Perspectives in Probabilistic Verification

Joost-Pieter Katoen
Software Modeling and Verification Group, RWTH Aachen University, Germany
katoen@cs.rwth-aachen.de

Abstract

Soon after the birth of the flourishing research area of model checking in the early eighties, researchers started to apply this technique to finite automata equipped with probabilities. The initial focus was on qualitative properties — e.g., does a program terminate with probability one? — but later efficient algorithms were developed for quantitative questions as well. Model checking of probabilistic models received quite some attention in the late nineties, and this popularity lasts until today. Application areas are, among others, security, distributed algorithms, systems biology, and performance analysis. What is the current state of this field? Probabilistic verification, quo vadis? This paper surveys the main achievements during the last two decades, reports on recent advances, and attempts to point out some research challenges for the coming years.

1. The jungle of probabilistic models

Traditional model checking aims at checking the validity of a temporal logic formula Φ (in LTL, CTL, CTL*, or the like) on a given Kripke structure $\mathcal{M}$, i.e., it checks whether $\mathcal{M} \models \Phi$? Kripke structures are transition systems, where states are labeled with propositions, and the transition relation is total. In the probabilistic setting, however, different models exist, and their appropriateness is mainly determined by the application, e.g., is continuous time needed, is there a need for nondeterminism, and so forth. Sokolova and de Vink gave an excellent survey on probabilistic models [47] and studied their relationship based on bisimulation equivalences.

Discrete time. Randomised distributed algorithms, for instance, are characterised by the fact that random phenomena such as flipping a coin for obtaining an IP address occur in just a small fragment of the algorithm. Most of the behaviour is determined by the independent (and deterministic) evolvement of the individual processes. Modeling this by interleaving, the typical way in which independent concurrent activities are modeled, naturally yields nondeterminism. For these algorithms, models with discrete probabilities and nondeterminism are thus most appropriate — Markov decision processes (or slight variants thereof), MDPs, for short. Put in a nutshell, an MDP is a Kripke structure in which transitions have states as sources and probability distributions over states as targets.

Definition 1 *A (discrete-time)* Markov decision process *(MDP) is a tuple* $(S, \mathit{Act}, \mathbf{P}, L)$ *with* S, *a countable set of* states, Act, *a set of* actions, $\mathbf{P} : S \times \mathit{Act} \times S \to [0,1]$ *such that for each pair* $(s,a) \in S \times \mathit{Act}$ *we have* $\sum_{s' \in S} \mathbf{P}(s,a,s') \in \{0,1\}$. *Finally,* $L : S \to 2^{AP}$ *labels states with sets of atomic propositions.*

A state may have several outgoing transitions; this models (as usual) nondeterminism. The operational character of an MDP is that on reaching a state s, say, nondeterministically one of its outgoing transitions $s \xrightarrow{a}$ is selected. The next state is selected probabilistically, i.e., with probability $\mathbf{P}(s,a,s')$, the successor state is s'.

Due to the presence of nondeterminism, it is not possible to refer to, e.g., the probability to reach a given (set of) goal state(s) as this likelihood depends on the resolution of the nondeterminism in all states on all paths to this goal. One therefore considers lower and upper bounds. Lower bounds are obtained by resolving the nondeterminism in the worst possible way. For reachability objectives, one way view this as attempting to resolve any possible nondeterminism that avoids the MDP of reaching the goal. Conversely, upper bounds are obtained by "helping" the system by resolving the nondeterminism in the best possible way. For reachability this amounts to selecting the best distribution in any state that leads to the goal. The way in which the nondeterminism is resolved is a *policy*. For maximal (and minimal) reachability properties it turns out that deterministic memoryless policies, i.e., policies that deterministically select an outgoing transition solely on the basis of the current state (and not on the staes visited at an earlier stage) suffice, cf. Bianco and de Alfaro [12]. These policies are elegant as their memory consumption is low (only the current state suffices), and they are easy to compute, e.g., by solving a linear program. One of the main aims in model checking MDPs is to find the simplest possible class of policies for a given objective.

In case all states in an MDP just have a single outgoing transition, one obtains a discrete-time Markov chain (DTMC). DTMCs thus form a subclass of MDPs and are appropriate for synchronous distributed algorithms and synchronous hardware circuits where processes progress in a lock-step fashion. They are also used in performance analysis of time-slotted systems, and in earlier days for modeling communication systems based on ATM. In fact, Google's original Pagerank algorithm is based on DTMCs. Due to the absence of nondeterminism, there is no need for policies, and rather than considering bounds, probabilities can be determined. The probability to reach a goal state, e.g., is simply determined by solving a system of linear equations with a variable for each state, cf. Hansson and Jonsson [29].

Continuous time. The discussed models so far are discrete as the notion of time involved is of a discrete nature: each transition represents a single time step. This contrasts with continuous-time models where state residence times are determined by some probability distribution like a normal, uniform, or negative exponential one. Let us focus on continuous-time Markov decision processes (CTMDPs), i.e., MDPs in which rates —parameters of exponential distributions— are associated with actions.

Definition 2 *A CTMDP is a tuple* $(S, \mathit{Act}, \mathbf{P}, L, r)$ *with* $(S, \mathit{Act}, \mathbf{P}, L)$ *an MDP and* $r : S \times \mathit{Act} \to \mathbb{R}$ *such that* $r(s, a) \geq 0$ *for any* $s \in S$ *and* $a \in \mathit{Act}$.

The operational behaviour of a CTMDP is like that of an MDP, except that after selecting a transition $s \xrightarrow{a}$, say, the system resides in state s for a while. The probability that it stays in s for at most t time units is given by $1 - e^{-r(s,a) \cdot t}$ where $r(s, a)$ is the rate associated with the selected transition $s \xrightarrow{a}$. After residing in state s, the next state s' is selected with probability $\mathbf{P}(s, a, s')$. There are thus two random phenomena: the state residence times are governed by a continuous random variable, where the successor states are selected according to a discrete probability distribution (over the states).

As before, due to the presence of nondeterminism —the selection of a transition in a state is made nondeterministically— it does not make much sense to refer to the probability of an event. Instead, worst case and best case resolutions of nondeterminism are considered. Deterministic memoryless policies which only "know" the current state, but not how long the CTMDP resided there, suffice for expected long-run and transient measures. The recent overview Guo, Hernández-Lerma and Prieto-Rumeau [24] surveys such measures for finite and infinite-state CTMDPs. As reachability objectives (what is the maximal probability to eventually reach a given goal state?) do not refer to timing aspects they can be treated as for MDPs. That is, again memoryless policies are optimal. This does not hold for time-bounded reachability objectives —what is the maximal probability to reach a goal state within a deadline?— as the optimal choice in a state may be infuenced by the remaining time to reach the goal. For the simple class of CTMDPs in which $r(s, a) = R$ for all pairs (s, a), it can be shown that policies that base their decision on the number of transitions taken to reach the current state are optimal (among all policies that are time-abstract). Intuitively, this comes from the fact that the expected duration of any transition is equal, namely $\frac{1}{R}$ time units, and thus having knowledge about the number of transitions that have been taken implicitly provides information about the (expected) remaining time until reaching the target. This information may influence the decisions to be taken by a policy. These results together with a polynomial-time algorithm to determine an ε-optimal policy originate by Baier et al. [9].

CTMDPs are of importance for controlling queueing systems, for studying epidemic diseases, and manufacturing control. A well-known application is the scheduling of tasks with a random (exponential) duration on a set of identical machines. The selection of tasks takes place nondeterministically whereas the completion of tasks is random. In the field of computer science, CTMDPs are of importance as semantical model for high-level modeling formalisms for randomly timed systems. Prominent examples are stochastic variants of process algebras, statecharts, and Petri nets.

In case all states in a CTMDP just have a single outgoing transition, one obtains a continuous-time Markov chain (CTMC). CTMCs thus form a subclass of CTMDPs and are heavily used in performance analysis —typically the inter-arrival times of jobs and customers is exponentially distributed— as well as in field such as systems biology. Due to the absence of nondeterminism, there is no need for policies, and rather than considering bounds, probabilities can be determined. The probability to reach a goal state within a deadline, e.g., is determined by solving a system of Volterra integral equations with a variable for each state, cf. Baier, Katoen, and Hermanns [10].

Is the restriction to exponential distributions a severe limitation? To a certain extent, yes. But, other distributions can be approximated by series-parallel combinations (such as Cox distributions) of exponential distributions arbitrary closely. In particular, phase-type distributions —specified as absorption times in a CTMC with an absorbing, i.e, deadlock state— are convenient for that purpose. The size of the CTMC depends on the distribution to be approximated, e.g., the approximation of deterministic distributions requires large structures. In addition, exponential distributions are appropriate to model quite a number of real-life phenomena, such as inter-arrival times, duration until catastrophic events, and reactions of enzymes and molecules. A final argument in favour of exponential distributions is that they are the most neutral choice for randomly describing a phenomenon for which only the expected value is known. In terms of information theory, exponential distributions maximize the entropy in case just the mean is known. All these arguments together make that exponential distributions are not so restrictive as may be thought of at first sight.

Other models. A variety of other probabilistic models exist for which model-checking algorithms have been developed. Without being exhaustive, we mention probabilistic variants of timed automata [39], pushdown automata [37], and two-player games. In a sense, the former two are symbolic models for infinite-state MDPs, whereas the last one is a generalization of MDPs in which there are two players and moves are resolved probabilistically (so-called $2\frac{1}{2}$ player games). They generalize MDPs that can be viewed as $1\frac{1}{2}$-player games. Variants of the above discussed models with costs (or: rewards) have also been considered, and recently with various cost variables. Objectives are then typically to maximize the expected accumulated costs, expected long-run costs, and variants thereof with discounting.

2. State of the art

Discrete-time setting. A qualitative property is an event that either holds with probability one or zero. Checking a qualitative property in finite DTMCs can be done by graph algorithms. This does not hold for infinite DTMCs. Reachability probabilities can be computed by a graph analysis, basically determining the states from which the goal is reachable, and solving a linear equation system. Reasoning about probabilities in MDPs requires the concept of policies. Policies resolve the nondeterminism and "reduce" the MDP to a (typically infinite) DTMC. Computing extreme (i.e., minimal or maximal) probabilities for reachability properties relies on graph algorithms and linear programs. The latter can be solved by means of an iterative approximation algorithm called value iteration. This observation is due to Courcoubetis and Yannakakis [16].

Almost surely, the long-run behavior of a finite DTMC ends in a bottom strongly connected component (BSCC), an SCC that cannot be left anymore once entered. The intuition behind this is that the probability to cycle ad infinitum in a finite SCC is zero. Quantitative (and qualitative) properties about the long-run behavior —such as repeated reachability ($\Box\Diamond$), persistence ($\Diamond\Box$), or Boolean combinations thereof— of a finite DTMC $\mathcal{M}$ can be checked by computing the reachability probabilities of accepting BSCCs in $\mathcal{M}$. This is similar for MDPs, as shown by de Alfaro.

Logics. One way to specify properties over DTMCs and MDPs is using LTL, or alternatively ω-regular properties. The quantitative model-checking problem is then to compute the probability for the set of paths in the model that satisfy the property. Computing this probability for an LTL formula φ on a finite Markov chain $\mathcal{M}$ can be reduced to computing the acceptance probability in the product of $\mathcal{M}$ and a deterministic Rabin automaton (DRA) for $\neg\varphi$. By using DRA, it is guaranteed that the resulting structure is again a DTMC. Vardi showed that qualitative LTL-model checking —is the probability zero or one?— for finite DTMCs is PSPACE-complete. Model-checking a finite MDP against an ω-regular property can be solved in an analogous way. Courcoubetis and Yannakakis [16] showed that qualitative LTL-model checking for finite MDPs has a double exponential lower bound.

Probabilistic Computation Tree Logic (PCTL) is a quantitative variant of CTL where the path quantifiers $\exists$ and $\forall$ are replaced by a probabilistic operator $\mathbb{P}_J(\varphi)$ that specifies lower and/or upper probability bounds (given by J) for the event φ. Hansson an Jonsson showed that PCTL model checking for finite DTMCs relies on the standard CTL model-checking procedure in combination with methods for computing reachability probabilities [29]. When interpreting PCTL on MDPs, the formula $\mathbb{P}_J(\varphi)$ ranges over all policies. The PCTL model-checking problem for MDPs is reducible to the reachability problem, as shown by Bianco and de Alfaro [12].

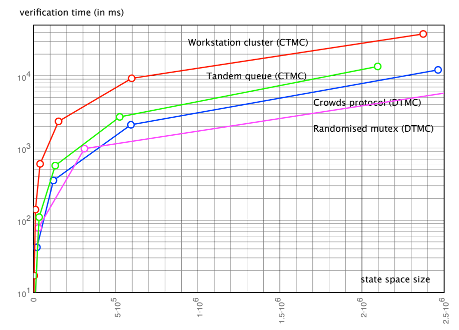
The qualitative fragment of PCTL is obtained by only allowing bounds > 0 and $=1$. For finite DTMCs, CTL is at least as expressive as the qualitative fragment of PCTL. For infinite Markov chains, the expressivity of the qualitative fragment of PCTL and CTL is incomparable. As opposed to CTL, persistence properties can be expressed in PCTL (both qualitative and quantitative). These results are reported in detail in the recent monograph by Baier and Katoen [11]. Variants of PCTL such as PCTL* (where φ can be any LTL-formula), PCTL$^\omega$ [13] and PCTL with regular expressions exist but are less common.

Continuous-time setting. Qualitative properties and quantitative properties that do not refer to timing aspects can be treated as for discrete-time models. The crucial issue for CTMDPs and CTMCs is to determine (extreme) time-bounded reachability probabilities, and long-run measures. Whereas for CTMCs these issues have been resolved this is not the case for CTMDPs. Long-run measures can be determined as for finite DTMCs by considering the reachability probabilities for BSCCs together with determining the equilibrium probabilities in such BSCC. Both steps boil down to solving a system of linear equations. As stated before, time-bounded reachability probabilities are determined by solving a Volterra integral equation system. Suppose that we are interested in reaching a *goal* state within t time units starting from state s. This amounts to determine $Pr(s, \Diamond^{\leq t}\, good)$, i.e., the probability of all paths starting in s leading to a *good* state within t time units. Baier, Katoen, and Hermanns [10] showed that this probability is the least solution of

- 1 if $good \in L(s)$
- $\int_0^t \sum_{s' \in S} \mathbf{P}(s, s', x) \cdot Pr(s', \Diamond^{\leq t-x}\, good)\; dx$ otherwise,

where $\mathbf{P}(s, s', x)$ is a shorthand for the probability to move from s to s' after x time units residing in s. It turns out that there is no need to compute the solution of this equation system by integration methods (such as Runge-Kutta), but a reduction to a well-studied problem in CTMCs allows efficient and numerically stable algorithm, see Baier et al. [8]. Time-bounded reachability probabilities are at the heart of model checking the logic CSL (Continuous Stochastic Logic), a logic that has originally proposed by Aziz et al. [4]. Extensions to CSL have been defined with costs, with time-bounds that are specified by single-clock deterministic timed automata [23], and regular expressions [5]. The latter two involve a product construction similar to that for DTMCs and DRAs described above for LTL verification.

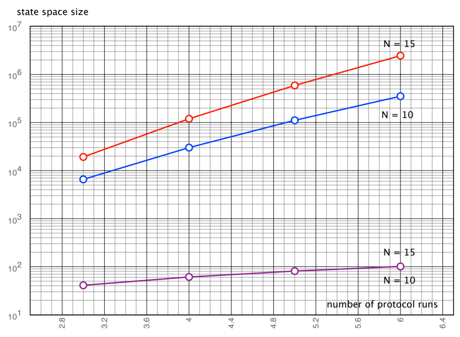
Tools. The development of efficient probabilistic model checking tools at the beginning of this century has resulted in a growing interest of the scientific community in probabilistic verification. In particular the model checker PRISM [42], developed in the group of Kwiatkowska in Oxford, has been applied by a large number of users from different areas. The main success of this tool is a combination of compact state-space representation techniques using a mixture of binary decision diagrams and explicit state techniques, a high-level modeling formalism, together with an easy-to-use and fancy graphical user interface. PRISM supports model checking of MDPs and CTMCs for the logics CSL and PCTL and provides means for checking expected costs. The first model checker for CTMCs [30] has been succeeded by MRMC (Markov Reward Model Checker) [34]. In contrast to PRISM, this tool uses a sparse matrix representation (a slight variant of the compressed rom, compressed column-format), and has a simple input format that is aimed for the usage as a back-end to existing performance modeling tools. Tools such as PRISM, GreatSPN, PEPA Workbench, and Statemate provide means to use MRMC as a back-end tool. MRMC supports DTMCs and CTMCs, bisimulation minimization, and means for computing time- and cost-bounded reachability properties. As indicated by the following plot, the verification of time-bounded reachability probabilities in DTMCs and CTMCs is a matter of a few seconds even for millions of states:



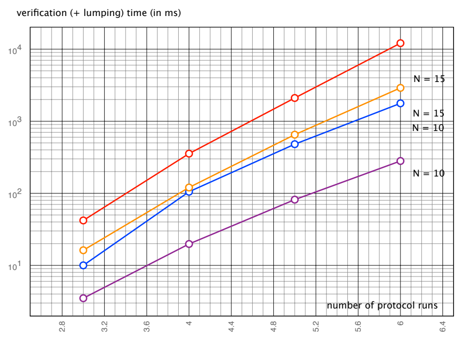
Other probabilistic model checkers are FHP-Murφ, a variant of the model checker Murφ that is tailored to bounded liveness properties for DTMCs, ProbVerus, a BDD-based model checker for DTMCs, LiQuor, APMC, VESTA and YMER. LiQuor is an LTL model checker for MDPs and supports partial-order reduction [6]. Its modeling language is a variant of PROMELA, the input language of the model checker SPIN, with support for random choices, and lossy communication channels. The tools APMC, VESTA and YMER are based on simulation approaches such as Monte Carlo simulation and statistical hypothesis testing. Zapreev recently investigated the use of (traditional) discrete-event simulation techniques to CTMCs model checking and reports promising results [49]. An experimental comparison of five probabilistic model checkers has been recently published [32].

3. Recent Advances and Challenges

Abstraction. Like in the traditional setting, probabilistic model checking suffers from the state space explosion problem: the number of states grows exponentially in the number of system components and cardinality of data domains. To combat this problem, various abstraction techniques have been recently investigated. Abstraction amounts to obtain smaller models by collapsing sets of concrete states to abstract states. The various abstraction techniques basically differ in the way in which states are grouped into abstract states (mostly partitions). Abstract interpretation, e.g., allows abstract states to be overlapping, i.e., concrete states may be mapped onto several abstract states. This principle has been applied to MDPs by Monniaux. Most abstraction techniques are based on a partitioning of the concrete state space. Example techniques in which the partitioning is done in an automated manner are symmetry reduction [40], bisimulation minimization [33], and partial-order reduction [6]. A probabilistic bisimulation relation only relates states that are equally labeled (with propositions) and whose cumulative probability to move to any of the equivalence classes coincides. Experimental studies with the former two techniques show promising results, both in terms of state space reductions (up to exponential savings), and in terms of verification times —minimization+verification of the abstract model mostly is much faster than verifying the concrete model. To illustrate this, the state space reductions for bisimulation minimization for the Crowds protocol are indicated in:



where the quotient for N=15 and N=10 coincides. Here, N indicates the number of crowd members and R (on the x-axis) the number of rounds in the protocol. The timing reduction for checking a liveness property is indicated by:



The appealing feature of symmetry reduction and bisimulation minimization is that the obtained abstract model is bisimilar to the concrete models, and as bisimulation implies logical equivalence (for DTMCs and CTMCs the converse also holds), any verification result on the abstract model carries over to the concrete models. As symmetry reduction is cheap and can performed by a static analysis, and bisimulation minimization reduces substantially more but does not run on-the-fly, a combination seems very fruitful. There is, however, a need for more aggressive abstraction techniques. Recent techniques that have been proposed include abstraction of MDPs by two-player stochastic games [41], magnifying-lens abstraction [19], and predicate abstraction [48]. These approaches are typically conservative (except for the game setting) in the sense that affirmative verification results for abstract models carry over to concrete models. That is to say, if the abstract model satisfies a formula, the concrete one does so too. This does not apply to negative verification results, as false negatives may occur due to over-approximation.

Fecher, Leucker, and Wolf [22] take a different approach and consider a three-valued semantics, i.e., an interpretation in which a formula evaluates to either true, false or indefinite. In this setting, abstraction is conservative for both positive and negative verification results. Only if the verification of the abstract model yields an indefinite answer (dont know), the validity in the concrete model is unknown. This concept can be lifted to DTMCs in a rather natural way by replacing transition probabilities by intervals. Lower and upper bounds of intervals now act as under- and over-approximation, respectively. In fact, the resulting abstract model is of interest on its own since often only bounds on probabilities are known rather than precise values. This framework has been recently extended towards CTMCs by Katoen et al. [35].

Although there are various abstraction techniques around, it is unclear which techniques will prevail in practice. This requires a substantial experimental investigation. In addition, various essential issues have not been tackled satisfactorily. First and foremost, obtaining an initial partitioning is an underdeveloped issue. Predicate abstraction by Wachter, Zhang and Hermanns [48], however, seems promising. Then there is the issue of refinement, i.e., how to refine our abstract model is too coarse? For instance, in the three-valued setting this occurs if the verifying the abstract model yields don't know. Maybe counterexamples (see below) can be used to yield a probabilistic variant of the counterexample-guided abstraction-refinement (CEGAR) framework [14]. This requires mechanisms to check whether a counterexample on an abstract model is spurious, i.e., a false alarm, or not, and techniques that indicate which states (and how) need to be refined. We also need better ways to judge the adequacy of a given abstraction, i.e., how good is an abstraction. A real-valued interpretation as proposed by de Alfaro et al. [20] is expected to be useful to that purpose.

Counterexamples. The power of traditional model checking is not exhaustive verification but rather its capability to generate useful diagnostic feedback in case a violation of the property is encountered. Due to this feature, model checking is seen as an effective and powerful bug-hunting technique: it does not only indicate that a property is refuted, but also indicates why. In contrast to traditional model-checking techniques, the support for diagnostic feedback in case a property is violated in probabilistic verification is rather limited; e.g., when the probability to reach a set of goal states (via legal paths) within 1,000 steps, say, exceeds the required threshold "at most 0.87", typically the feedback is just a list of states for which this is true, possibly accompanied with a curve showing the probability vs the number of steps.

One of the main reasons of this restricted form of feedback has been the absence of a clear notion of a counterexample. Whereas it is clear that in case of a traditional safety property (e.g., always $x > 0$ for variable x), a single finite path that ends in a state where $x \leq 0$ suffices, this is no longer true for reachability probabilities in DTMCs. In fact, to show that the probability to reach a goal state exceeds 0.87, a *set* of paths is needed that all end in a goal state and whose total probability mass is larger than 0.87. In line with shortest counterexamples in classical model checking, preferably this set is as small as possible. Han and Katoen [25] have shown that the computation of the smallest set of paths that can act as a counterexample can be carried out using (small amendments of) k-shortest path algorithms, i.e., algorithms that compute k paths consisting of the shortest path, the one-but-shortest path, and so forth. To be more precise, the shortest-path characterization in [25] yields a minimal counterexample for which there does not exist another equally-sized counterexample with higher probability mass. These results have been recently generalized to CTMC counterexamples [26], and have been adopted to steer the refinement phase in a CEGAR framework for MDPs [31], as well as for counterexample generation for cpCTL, a variant of PCTL with means to reason about conditional probabilities [3]. An alternative approach by Aljazzar and Leue [2] characterizes counterexamples by rooted digraphs, basically fragments of Markov chains where all paths from root to a leaf reach a goal state. Heuristic search algorithms are employed to generate counterexamples, see [1][2]. Initial experiments fo generating these so-called smallest counterexamples indicate that the size of counterexamples may be excessive and succinct representations may be obtained by exploiting regular expressions, see for details [27].

It is fair to say that the work on counterexamples is in a rather initial phase. Experiments need to be conducted to see what the relevance of counterexamples in practice could be and whether the current notions are the right ones. Other open issues are, among others, to investigate whether variants of tree-like counterexamples for CTL [15] could be appropriate, to develop effective means for checking whether a counterexample is spurious or not —a key step in the CEGAR framework— and notions for other models such as PTA, (CT)MDPs, and the like.

Towards stochastic hybrid systems. Probabilistic verification is currently basically aimed at models in which the dynamics is assumed to be *time-homogeneous*. That is to say, the probabilistic nature of mode (or: state) transitions as well as the time-driven behavior are independent of the global time. This is, however, a serious drawback to adequately model random phenomena that occur in practice such as failure rates of hardware components (that typically exhibit a bath-tub curve), software reliability (which reduces due to memory leaks and increases after a restart), and battery depletion (where the power extraction rate non-linearly de- pends on the remaining amount of energy), to mention a few.

To be able to capture the time-driven random behaviour in such applications, it is natural to consider Markov chains (or decision processes) that are *inhomogeneous*. Roughly speaking, this means that rates and transition probabilities are functions of time, i.e., $r(s, a, t)$ denotes the rate of transition $s \xrightarrow{a}$ at time t and the next state s' is selected with probability $\mathbf{P}(s, a, s')(t)$. Time-inhomogeneous CTMCs (ICTMCs, for short) is a very versatile class of models and is a natural stepping-stone towards more full-fledged stochastic hybrid system models such as piecewise deterministic Markov processes (PDPs), a more general class of continuous-time stochastic discrete-event dynamic systems proposed by Davis [17]. The probabilistic nature of mode transitions in PDPs is as for ICTMCs; in fact, ICTMCs are a subclass of PDPs where the global time t has a clock dynamics, i.e., $\dot{t} = 1$.

As a generalization of results on ordinary lumpability on Markov chains [3], it has been recently shown by Han, Katoen, and Mereacre [28] that strong bisimulation preserves transient and long-run state probabilities in ICTMCs. This allows to minimize symbolically ICTMCs prior to their analysis. A quotienting algorithm for a rich class of ICTMDPs —the nondeterministic variant of ICTMCs— is provided as well. The rate functions in this class are either piecewise uniform, i.e., rate $\mathbf{R}_k(t)$ on piece k is of the form $f_k(t) \cdot \mathbf{R}$ for integrable function $\mathbf{R}$, or polynomial, or piecewise polynomial (where each polynomial is of degree three). The worst-case time and space complexity in $\mathcal{O}(ma \log n + M \cdot mr \cdot \log n)$ and $\mathcal{O}(ma + mr)$, respectively, where $M+1$ is the number of pieces (or degrees of the polynomial), ma is the number of action-labeled transitions and m the number of rate-labeled transitions.

Logics and model-checking procedures for this class of models have received scant attention so far. For piecewise constant rate functions, Katoen and Mereacre [36] considered the model-checking problem for a stochastic variant of Hennessy-Milner logic on ICTMCs. The main ingredient of this logic is the modal operator $\langle \Phi \rangle^{I}_{\unlhd p}$ where Φ is a formula, I a time interval, p a probability and $\unlhd$ a binary comparison operator. Intuitively, a state s at time t satisfies $\langle \Phi \rangle^{I}_{>p}$ if the probability to jump to a Φ-state (in a single step) in the interval I exceeds p. Model checking this logic is reducible to finding the zeros of an exponential polynomial. Using Sturm sequences and Newton's method, an approximative model-checking algorithm can be obtained which is linear in the size of the ICTMC, logarithmic in the number of bits precision, and exponential in the nesting depth of the formula.

Treatments of logics with modal operators that are characterized as fixpoints, are however, still out of reach. ICTMCs are also still a relatively simple class of stochastic hybrid systems and much needs to be done to consider more realistic models in which the continuous dynamics are described by differential equations. Aggresive abstraction techniques, as well as approximate techniques need to be developed to treat these models. The field of stochastic hybrid systems provides an enormous potential for probabilistic verification techniques!

Other topics. In the previous paragraphs three topics have been described were recently advances have been made and were further research is needed. There are various other aspects that need further investigation too. In the remainder of this note, I'd like to discuss some of them (without the intention to be complete).

- *Mobility.* Global computing (or: network-aware) computing are highly dynamic and have to deal with frequent changes of the network environment. Distribution awareness and code mobility play a prominent role in the global computing paradigm. Due to their enormous size —networks typically consists of thousands or even millions of nodes— and their strong reliance on mobility and interaction, performance issues are highly relevant. Spontaneous network node crashes and spurious hick ups are random phenomena that occur. Whereas current probabilistic verification techniques are focused on fixed and static topologies, global computing urges the investigation of dynamic topologies. This entails models in which links dynamically change, and logics in which the dynamics can be addressed. Initial attempts have been made to make such extensions for KLAIM [21] and for the π-calculus [44].

- *Synthesis.* Whereas the current focus is on a posteriori verification, there is a clear need to consider the synthesis problem: given a (CT)MDP and a reachability objective, it is possible to effectively synthesize a controller. This amounts to investigating whether such problems are decidable and if so, to come up with algorithms to construct such controllers. For MDPs and PCTL extended with long-run averages, Kucera and Strazowsky showed decidability for the synthesis problem and provided an algorithm [38]. Baier et al. [7] show the NP-hardness for a similar logic for a model in which some (but not all) nondeterminism is controllable.

- *Robustness.* In the current setting, a given probabilistic model is verified. It is assumed that the random information in these models is exact, which in practice, however, is rarely the case. In most circumstances, imprecise indications about random behaviour are available, and the question arises what happens to the satisfaction of a given property when the probabilities in the considered model slightly deviate from their original values. Does the system then still satisfy the property, or not? How large can a maximal deviation be, such that satisfaction is still guaranteed? Initial attenpts towards parametric probabilistic verification of DTMCs have recently been made by, e.g., Daws [18] and Lanotte et al. [43] but suffer from restrictions. It would be of interest to investigate the influence of robustness in the probabilistic setting. Work in the area of robustness of timed automata could be helpful here.

- *Infinite-state models.* These models occur in e.g., communication networks with infinite-sized buffers, or in recursive programs where due to the (possibly mutual) recursive invocations of processes the state space may grow without bound. Advances in this field have recently been made by investigating CTMCs with a certain regular structure [45], probabilistic lossy channel systems [46], pushdown automata [37], and probabilistic timed automata [39]. It would be interesting to investigate whether the framework of regular model checking, for instance, can be carried over to the probabilistic setting (if at all).

4. Epilogue

Probabilistic verification is a rapidly evolving research field, with a potential for applications from a large number of diverse areas. Various research topics in this field require further investigation, both on the theoretical side —some of which have been described in this note— but certainly also on the practical side. In particular, a lot of effort has to be made to bridge the gap towards design engineer notations such as UML (e.g., statecharts and MSCs) and more recent developments such as AADL. Case studies need to be carried out to show the benefits of probabilistic verification over techniques such as simulation that are heavily used in practice. The usage of probabilistic verification by software and hardware developers (at some point in the future) is of utmost importance for the relevance of this challenging and interesting field!

Acknowledgment. All co-workers that have contributed to the reported work are kindfully acknowledged. My research is supported by the EU FP7 QUASIMODO project, the NWO projects MC=MC and QUPES, as well as the DFG research training group ALGOSYN.

References

[1] H. Aljazzar, H. Hermanns and S. Leue. Counterexamples for timed probabilistic reachability. *FORMATS*, LNCS 3829: 177-195, (2005).

[2] H. Aljazzar and S. Leue. Extended directed search for probabilistic timed reachability. *FORMATS*, LNCS 4202: 33-51, (2006).

[3] M. Andres and P. van Rossum: Conditional probabilities over probabilistic and nondeterministic systems. *TACAS* (2008).

[4] A. Aziz, K. Sanwal, V. Singhal and R.K. Brayton. Model-checking continous-time Markov chains. *ACM Trans. Comput. Log.* 1(1): 162-170 (2000).

[5] Baier, C. and Cloth, L. and Haverkort, B.R.H.M. and Kuntz, G.W.M. and Siegle, M.: Model checking Markov chains with actions and state labels. *IEEE TSE*, 33(4):209-224 (2007).

[6] C. Baier and M. Größer and F. Ciesinski: Partial order reduction for probabilistic systems. In *QEST*, 230–239 (2004).

[7] C. Baier, M. Größer, M. Leucker, B. Bollig, F. Ciesinski: Controller synthesis for probabilistic systems. *IFIP TCS*, 493-506 (2004).

[8] C. Baier, B.R. Haverkort, H. Hermanns and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE TSE*, 29(6): 524-541 (2003).

[9] C. Baier, H. Hermanns, J.-P. Katoen and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *TCS*, 345(1): 2-26 (2005).

[10] C. Baier, J.-P. Katoen and H. Hermanns. Approximate symbolic model checking of continuous-time Markov chains. *CONCUR*, LNCS 1664:146-161 (1999).

[11] C. Baier and J.-P. Katoen: *Principles of Model Checking*. MIT Press, 2008.

[12] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. *FSTTCS*, LNCS 1026: 499-513, 1995.

[13] K. Chatterjee, K. Sen and T. Henzinger: Model-checking ω-regular properties of interval Markov chains. *FOSSACS*, (2008).

[14] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith: Counterexample-guided abstraction refinement. *CAV*, LNCS 1855: 154-169, 2000.

[15] E.M. Clarke, S. Jha, Y. Lu and H. Veith. Tree-like counterexamples in model checking. *LICS*: 19-29 (2002).

[16] C. Courcoubetis, M. Yannakakis: The complexity of probabilistic verification. *J. ACM*, 42(4): 857-907 (1995).

[17] Davis, M. H. A.: *Markov Models and Optimization*. Chapman and Hall, 1993.

[18] C. Daws: Symbolic and parametric model checking of discrete-time Markov chains. *ICTAC*, LNCS 3407: 280-294 (2004).

[19] L. de Alfaro, P. Roy: Magnifying-lens abstraction for Markov decision processes. *CAV*, LNCS 4590: 325-338 (2007).

[20] L. de Alfaro, M. Faella, T. Henzinger, R. Majumdar, M. Stoelinga: Model checking discounted temporal properties. *TCS*, 345(1): 139-170 (2005).

[21] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. Model checking mobile stochastic logic. *TCS*, 382:4270, (2007).

[22] H. Fecher, M. Leucker, V. Wolf: Don't know in probabilistic systems. *SPIN*, LNCS 3925:71-88 (2006).

[23] S. Donatelli, S. Haddad and J. Sproston. CSL^{TA}: an expressive logic for continuous-time Markov chains. *QEST*, 31–41 (2007).

[24] X.P. Guo, O. Hernández-Lerma and T. Prieto-Rumeau. A survey of recent results on continuous-time Markov decision processes. *TOP*, 14(2): 177–257, (2006).

[25] T. Han and J.-P. Katoen. Counterexamples in probabilistic model checking. *TACAS*, LNCS 4424:72–86, (2007).

[26] T. Han and J.-P. Katoen: Providing evidence of likely being on time: Counterexample generation for CTMC model checking. *ATVA*, LNCS 4762: 331-346 (2007).

[27] T. Han, J.-P. Katoen, B. Damman. Regular expressions for PCTL counterxamples. Submitted, (2008).

[28] T. Han, J.-P. Katoen and A. Mereacre. Compositional modeling and minimization of time-inhomogeneous Markov chains. *HSCC*, LNCS 4981:244–258, (2008).

[29] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.* 6(5): 512-535 (1994).

[30] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, M. Siegle: A Markov chain model checker. *TACAS*, LNCS 1785: 347-362 (2000).

[31] H. Hermanns, B. Wachter and L. Zhang. On probabilistic CEGAR. AVACS Tech. Rep. 33, (2007).

[32] D.N. Jansen, J.P. Katoen, M. Oldenkamp, M. Stoelinga, and I.S. Zapreev: How fast and fat is your probabilistic model checker? An experimental comparison. *HVC*, LNCS 4899:69–85 (2008).

[33] J.-P. Katoen, T. Kemna, I.S. Zapreev, and D.N. Jansen: Bisimulation minimisation mostly speeds up probabilistic model checking. *TACAS*, LNCS 4424:87-101 (2007).

[34] J.-P. Katoen, M. Khattri and I.S. Zapreev: A Markov reward model checker. *QEST*, 243-244 (2005).

[35] J.-P. Katoen, D. Klink, M. Leucker, V. Wolf: Three-valued abstraction for continuous-time Markov chains. *CAV*, LNCS 4590:311-324 (2007).

[36] J.-P. Katoen, A. Mereacre: Verifying inhomogeneous Markov chains. Submitted, (2008).

[37] A. Kucera, J. Esparza, R. Mayr: Model checking probabilistic push-down automata. *LMCS* 2(1): (2006).

[38] A. Kucera, O. Strazovsky: On the controller synthesis for finite-state Markov decision processes. *Fund. Inf.*, 82:141–153 (2008).

[39] M.Z. Kwiatkowska, G. Norman, R. Segala, J. Sproston: Automatic verification of real-time systems with discrete probability distributions. Theor. Comput. Sci. 282(1): 101-150 (2002).

[40] M.Z. Kwiatkowska, G. Norman, D. Parker: Symmetry reduction for probabilistic model checking. CAV, LNCS 4144: 234-248 (2006)

[41] M.Z. Kwiatkowska, G. Norman, D. Parker: Game-based abstraction for Markov decision processes. QEST, 157-166 (2006).

[42] M.Z. Kwiatkowska, G. Norman and D. Parker. PRISM 2.0: A tool for probabilistic model checking. QEST, 322-323 (2004).

[43] R. Lanotte, A. Maggiolo-Schettini, A. Troina: Parametric probabilistic transition systems for system design and analysis. *FAC*, 19(1): 93-109 (2007).

[44] G. Norman, C. Palamidessi, D. Parker and P. Wu. Model checking the probabilistic pi-calculus. *QEST*, (2007).

[45] Remke, A., Haverkort, B.R., Cloth, L: CSL model checking algorithms for QBDs. TCS 382(1):24–41 (2007).

[46] Ph. Schnoebelen: The verification of probabilistic lossy channel systems. *VOSS*, LNCS 2925:445-466 (2004).

[47] A. Sokolova, E.P. de Vink: Probabilistic automata: system types, parallel composition and comparison. *VOSS*, LNCS 2925:1–43 (2004).

[48] B. Wachter, L. Zhang and H. Hermanns. Probabilistic model checking modulo theories. QEST, 129–139 (2007).

[49] I.S. Zapreev. Model checking Markov chains: Techniques and tools. PhD thesis, Univ. of Twente (2008).