54] M.A. QURESHI AND W.H. SANDERS. A new methodology for calculating distributions of reward accumulated during a finite interval. In *Fault-Tolerant Computing Symposium*, IEEE CS Press, pp. 116–125, 1996.

[55] J. STAUNSTRUP, H.R. ANDERSEN, J. LIND-NIELSEN, K.G. LARSEN, G. BEHRMANN, K. KRISTOFFERSEN, H. LEERBERG AND N.B. THEILGAARD. Practical verification of embedded software. *IEEE Computer*, **33**(5):68–75, 2000.

[56] W.J. STEWART. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, 1994.

[57] H.C. TIJMS. *A First Course in Stochastic Models.* John Wiley & Sons, 2003.

[58] H.C. TIJMS AND R. VELDMAN. A fast algorithm for the transient reward distribution in continuous-time Markov chains. *Operations Research Letters*, **26**:155–158, 2000.

[59] A.M. TURING. On checking a large routine. In: *Report of a Conference on High Speed Calculating Machines*, pp. 76–69, 1949.

[60] M.Y. VARDI. Automatic verification of probabilistic concurrent finite state programs. In *IEEE Symposium on Foundations of Computer Science*, pp. 327–338, 1985.

Chapter 2

Stochastic reachability: Theoretical foundations and numerical approximation

Maria Prandini
Politecnico di Milano

2.1 Introduction

2.2 Other stuff

2.3 Conclusions

0-8493-0052-5/00/$0.00+$.50