Multi-Terminal Decision Diagrams: a Data Structure for Numerical Integration*

(extended abstract)

Christel Baier[a], Joost-Pieter Katoen[b] and Holger Hermanns[c]

[a] *Lehrstuhl für Praktische Informatik II, University of Mannheim*
68131 Mannheim, Germany

[b] *Lehrstuhl für Informatik 7, University of Erlangen-Nürnberg*
Martensstraße 3, 91058 Erlangen, Germany

[c] *Systems Validation Centre, University of Twente*
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract. This paper introduces *multi-terminal decision diagrams* (MTDDs), a generalisation of MTBDDs [15], as a data structure for representing real-valued functions whose arguments are boolean or real variables. These new structures are suitable for numerical integration using quadrature formulas, like trapezoidal, Simpson and Romberg integration schemes. As an application example, we show how MTDDs can be used for a symbolic model checker for continuous-time Markov chains that are at the basis of contemporary performance evaluation and reliability analysis methodologies.

1 Introduction

The mechanised verification of a given (usually) finite-state model against a property expressed in some temporal logic is known as *model checking*. One of the major reasons for the success of model checking tools in practice is the efficient way to cope with the state-space explosion problem. A prominent technique is to adopt a compact representation of state spaces using (ordered) *binary decision diagrams*, BDDs for short [11]. In the literature, several variants of BDDs for representing real-valued functions such as multi-terminal BDDs (MTBDDs) [15, 23] (also called algebraic decision diagrams, ADDs [4]), edge-valued BDDs [29], binary moment diagrams (BMDs) [12] or hybrid decision diagrams [16] are proposed and used e.g. for verifying arithmetic circuits [12, 37] or discrete-time Markov chains [24, 6, 26]. This paper follows this line by proposing an alternative variant, referred to as *multi-terminal decision diagrams* (MTDDs). MTDDs are a novel generalisation of MTBDDs. MTBDDs (and MTDDs) allow arbitrary real numbers in the terminal nodes instead of just 0 and 1 (like in BDDs). Whereas MTBDDs are defined on boolean variables, MTDDs allow both boolean and real variables. This generalisation is suitable for *numerical integration* using quadrature formulas like trapezoidal, Simpson and Romberg integration schemes [31].

In the second part of this paper, we explain how the proposed MTDD approach can be applied to the verification of continuous-time Markov chains (CTMCs for short). For this purpose, a branching-time logic called *continuous-time stochastic logic* (**CSL**) is used to express properties over CTMCs. This logic is an extension of the (equally named) logic by Aziz et. al [2] with an operator to reason about steady-state probabilities: e.g. the formula $\mathcal{S}_{\geqslant p}(\Phi)$ asserts that the steady-state probability for a Φ-state is at least p, for $p \in [0,1]$. Apart from the usual path-formulas like next and until, a time-bounded until $\mathcal{U}^{\leqslant t}$, for t a non-negative real, is incorporated, together with standard derivatives, such as a time-bounded eventually $\Diamond^{\leqslant t}$. The usual path quantifiers $\forall$ and $\exists$ are replaced by the probabilistic operator $\mathcal{P}_{\bowtie p}(.)$ for comparison operator $\bowtie$ and $p \in [0,1]$. For instance, $\mathcal{P}_{<0.001}(\Diamond^{\leqslant 4} error)$ asserts that the probability for a system error within 4 time-units is less than 10^{-3}. The model checking problem for **CSL** is known to be decidable [2] (for rational time bounds), but to our knowledge no algorithms have been considered yet to verify CTMCs automatically, let alone symbolically. This paper investigates which numerical methods can be adapted to "model check" **CSL**-formulas over CTMCs as models. We show that next and (unbounded) until-formulas can be treated similarly as in the discrete-time probabilistic setting. Checking steady-state probability-properties reduces to solving a linear equation system combined with standard graph analysis

* The first and second author are sponsored by the DAAD-Project AZ 313-ARC-XII-98/38 on stochastic modelling and verification.

methods, while checking the time-bounded until reduces to solving a (recursive) Volterra integral equation system. These integrals are characterised as least fixed points of appropriate higher-order functions, and can thus be approximated by an iterative MTDD-approach. Although it is difficult to obtain precise estimates for the time complexity of model checking using MTDDs (as with BDDs and MTBDDs), the success of (MT)BDD-based model checkers for large-scale examples (for BDDs [13] and for MTBDDs [24, 26]) provides sufficient evidence to investigate MTDDs for our setting. For instance, [24] reports experimental results of the computation of steady-state probabilities for discrete-time Markov chains of over 10^{27} states.

Organisation of the paper. Section 2 introduces MTDDs and presents several operators (including the integral operator) on them. In Section 3, we show how MTDDs can be used for verifying continuous-time Markov chains. Finally, Section 4 concludes the paper.

The contents of this paper is currently submitted elsewhere. Hence, the paper should not be published in the proceedings of SMC'99.

2 Multi-terminal decision diagrams

BDDs and MTBDDs. While (ordered) binary decision diagrams (BDDs) are data structures for representing boolean functions $f : \{0,1\}^n \to \{0,1\}$, multi-terminal BDDs (MTBDDs [15], also called algebraic decision diagrams [4]) allow terminals to be labelled with values of some domain D (e.g. $\mathbb{R}$), i.e. they represent functions of the type $f : \{0,1\}^n \to D$. The main idea behind the MTBDD representation is the use of acyclic rooted directed graphs for a simplified (more compact) representation of the (binary) decision tree which results from the Shannon expansion:

$$f(b_1,\ldots,b_n) \;=\; (1-b_1)\cdot f(0,b_2,\ldots,b_n) \;+\; b_1\cdot f(1,b_2,\ldots,b_n).$$

It has been shown that MTBDDs can be effectively used for various matrix operations [15, 4, 23] and verifying discrete-time Markov chains [24, 6, 26]. We introduce a generalization of MTBDDs that is focussed on dealing with numerical integration.

MTDDs. *Multi-terminal decision diagrams* (MTDDs) are a variant of MTBDDs that yield a discrete representation of real-valued functions whose arguments are either *boolean variables* or real variables (called *integral variables*, since they represent variables over which numerical integration takes place). For the boolean variables, the aforementioned Shannon expansion is used. For an integral variable x, a finite set $\{x_0,\ldots,x_N\}$ is chosen from the range of x. The function $(\ldots,x,\ldots) \mapsto f(\ldots,x,\ldots)$ is represented by the function values $f(\ldots,x_j,\ldots)$, for $0 \leqslant j \leqslant N$. To accomplish this, we use a representation of f that is based on a discrete fragment of the decision tree where the branches for the integral variables represent the cases where $x \in \{x_0,\ldots,x_N\}$. In the sequel, let Var be a fixed set of boolean and integral variables and $<$ total order on Var such that $\mathsf{s} < \mathsf{x}$ for all state variables s and integral variables x. Moreover, we assume that with each integral variable x the following components are associated:

- a natural number $N(\mathsf{x})$ that denotes the number of abscissas of x,
- a range $rng(\mathsf{x}) = \{abs_0(\mathsf{x}),\ldots,abs_{N(\mathsf{x})}(\mathsf{x})\}$ where $abs_j(\mathsf{x})$ denotes the j-th abscissa,
- a number of weights for $wt_i^{[j,J]}(\mathsf{x})$ for $0 \leq j \leqslant J \leqslant N(\mathsf{x})$, and $0 \leqslant j \leqslant i \leqslant J$.

For boolean variable s, we define $rng(\mathsf{s}) = \{0,1\}$ and $N(\mathsf{s}) = 1$. The basic idea is that this representation facilitates numerical integration based on the quadrature formula:

$$\int_{x_j}^{x_J} f(\mathsf{x})\; d\mathsf{x} \;\;\approx\;\; \sum_{i=j}^{J} wt_i^{[j,J]}(\mathsf{x})\cdot f(x_i)$$

where $abs_i(\mathsf{x}) = x_i$ is the i-th abscissa of x. For example, if we assume a quadrature formula with *equally-spaced* abscissas then $x_i = x_0 + i * h$ for step-size $h = t/N(\mathsf{x})$. Well-known methods applied in practice, like trapezoidal, Simpson, and Romberg integration schemes belong to this category [31]. E.g. for the trapezoidal method $wt_j^{[j,J]} = wt_J^{[j,J]} = \frac{h}{2}$ and $wt_i^{[j,J]} = h$ for $j < i < J$.

Definition 1. A *multi-terminal decision diagram* (MTDD) over $\langle \mathsf{Var}, < \rangle$ is a rooted acyclic directed graph with vertex set V containing 3 types of vertices:

– each *boolean vertex* v is labelled by a boolean variable $var(v)$ and has two children $child_i(v) \in V$, $i = 0, 1$,
– each *integral vertex* v is labelled by an integral variable $var(v) = \mathsf{x}$ and an "interval" $Int(v) = [j, J]$ (where $0 \leqslant j \leqslant J \leqslant N(\mathsf{x})$) and has $N(\mathsf{x})+1$ children $child_0(v), \ldots, child_{N(\mathsf{x})}(v)$,
– each *terminal vertex* v is labelled by a value $val(v)$ where $val(v)$ is either a real number or $val(v) = \perp$.

such that $var(v) < var(w)$ for each non-terminal vertex v and non-terminal child w of v.

For $\mathsf{Var} = \{\mathsf{v}_1, \ldots, \mathsf{v}_n\}$ with $\mathsf{v}_i < \mathsf{v}_{i+1}$ we refer to an MTDD over $\langle \mathsf{Var}, < \rangle$ as an MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$. The constraint on the labelling of the non-terminal vertices is standard for (ordered) BDDs, and requires that on any path from the root to a terminal vertex, the variables respect the given ordering $<$. Note that an MTDD over $\langle \mathsf{Var}, < \rangle$ is also an MTDD over $\langle \mathsf{Var}', <' \rangle$ for any superset Var' of Var and total order $<'$ on Var' such that $\mathsf{v}_1 < \mathsf{v}_2$ iff $\mathsf{v}_1 <' \mathsf{v}_2$ for all $\mathsf{v}_1, \mathsf{v}_2 \in \mathsf{Var}$.

An MTDD M over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$ represents a partial n-ary function f_{M}, the values of which are obtained by traversing M starting at the root vertex as follows. For boolean vertex v, the edge from v to $child_0(v)$ represents the case $var(v)$ is false; the edge from v to $child_1(v)$ the case $var(v)$ is true. For integral vertex v, the edge from v to $child_j(v)$ stands for the case where the value of the real variable $var(v) = \mathsf{x}$ is $abs_j(\mathsf{x})$. The terminal vertices represent the possible function values where the special symbol $\perp$ indicates that the function value is undefined. Thus, if M is a MTDD over $(\mathsf{s}_1, \ldots, \mathsf{s}_k, \mathsf{x}_1, \ldots, \mathsf{x}_m)$ (where $\mathsf{s}_1, \ldots, \mathsf{s}_k$ are boolean variables and $\mathsf{x}_1, \ldots, \mathsf{x}_m$ are integral variables) then f_{M} is a partial function $\{0,1\}^k \times \mathbb{R}^m \to \mathbb{R}$ which is undefined (i.e. $f_{\mathsf{M}}(\cdot) = \perp$) for those arguments $(b_1, \ldots, b_k, y_1, \ldots, y_m)$ where $y_j \notin rng(\mathsf{x}_j)$ for some j.

$Int(v)$ is needed to perform the operator INTEGRATE (defined below). If $Int(v) = [j, J]$ then in vertex v the range of integration is $[x_j, x_J]$ where $x_* = abs_*(\mathsf{x})$.[2]

The relationship between BDDs, MTBDDs and MTDDs is as follows. An MTBDD is an MTDD without integral vertices; a BDD is an MTBDD with $val(v) \in \{0, 1\}$ for all terminal vertices v.

For efficiency reasons, an implementation will internally represent MTDDs in a *reduced* form [11] without redundant (boolean or integral) variables and isomorphic subgraphs. Reduced MTDDs yield a compact and canonical representation.

Operators on MTDDs. Several standard operators on BDDs [11] and MTBDDs [15, 4] can easily be modified for MTDDs. For space reasons we only briefly describe these operators and focus on the new operators, in particular substitution and computing integrals.[3]

Combining MTDDs via binary operators. The operator APPLY allows a point-wise application of the binary operator op (like summation or multiplication) to two MTDDs. For MTDDs M_1 and M_2 over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$,[4] $\textsc{Apply}(\mathsf{M}_1, \mathsf{M}_2, op)$ yields an MTDD M over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$ which represents the function

$$f_{\textsc{Apply}(\mathsf{M}_1, \mathsf{M}_2, op)} \;=\; f_{\mathsf{M}_1} \mathbin{op} f_{\mathsf{M}_2}.$$

Variable renaming. Operator $\textsc{Rename}(\mathsf{M}, \mathsf{v}_i, \mathsf{w})$ changes the variable labelling of any v_i-labelled vertex of MTDD M over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$ into w, for $\mathsf{w} \neq \mathsf{v}_j$, $0 < j \leqslant n$. (Thus, it yields a MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_{i-1}, \mathsf{w}, \mathsf{v}_{i+1}, \ldots, \mathsf{v}_n)$.)

Restriction. For boolean variable $\mathsf{v}_i = \mathsf{s}$ and $b \in \{0, 1\}$, $\textsc{Restrict}(\mathsf{M}, \mathsf{s}, b)$ denotes the MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_{i-1}, \mathsf{v}_{i+1}, \ldots, \mathsf{v}_n)$ that is obtained from M by replacing any edge from a vertex v to an s-labelled vertex w by an edge from v to $child_b(w)$, followed by removing all s-labelled vertices. In a similar way, $\textsc{Restrict}(\mathsf{M}, \mathsf{x}, x_j)$ is defined for $\mathsf{v}_i = \mathsf{x}$ an integral variable, $0 \leqslant j \leqslant N(\mathsf{x})$ and $x_j = abs_j(\mathsf{x})$.

[2] In most cases, e.g. in our **CSL** model checking procedure in Section 3.3, for any integral vertex v with $var(v) = \mathsf{x}$ and $Int(v) = [j, J]$, the branches representing the cases where $\mathsf{x} = x_i$ and $i < j$ or $i > J$ (i.e. the edges to the children $child_i(v)$) are not of importance. Accordingly, we may assume that any integral vertex v with $Int(v) = [j, J]$ has exactly $(J-j)+1$ children $child_j(v), \ldots, child_J(v)$.

[3] As it is standard in the (MT)BDD setting, an implementation of an MTDD-based method might use a hash table that organizes pointers to the vertices of the (already generated) MTDDs and a "computed table" which keeps book about the functions for which the MTDDs are already computed. These tables can be used to avoid the multiple computation of MTDDs for the same function and makes it possible to generate reduced MTDDs without an explicit call of Bryant's reduction procedure. We refer the reader to [11, 15, 4]. Moreover, Rudell's sifting algorithm [32] or other well-known heuristic methods for BDDs (see e.g. [22, 9]) that improve the chosen variable ordering can be adapted for MTDDs.

[4] Moreover, we assume that $Int(v_1) = Int(v_2)$ for all integral vertices v_i in M_i with $var(v_1) = var(v_2)$.

Comparison operators. Given MTDD M without integral vertices (i.e. M is a MTBDD) over n boolean variables, and interval I, $\textsc{Compare}(\mathsf{M}, I)$ is the BDD representing the function that equals 1 if $f_{\mathsf{M}}(b_1, \ldots, b_n) \in I$ and 0, otherwise.
Matrix/vector multiplication. Let MTBDDs Q and B without integral vertices over $2n$ and n boolean variables, respectively, represent the matrix $\mathbf{Q}$ and vector $\mathbf{b}$. Then, $\textsc{Multi}(\mathsf{Q}, \mathsf{B})$ denotes the MTBDD over n variables that represents the vector $\mathbf{Q} \cdot \mathbf{b}$. This operator can easily be modified for MTDDs without shared integral variables. E.g. if Q is a MTDD over $(\mathsf{s}_1, ..., \mathsf{s}_k, \mathsf{v}_1, .., \mathsf{v}_m)$ and B a MTDD over $(\mathsf{s}_1, .., \mathsf{s}_k, \mathsf{w})$ then $\mathsf{MULTI}(\mathsf{Q}, \mathsf{B})$ is a MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_m, \mathsf{w})$ represents the function

$$(v_1, \ldots, v_m, w) \mapsto \sum_{s_1, \ldots, s_k} f_{\mathsf{Q}}(s_1, \ldots, s_k, v_1, \ldots, v_m) \cdot f_{\mathsf{B}}(s_1, \ldots, s_k, w).$$

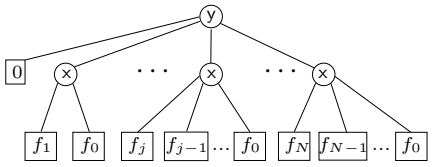
Substitution. Let M be a MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_n)$ where $\mathsf{v}_n = \mathsf{x}$ is an integral variable. Let N be a MTDD over $(\mathsf{y}_1, \ldots, \mathsf{y}_k)$ where y_i are integral variables not contained in $\{\mathsf{v}_1, \ldots, \mathsf{v}_{n-1}\}$. Assume that for any x-labelled vertex v in M we have $\mathit{Int}(v) = N(\mathsf{x})$ and that $\mathit{val}(v) \in \mathit{rng}(\mathsf{x}) \cup \{\bot\}$ for any terminal vertex v in N. Then, $\textsc{Subst}(\mathsf{M}, \mathsf{x}/\mathsf{N})$ denotes the MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_{n-1}, \mathsf{y}_1, \ldots, \mathsf{y}_k)$ which represents the partial function given by

$$f_{\textsc{Subst}(\mathsf{M},\mathsf{x}/\mathsf{N})}(v_1, \ldots, v_{n-1}, y_1, \ldots, y_k) \;=\; f_{\mathsf{M}}\left(v_1, \ldots, v_{n-1}, f_{\mathsf{N}}(y_1, \ldots, y_k)\right).$$

$\textsc{Subst}(\mathsf{M}, \mathsf{x}/\mathsf{N})$ results from M by replacing any x-labelled integral vertex v by the MTDD N' where the latter arises from N by replacing the terminal vertex with value $\mathit{abs}_j(\mathsf{x})$ in the subgraph of v by the terminal vertex with value $\mathit{val}(\mathit{child}_j(v))$. For instance, let N be a MTDD representation for the function $(y, x) \mapsto y - x$ if $y \geq x$ and which is undefined for all arguments (y, x) where $y < x$. More precisely, we assume that N is a MTDD over (y, x) where y is a fresh integral variable of the same type as x (i.e. the associated components $N(\cdot)$, $\mathit{rng}(\cdot)$ and $wt_i^{[j,J]}(\cdot)$ are the same for y and x). Then,

$$\textsc{Subst}(\mathsf{M}, \mathsf{x}/\mathsf{y} - \mathsf{x}) \;=\; \textsc{Subst}(\mathsf{M}, \mathsf{x}/\mathsf{N})$$

results from M by replacing any x-labelled vertex v by the subgraph depicted as:



In the figure (visualising the decision tree instead of the reduced MTDD), children are depicted from left to right and vertices $\mathit{child}_i(v'_j)$ where $i > j$ are omitted. Here, $f_j = \mathit{val}(\mathit{child}_j(v))$ for $0 \leqslant j \leqslant N$. More precisely, new vertices v', $v'_1, \ldots, v'_N$ are introduced with $\mathit{var}(v') = \mathsf{y}$, $\mathit{var}(v'_j) = \mathsf{x}$, $\mathit{child}_j(v') = v'_j$, $\mathit{Int}(v') = [0, N(\mathsf{x})]$ and $\mathit{Int}(v'_j) = [0, j]$.
Computing integrals. Let $\mathsf{v}_n = \mathsf{x}$ be an integral variable with $N(\mathsf{x}) = N$, $\mathit{rng}(\mathsf{x}) = \{\, x_0, \ldots, x_N \,\}$ and $wt_j^J(\mathsf{x}) = \alpha_j^J$. Then, $\textsc{Integrate}(\mathsf{M}, \mathsf{x})$ denotes the MTDD over $(\mathsf{v}_1, \ldots, \mathsf{v}_{n-1})$ that results from M by replacing any x-labelled vertex v with $\mathit{Int}(v) = [j, J]$ by the terminal vertex labelled by

$$\sum_{i=j}^{J} wt_i^{[j,J]} \cdot \mathit{val}(\mathit{child}_i(v))$$

Note that we assume that $\mathsf{x} = \mathsf{v}_n$; hence v_n is the largest in the variable ordering, and the children of a x-labelled vertex are terminal vertices. If $\mathit{Int}(v) = [0, N(\mathsf{x})]$ for each x-labelled vertex v in M then $\textsc{Integrate}(\mathsf{M}, \mathsf{x})$ represents an approximation of the function $(\ldots) \mapsto \int_0^t f_{\mathsf{M}}(\ldots, \mathsf{x}) d\mathsf{x}$.

3 Application: symbolic model checking for CTMCs

In this section, we show how the MTDD approach is applicable for verifying continuous-time Markov chains (CTMCs) that are at the basis of contemporary performance evaluation and reliability analysis methodologies.

For probabilistic systems, transition systems where branching is governed by discrete probability distributions, qualitative and quantitative model checking algorithms have been investigated extensively [1, 3, 6, 7, 8, 14, 18, 19, 20, 25, 30, 36]. In a qualitative setting it is checked whether a property holds with probability 0 or 1; in a quantitative setting it is typically verified whether the probability for a certain property meets given lower- or upper-bounds. For discrete-time systems, the quantitative approach has been investigated quite thoroughly: model checking algorithms have been developed for fully probabilistic transition systems [3, 6, 18, 25], like discrete-time Markov chains or generative transition systems, as well as for probabilistic systems that contain non-determinism [7, 8, 10, 20], like Markov decision processes.

In this section, we consider real-time probabilistic systems and present an approximative MTDD-based model checking algorithm for CTMCs. A branching-time logic called *continuous-time stochastic logic* (**CSL**) is used to express properties over CTMCs. This logic is an extension of the (equally named) logic by Aziz et. al [2] with an operator to reason about steady-state probabilities.

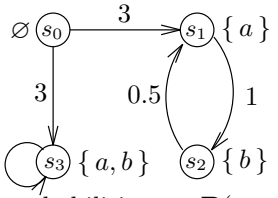
We proceed in the following way. Section 3.1 introduces the necessary concepts of CTMCs. Section 3.2 presents the logic **CSL** and provides some useful characterisations of **CSL**-formulas that facilitate a model checking procedure. In Section 3.3, we explain how CTMCs can be encoded with MTDDs and present an approximative symbolic model checking algorithm.

3.1 Continuous-time Markov chains

Basic definitions. Let AP be a fixed, finite set of atomic propositions. A (labelled) *continuous-time Markov chain* (CTMC for short) is a tuple $\mathcal{M} = (S, \mathbf{Q}, L)$ where S is a finite set of *states*, $\mathbf{Q} : S \times S \to \mathbb{R}_{\geqslant 0}$ the *generator matrix*[5], and $L : S \to 2^{AP}$ the *labelling* function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions $a \in AP$ that are valid in s.

Intuitively, $\mathbf{Q}(s, s')$ specifies that the probability of moving from state s to s' within t time-units (for positive t) is $1 - e^{-\mathbf{Q}(s,s') \cdot t}$, an exponential distribution with rate $\mathbf{Q}(s, s')$. If $\mathbf{Q}(s, s') > 0$ for more than one state s', a competition between the transitions is assumed to exist, known as the *race condition*. Let $\mathbf{E}(s) = \sum_{s' \in S} \mathbf{Q}(s, s')$, the total rate at which any transition emanating from state s is taken. This rate is the reciprocal of the mean sojourn time in s. More precisely, $\mathbf{E}(s)$ specifies that the probability of leaving s within t time-units (for positive t) is $1 - e^{-\mathbf{E}(s) \cdot t}$, due to the fact that the minimum of exponential distributions (competing in a race) is characterised by the sum of their rates. Consequently, the probability of moving from state s to s' by a single transition, denoted $\mathbf{P}(s, s')$, is determined by the probability that the delay of going from s to s' finishes before the delays of other outgoing edges from s; formally, $\mathbf{P}(s, s') = \mathbf{Q}(s, s')/\mathbf{E}(s)$ (except if s is an absorbing state, i.e. if $\mathbf{E}(s) = 0$; in this case we define $\mathbf{P}(s, s') = 0$). Remark that the matrix $\mathbf{P}$ describes an embedded discrete time Markov chain. (For a more extensive treatment of CTMCs see [34].)

Example 1. As a running example we consider $AP = \{ a, b \}$, $S = \{ s_0, \ldots, s_3 \}$ with $L(s_0) = \varnothing$, $L(s_1) = \{ a \}$, $L(s_2) = \{ b \}$ and $L(s_3) = \{ a, b \}$. The details of the CTMC are:



$$\mathbf{Q} = \begin{pmatrix} 0 & 3 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \text{ and } \mathbf{E} = \begin{pmatrix} 6 \\ 1 \\ 0.5 \\ 2 \end{pmatrix}.$$

Some transition probabilities are $\mathbf{P}(s_0, s_3) = \mathbf{P}(s_0, s_1) = \frac{1}{2}$ and $\mathbf{P}(s_1, s_2) = 1$.

Paths. A *path* is a (finite or infinite) alternating sequence $s_0, t_0, s_1, t_1, s_2, t_2, \ldots$, written as

$$\sigma \;=\; s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \;\ldots,$$

with for natural i, $s_i \in S$ and $t_i \in \mathbb{R}_{>0}$ such that $\mathbf{Q}(s_i, s_{i+1}) > 0$, if σ is infinite. Otherwise, if $\sigma = s_0 \xrightarrow{t_0} \ldots \xrightarrow{t_{l-1}} s_l$ is finite, we require that s_l is absorbing, and $\mathbf{Q}(s_i, s_{i+1}) > 0$ for all $i < l$. For σ a path, $t \in \mathbb{R}_{>0}$ and natural i let $\sigma[i] = s_i$, the i-th state of σ, $\delta(\sigma, i) = t_i$, the time spent in state s_i, and $\sigma(t) = s_0$ for $t < t_0$, and $\sigma(t) = \sigma[i]$ where i is the smallest index i with $t \leqslant \sum_{0 \leqslant j \leqslant i} t_j$, otherwise.

[5] Whereas usually the diagonal elements are defined as $\mathbf{Q}(s, s) = -\sum_{s' \neq s} \mathbf{Q}(s, s')$ we allow self-loops. This does not affect the transient and steady state behaviour of the chain, but allows the standard interpretations of the next-state operator of the logic.

(For σ a finite path with absorbing state s_l, $\sigma[i]$ and $\delta(\sigma,i)$ are only defined for $i \leqslant l$, $\delta(\sigma,l)=\infty$, and $\sigma(t)=s_l$ for $t > t_1+\ldots+t_{l-1}$.) Let $Path(s)$ denote the set of paths in $\mathcal{M}$ starting in s, and $Reach(s)$ the set of states reachable from s.

Borel space. Let $s_0,\ldots,s_k \in S$ with $\mathbf{Q}(s_i,s_{i+1})>0$, $(0<i<k)$, and $I_0,\ldots,I_{k-1}$ non-empty intervals in $\mathbb{R}_{\geqslant 0}$. Then, $C(s_0,I_0,\ldots,I_{k-1},s_k)$ denotes the *cylinder set* consisting of all paths $\sigma \in Path(s_0)$ such that $\sigma[i]=s_i$ $(i \leqslant k)$, and $\delta(\sigma,i) \in I_i$ $(i<k)$. Let $\mathcal{F}(Path(s))$ be the smallest σ-algebra on $Path(s)$ which contains all sets $C(s,I_0,\ldots,I_{k-1},s_k)$ where $s_0,\ldots,s_k$ ranges over all sequences of states such that $s=s_0$ and $\mathbf{Q}(s_i,s_{i+1})>0$ $(0 \leqslant i<k)$ and $I_0,\ldots,I_{k-1}$ ranges over all sequences of non-empty intervals in $\mathbb{R}_{\geqslant 0}$. The probability measure $\mathcal{P}$ on $\mathcal{F}(Path(s))$ is the unique measure defined by induction on k by $\mathcal{P}(C(s_0))=1$ and for $k \geqslant 0$:

$$\mathcal{P}(C(s_0,\ldots,s_k,I',s') = \mathcal{P}(C(s_0,\ldots,s_k)) \cdot \mathbf{P}(s_k,s') \cdot \left(e^{-\mathbf{E}(s_k)\cdot a} - e^{-\mathbf{E}(s_k)\cdot b}\right)$$

where $a=\inf I'$ and $b=\sup I'$. (For $b=\infty$ and $\lambda>0$ let $e^{-\lambda\cdot\infty}=0$.)

3.2 The continuous stochastic logic CSL

CSL is a branching-time, CTL-like temporal logic where the state-formulas are interpreted over states of a CTMC. It adopts operators of PCTL [25], like a time-bounded until operator and a probabilistic operator asserting that the probability for a certain event meets given lower or upper bounds. We treat a slight variant of the (equally named) logic of [2] with, for reasons of simplicity, an unnested time-bounded until operator plus a novel steady-state probability operator.

Syntax. Let $a \in AP$, $p \in [0,1]$ and $\bowtie \in \{\leqslant,<,\geqslant,>\}$. The *state-formulas* of **CSL** are defined by the grammar

$$\Phi ::= tt \;\Big|\; a \;\Big|\; \Phi \wedge \Phi \;\Big|\; \neg\Phi \;\Big|\; \mathcal{S}_{\bowtie p}(\Phi) \;\Big|\; \mathcal{P}_{\bowtie p}(\varphi)$$

where for $t \in \mathbb{R}_{\geqslant 0}$ *path-formulas* are defined by

$$\varphi ::= X\Phi \;\Big|\; \Phi\,\mathcal{U}\,\Phi \;\Big|\; \Phi\,\mathcal{U}^{\leqslant t}\,\Phi.$$

The other boolean connectives are derived in the usual way, i.e. $ff=\neg tt$, $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, and $\Phi_1 \rightarrow \Phi_2 = \neg\Phi_1 \vee \Phi_2$. The intended meaning of the temporal operators $\mathcal{U}$ ("until") and X ("next step") is standard. The temporal operator $\mathcal{U}^{\leqslant t}$ is the real-time variant of $\mathcal{U}$; $\Phi_1\,\mathcal{U}^{\leqslant t}\,\Phi_2$ asserts that $\Phi_1\,\mathcal{U}\,\Phi_2$ will be satisfied in the time interval $[0,t]$; i.e. there is some $x \in [0,t]$ such that Φ_1 continuously holds during the interval $[0,x[$ and Φ_2 becomes true at time instant x. The state formula $\mathcal{S}_{\bowtie p}(\Phi)$ asserts that the steady-state probability for a Φ-state falls in the interval $I_{\bowtie p}=\{\,q \in [0,1] \mid q \bowtie p\,\}$. $\mathcal{P}_{\bowtie p}(\varphi)$ asserts that the probability measure of the paths satisfying φ falls in the interval $I_{\bowtie p}$.

Temporal operators like $\Diamond$, $\Box$ and their real-time variants $\Diamond^{\leqslant t}$ or $\Box^{\leqslant t}$ can be derived, e.g. $\mathcal{P}_{\bowtie p}(\Diamond^{\leqslant t}\,\Phi) = \mathcal{P}_{\bowtie p}(tt\,\mathcal{U}^{\leqslant t}\,\Phi)$ and $\mathcal{P}_{\geqslant p}(\Box\,\Phi) = \mathcal{P}_{\leqslant 1-p}(\Diamond\,\neg\Phi)$. As an example property, $\mathcal{P}_{\geqslant 0.99}(\Box\ (req \rightarrow \mathcal{P}_{\geqslant 1}(\Diamond^{\leqslant 5}\,resp)))$ asserts that there is a chance of at least 99% that every request will be responded within the next 5 time-units.

Semantics. The state-formulas are interpreted over the states of a CTMC. Let $\mathcal{M}=(S,\mathbf{Q},L)$ with proposition labels in AP. The definition of the satisfaction relation $\models \subseteq S \times \mathbf{CSL}$ is as follows. Let $Sat(\Phi)=\{\,s \in S \mid s \models \Phi\,\}$.

$s \models tt$ for all $s \in S$	$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_i, i{=}1,2$
$s \models a$ iff $a \in L(s)$	$s \models \mathcal{S}_{\bowtie p}(\Phi)$ iff $\pi_{Sat(\Phi)}(s) \in I_{\bowtie p}$
$s \models \neg\Phi$ iff $s \not\models \Phi$	$s \models \mathcal{P}_{\bowtie p}(\varphi)$ iff $Prob(s,\varphi) \in I_{\bowtie p}$.

Here, $\pi_{S'}(s)$ denotes the steady-state probability for $S' \subseteq S$ with respect to state s, i.e.

$$\pi_{S'}(s) \;=\; \lim_{t\rightarrow\infty} \mathcal{P}\{\,\sigma \in Path(s) \mid \sigma(t) \in S'\,\}.$$

The limit exists, a consequence of S being finite [34]. Obviously, $\pi_{S'}(s)=\sum_{s'\in S'}\pi_{s'}(s)$, where we write $\pi_{s'}(s)$ instead of $\pi_{\{\,s'\,\}}(s)$. We let $\pi_{\varnothing}(s)=0$. $Prob(s,\varphi)$ denotes the probability measure of all paths $\sigma \in Path(s)$ satisfying φ, i.e.

$$Prob(s,\varphi) \;=\; \mathcal{P}\{\,\sigma \in Path(s) \mid \sigma \models \varphi\,\}.$$

The fact that, for each state s, the set $\{\,\sigma \in Path(s) \mid \sigma \models \varphi\,\}$ is measurable, follows by easy verification. The satisfaction relation (also denoted $\models$) for the path-formulas is defined as usual:

$$\begin{array}{ll}
\sigma \models X\Phi & \text{iff } \sigma[1] \text{ is defined and } \sigma[1] \models \Phi \\
\sigma \models \Phi_1 \,\mathcal{U}\, \Phi_2 & \text{iff } \exists k \geqslant 0.\ (\sigma[k] \models \Phi_2 \wedge \forall 0 \leqslant i < k.\, \sigma[i] \models \Phi_1) \\
\sigma \models \Phi_1 \,\mathcal{U}^{\leqslant t}\, \Phi_2 & \text{iff } \exists x \in [0,t].\ (\sigma(x) \models \Phi_2 \wedge \forall y \in [0,x[.\, \sigma(y) \models \Phi_1)\,.
\end{array}$$

In the remainder of this section we present alternative characterisations for $\pi_{S'}(s)$ and $Prob(s,\varphi)$ that will serve as a basis for our model checking algorithm. Since the derivation of these characterisations from the theory of CTMCs and DTMCs is not much involved, proofs are omitted.

Computing steady-state probabilities. It is well known that the steady state probabilities exist for arbitrary CTMCs and can be obtained by solving a *linear equation system* [34]. Let G be the underlying directed graph of $\mathcal{M}$ where vertices represent states and where there is an edge from s to s' iff $\mathbf{Q}(s,s') > 0$. Sub-graph B is a *bottom strongly connected component* (bscc) of G if it is a strongly connected component such that for any $s \in B$, $Reach(s) \subseteq B$. We have $\pi_{s'}(s) = 0$ iff s' does not occur in any bscc reachable from s. Let B be a bscc of G with $Reach(s) \cap B \neq \varnothing$ (or equivalently, $B \subseteq Reach(s)$) and assume that a_B is an atomic proposition such that $a_B \in L(s)$ iff $s \in B$. Then $\Diamond a_B$ is a path-formula in **CSL** and $Prob(s, \Diamond B) = Prob(s, \Diamond a_B)$ is the probability of reaching B from s at some time t. For $s' \in B$, $\pi_{s'}(s)$ is given by $\pi_{s'}(s) = Prob(s, \Diamond B) \cdot \pi_B(s')$ where $\pi_B(s') = 1$ if $B = \{s'\}$, and otherwise

$$\pi_B(s') = \frac{\pi'_B(s')/\mathbf{E}(s')}{\sum_{s \in B} \pi'_B(s)/\mathbf{E}(s)}$$

for which $\pi'_B(s'')$ satisfies the linear equation system

$$\pi'_B(s'') = \sum_{s \in B} \mathbf{P}(s,s'') \cdot \pi'_B(s) \text{ such that } \sum_{s \in B} \pi'_B(s) = 1.$$

Example 2. Consider $\mathcal{S}_{>0.5}(\Phi)$ where $\Phi = (a \wedge b) \vee \mathcal{P}_{\leqslant 0.8}(a\,\mathcal{U}^{\leqslant 2}\, b)$, for the CTMC of Example 1. Note that the CTMC is not strongly connected, since e.g. s_3 cannot be reached from s_2 (and vice versa). Assume that Φ is valid in states s_2 and s_3, and invalid otherwise (as we will see later on). Then we have $s_0 \models \mathcal{S}_{>0.5}(\Phi)$, since from s_0 both the bscc $B_1 = \{\, s_3 \,\}$ and the bscc $B_2 = \{\, s_1, s_2 \,\}$ can be reached with probability $\mathbf{P}(s_0, s_3) = \mathbf{P}(s_0, s_2) = 1/2$, and s_2 has a non-zero steady-state probability in B_2. Thus, the steady-state probability for Φ exceeds 0.5. Formally:

$$\pi_{Sat(\Phi)}(s_0) = \pi_{\{\, s_2, s_3 \,\}}(s_0) = \pi_{s_2}(s_0) + \pi_{s_3}(s_0)$$

where $\pi_{s_2}(s_0) = Prob(s_0, \Diamond B_2) \cdot \pi_{B_2}(s_2)$ and $\pi_{s_3}(s_0) = Prob(s_0, \Diamond B_1) \cdot \pi_{B_1}(s_3)$. We have $Prob(s_0, \Diamond B_1) = Prob(s_0, \Diamond B_2) = 1/2$, $\pi_{B_1}(s_3) = 1$, and obtain $\pi'_{B_2}(s_2) = 1/2$ by solving the equation system

$$\pi'_{B_2}(s_2) = \pi'_{B_2}(s_1),\ \pi'_{B_2}(s_1) = \pi'_{B_2}(s_2),\ \pi'_{B_2}(s_1) + \pi'_{B_2}(s_2) = 1.$$

Subsequently, calculation of $\pi_{B_2}(s_2)$ yields $2/3$. Thus, $\pi_{\{\, s_2, s_3 \,\}}(s_0) = 1/2 \cdot 2/3 + 1/2 = 5/6$ which indeed exceeds 0.5.

Computing $Prob(s,\varphi)$. The next step and until operator can be handled as in the discrete-time probabilistic case, cf. [18, 25, 5]. This entails that model checking for these formulas can be carried out by well-known methods. The values $Prob(s, X\Phi)$ can be calculated by multiplying the transition probability matrix $\mathbf{P}$ with the (boolean) vector $\mathbf{i}_\Phi = (i_\Phi(s))_{s \in S}$ characterising $Sat(\Phi)$, i.e. $i_\Phi(s) = 1$ if $s \models \Phi$, and 0 otherwise. $Prob(s, \Phi_1 \,\mathcal{U}\, \Phi_2)$ can be obtained by solving a linear equation system of the form $\mathbf{x} = \overline{\mathbf{P}} \cdot \mathbf{x} + \mathbf{i}_{\Phi_2}$ where $\overline{\mathbf{P}}(s,s') = \mathbf{P}(s,s')$ if $s \models \Phi_1 \wedge \neg\Phi_2$ and 0 otherwise. The vector $(\mathcal{P}_{(s,)}\Phi_1 \,\mathcal{U}\, \Phi_2))_{s \in S}$ is the least solution of this set of equations. Note, however, that this system of equations can, in general, have more than one solution. The least solution can be obtained by applying an iterative approximative method or a graph analysis combined with standard methods (like Gauss elimination) to solve regular linear equation systems.

The basis for calculating the probabilities $Prob(s, \Phi\,\mathcal{U}^{\leq t}\Phi_2)$ is the following result.

Theorem 2. *The function $S \times \mathbb{R}_{\geqslant 0} \to [0,1]$, $(s,t) \mapsto Prob(s, \Phi_1 \,\mathcal{U}^{\leqslant t}\, \Phi_2)$ is the least fixed point of the higher-order operator $\Omega : (S \times \mathbb{R}_{\geqslant 0} \to [0,1]) \to (S \times \mathbb{R}_{\geqslant 0} \to [0,1])$ where*[6]

$$\Omega(F)(s,t) = \begin{cases} 1 & \text{if } s \models \Phi_2 \\ \sum_{s' \in S} \mathbf{Q}(s,s') \cdot \int_0^t e^{-\mathbf{E}(s)\cdot x} \cdot F(s', t-x)\ dx & \text{if } s \models \Phi_1 \wedge \neg\Phi_2 \\ 0 & \text{otherwise.} \end{cases}$$

Example 3. Consider our running CTMC example, $\Phi = (a \wedge b) \vee \mathcal{P}_{\leqslant 0.8}(a\,\mathcal{U}^{\leqslant 2}\, b)$ and suppose we want to check $s_1 \models \Phi$. It follows from $\mathbf{Q}(s_1, s_2) = 1$ that the probability of reaching b-state s_2 from s_1 within two time-units equals $1 - e^{-1\cdot 2} \approx 0.864664$. Formally, we have $s_2 \models \Phi$, since $s_2 \models b$, and $s_1 \not\models \Phi$, since $s_1 \not\models a \wedge b$ and using Theorem 2 we have that $Prob(s_1, a\,\mathcal{U}^{\leqslant 2}\, b)$ equals

$$\sum_{s' \in S} \mathbf{Q}(s_1, s') \cdot \int_0^2 e^{-\mathbf{E}(s_1)\cdot x} \cdot F(s', 2-x)\ dx = \int_0^2 e^{-x} dx = [-e^{-x}]_0^2 = 1 - e^{-2}$$

which exceeds 0.8.

Theorem 2 is due to the fact that the probability density function of the sojourn time in state s is given by $\mathbf{E}(s) \cdot e^{-\mathbf{E}(s)\cdot t}$. The resulting recursive integral formula can be reformulated into a *heterogeneous linear differential equation* of the form

$$\mathbf{y}'(t) \;=\; \overline{\mathbf{Q}} \cdot \mathbf{y}(t) \;+\; \mathbf{b}(t)$$

where $\mathbf{y}(t)$ denotes the vector $(Prob(s, \Phi_1\, \mathcal{U}^{\leq t} \Phi_2))_{s \in S}$, and $\overline{\mathbf{Q}}$ is derived from $\mathbf{Q}$, by $\overline{\mathbf{Q}}(s,s') = \mathbf{Q}(s,s')$ if $s, s' \models \Phi_1 \wedge \neg\Phi_2$, and otherwise $\overline{\mathbf{Q}}(s,s') = 0$. The vector $\mathbf{b}(t) = (b_s(t))_{s \in S}$ is given by $b_s(t) = \sum_{s' \in Sat(\Phi_2)} \mathbf{Q}(s,s') \cdot e^{-\mathbf{E}(s)t}$, if $s \models \Phi_1 \wedge \neg\Phi_2$, and otherwise $b_s(t) = 0$. The vector $(Prob(s, \Phi_1\, \mathcal{U}^{\leq t} \Phi_2))_{s \in S}$ agrees with the following solution of the above heterogeneous linear differential equation:

$$\mathbf{y}(t) \;=\; e^{\overline{\mathbf{Q}}t} \cdot \left(\mathbf{i}_{\Phi_2} \;+\; \int_0^t e^{-\overline{\mathbf{Q}}x} \cdot \mathbf{b}(x)\ dx \right), \text{ where } \quad e^{\overline{\mathbf{Q}}x} \;=\; \sum_{k=0}^{\infty} \frac{(\overline{\mathbf{Q}}x)^k}{k!}.$$

Unfortunately, it is not clear (at least to the authors) how to obtain a closed solution for the above integral. Using a numerical approximation method instead is also not an accurate way out, essentially because known approximative methods for computing $e^{\mathbf{A}x}$ (for some square matrix $\mathbf{A}$) are instable, yet computationally expensive [34]. For that reasons, our algorithm to compute $Prob(s, \Phi_1\, \mathcal{U}^{\leqslant t}\, \Phi_2)$ is directly based on the last result of Theorem 2. The result suggests the following *iterative* method to approximate $Prob(s, \Phi_1\, \mathcal{U}^{\leqslant t}\, \Phi_2)$: let $F_0(s,t) = 0$ for all s, t and $F_{k+1} = \Omega(F_k)$. Then,

$$\lim_{k \to \infty} \; F_k(s,t) \;=\; Prob(s, \Phi_1\, \mathcal{U}^{\leqslant t}\, \Phi_2).$$

(The general nested time-bounded until in [2] can be treated in a similar way.) Each step in the iteration amounts to solve an integral of the following form:

$$F_{k+1}(s,t) = \int_0^t \sum_{s' \in S} \mathbf{Q}(s,s') \cdot e^{-\mathbf{E}(s)\cdot x} \cdot F_k(s', t-x)\ dx,$$

if $s \models \Phi_1 \wedge \neg\Phi_2$. These integrals can be solved numerically.

3.3 The model checking algorithm

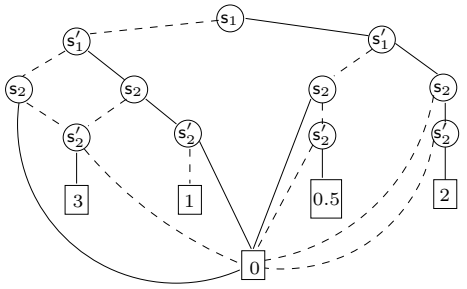
Our symbolic model checking algorithm for **CSL** works as follows. Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a CTMC which is represented by a MT(B)DD Q over $2n$ boolean variables as explained below. For each **CSL**-state-formula Φ we define a BDD $\mathsf{Sat}[\![\,\Phi\,]\!]$ over $(\mathsf{s}_1, \ldots, \mathsf{s}_n)$ that represents the characteristic function of the set $Sat(\Phi)$; for each **CSL**-path formula φ we define a MTBDD $\mathsf{PR}[\![\,\varphi\,]\!]$ representing the function $s \mapsto Prob(s, \varphi)$.

Encoding CTMCs by MT(B)DDs. In BDD-approaches transition systems are symbolically represented by encoding states by bit vectors, and encoding the transition relation by its characteristic function. To represent the generator matrix of a CTMC by a MTDD we abstract from the names of the states,

[6] The underlying partial order on $S \times \mathbb{R}_{\geqslant 0} \to [0,1]$ is defined for F_1, $F_2 : S \times \mathbb{R}_{\geqslant 0} \to [0,1]$ by $F_1 \leqslant F_2$ iff $F_1(s,t) \leqslant F_2(s,t)$ for all s, t.

and instead, similar to [17], use binary tuples of atomic propositions that are true in that state. Using this scheme, CTMCs are encoded as MTDDs as follows. Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC. We assume that $|S| = 2^n$ and that the labelling function L is injective. (Any labelled CTMCs may be transformed into one satisfying these conditions by adding dummy states and new propositions.) We fix an enumeration $a_1, \ldots, a_n$ of atomic propositions and identify each state s with the boolean n-tuple $(b_1, \ldots, b_n)$ where $b_i = 1$ iff $a_i \in L(s)$. In what follows, we assume that $S = \{0,1\}^n$ where we identify each state s with its encoding and the generator matrix $\mathbf{Q}$ with the function $F : \{0,1\}^{2n} \to \mathbb{R}$ where $F(\mathsf{s}_1, \mathsf{s}'_1, \ldots, \mathsf{s}_n, \mathsf{s}'_n) = \mathbf{Q}((\mathsf{s}_1, \ldots, \mathsf{s}_n), (\mathsf{s}'_1, \ldots, \mathsf{s}'_n))$. We represent $\mathcal{M}$ by the MTBDD Q for $\mathbf{Q}$ over (s_1, $\mathsf{s}'_1, \ldots, \mathsf{s}_n, \mathsf{s}'_n$), in other words $f_{\mathsf{Q}} = F$. Note that Q does not contain integral variables and hence is a MTBDD.

Example 4. Consider the CTMC of Example 1. According to the above scheme we encode the states by $s_0 \mapsto 00$, $s_1 \mapsto 01$, $s_2 \mapsto 10$ and $s_3 \mapsto 11$. The function $F = f_{\mathsf{Q}}$ and the MTDD Q are given by:



$$
\begin{aligned}
F(0,1,0,1) &= 3\\
F(0,0,0,1) &= 3\\
F(0,1,1,0) &= 1\\
F(1,0,0,1) &= 0.5\\
F(1,1,1,1) &= 2\\
F(s_1, s'_1, s_2, s'_2) &= 0 \text{ otherwise}
\end{aligned}
$$

where dotted lines denote zero-edges and solid lines one-edges.

The model checking algorithm: Besides the described operators on MTDDs, our model checking algorithm uses methods for boolean combinators and for a BDD-based graph analysis, e.g. to obtain the bottom strongly connected components of the graph underlying a CTMC, and MTBDD-based methods for solving linear equation systems, e.g. to compute the probabilities $\pi_{s'}(s)$. For these algorithms we refer to [11, 15, 4].

By applying standard operators on MTBDDs we determine the MTBDDs P, representing the transition probability matrix $\mathbf{P}$ of $\mathcal{M}$, SP representing the steady-state probabilities $\pi_{s'}(s)$ for $s, s' \in S$, and E representing the total rates $\mathbf{E}(s)$. The BDDs $\mathsf{Sat}[\![\, \Phi \,]\!]$ are computed by a recursive procedure:

$$
\begin{aligned}
\mathsf{Sat}[\![\, tt \,]\!] &= \mathbf{1}\\
\mathsf{Sat}[\![\, a_i \,]\!] &= \text{the BDD for the boolean function } (\mathsf{s}_1, \ldots, \mathsf{s}_n) \mapsto \mathsf{s}_i\\
\mathsf{Sat}[\![\, \neg\Phi \,]\!] &= \neg\mathsf{Sat}[\![\, \Phi \,]\!]\\
\mathsf{Sat}[\![\, \Phi_1 \wedge \Phi_2 \,]\!] &= \mathsf{Sat}[\![\, \Phi_1 \,]\!] \wedge \mathsf{Sat}[\![\, \Phi_2 \,]\!]\\
\mathsf{Sat}[\![\, \mathcal{S}_{\trianglerighteq p}(\Phi) \,]\!] &= \textsc{Compare}(\textsc{Multi}(\mathsf{SP}, \mathsf{Sat}[\![\, \Phi \,]\!]'), I_{\trianglerighteq p})\\
\mathsf{Sat}[\![\, \mathcal{P}_{\trianglerighteq p}(\varphi) \,]\!] &= \textsc{Compare}(\mathsf{PR}[\![\, \varphi \,]\!], I_{\trianglerighteq p}).
\end{aligned}
$$

Here, $\mathbf{1}$ denotes the BDD consisting of a single, terminal vertex labelled by 1. $\mathsf{Sat}[\![\, a_i \,]\!]$ is a BDD consisting of a single state-vertex v labelled with s_i such that $\mathit{child}_0(v)$ and $\mathit{child}_1(v)$ are labelled with 0 and 1, respectively. $\mathsf{Sat}[\![\, \Phi \,]\!]'$ denotes $\mathsf{Sat}[\![\, \Phi \,]\!]$ where s_i is renamed into s'_i (using nested applications of $\textsc{Rename}$). The formula for $\mathsf{Sat}[\![\, \mathcal{S}_{\trianglerighteq p}(\Phi) \,]\!]$ is justified by the characterisation of $\pi_{Sat(\Phi)}(s)$ in Section 3.2.

MTBDD $\mathsf{PR}[\![\, \varphi \,]\!]$ is defined by induction over the structure of φ. For $\varphi = X\Phi$ and $\varphi = \Phi_1 \,\mathcal{U}\, \Phi_2$, the MTBDD $\mathsf{PR}[\![\, \varphi \,]\!]$ can be obtained in the same way as for the discrete-time probabilistic case [6]. For the time-bounded until-operator we define:

$$\mathsf{PR}[\![\, \Phi_1 \,\mathcal{U}^{\leqslant t}\, \Phi_2 \,]\!] = \textsc{BoundedUntil}(\mathsf{Q}, \mathsf{E}, \mathsf{Sat}[\![\, \Phi_1 \,]\!], \mathsf{Sat}[\![\, \Phi_2 \,]\!], t, k_{max}, \epsilon)$$

where k_{max} indicates the maximum number of iterations and ϵ is the maximum desired tolerance of the approximation. The algorithm for $\textsc{BoundedUntil}$ is listed in Table 1. Here, F_0 represents the first approximation $F_0(s,t) = 0$. First the MTDD H for the function

$$H(s, s', x) = \mathbf{Q}(s, s') \cdot e^{-\mathbf{E}(s) \cdot x}$$

is constructed. This requires as input the MTDD X which consists of a single, integral vertex labelled by x with $N{+}1$ terminal vertices labelled with the values $x_0, \ldots, x_N$. Here we assume that $N = N(\mathsf{x})$ is

algorithm BOUNDEDUNTIL $(\mathsf{Q}, \mathsf{E}, \mathsf{B}_1, \mathsf{B}_2, t, k_{max}, \epsilon)$:
begin $\mathsf{F}_0 := \mathbf{0}; k := 0;$
$\mathsf{H} :=$ APPLY$(\mathsf{Q},$ APPLY$(\mathsf{E}, \mathsf{X}, (q_1, q_2) \mapsto e^{-q_1 \cdot q_2}), \cdot);$
repeat
$\mathsf{I} :=$ SUBST$(\mathsf{F}_k, \mathsf{x}/\mathsf{y} - \mathsf{x});$
$\mathsf{J} :=$ APPLY$(\mathsf{H}, \mathsf{I}', \cdot);$
$\mathsf{K} :=$ MULTI$(\mathsf{J}, \mathbf{1});$
$\mathsf{L} :=$ RENAME(INTEGRATE$(\mathsf{K}, \mathsf{x}), \mathsf{y}, \mathsf{x});$
$\mathsf{F}_{k+1} :=$ APPLY(APPLY$(\mathsf{L}, \mathsf{B}_1, min), \mathsf{B}_2, max);$
$\mathsf{D}_{k+1} :=$ APPLY$(\mathsf{F}_{k+1}, \mathsf{F}_k, -);$
$\Delta_{k+1} := \max_{s,j} \left| f_{\mathsf{D}_{k+1}}(s, x_j) \right|;$
$k := k + 1;$
until $(k = k_{max}$ or $\Delta_k \leqslant \epsilon);$
if $\Delta_k \leqslant \epsilon$ **then return** RESTRICT$(\mathsf{F}_k, \mathsf{x}, t)$ **else return** *error*;
(* if approximation could be computed within the allowed tolerance *)
(* ϵ then RESTRICT$(\mathsf{F}_k, \mathsf{x}, t)$ approximates $\lim_{k\to\infty}\ F_k(s,t)$. *)
end.

Table 1. Algorithm for BOUNDEDUNTIL

"sufficiently large" according to the chosen quadrature formula with equally-spaced abscissas $abs_i(\mathsf{x}) = x_i = i * h$ where $h = t/N$. In the first five steps of the iteration, the MTDD-representation of F_{k+1} is constructed systematically. More precisely, F_k represents (approximations) for the values $F_k(s, x_j)$ $(0 \leqslant j \leqslant N)$ where $x_j = j \cdot h$ and $h = t/N$. I' represents the function $f_{\mathsf{I}'}(s', y, x) = F_k(s', y{-}x)$. MTDD J represents the function

$$f_{\mathsf{J}}(s, s', y, x) = \mathbf{Q}(s, s') \cdot e^{-\mathbf{E}(s)\cdot x} \cdot F_k(s', y{-}x).$$

For this, we consider J as the MTDD representation of a matrix whose rows are indexed by triples (s, y, x), and whose columns are indexed by s'. Matrix-vector multiplication with $\mathbf{1}$, the MTDD over $(\mathsf{s}'_1, \ldots, \mathsf{s}'_n)$ that represents the constant function $s' \mapsto 1$, yields MTDD K over $(\mathsf{s}_1, \ldots, \mathsf{s}_n, \mathsf{y}, \mathsf{x})$ representing

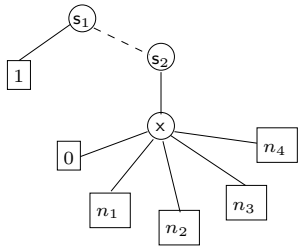
$$f_{\mathsf{K}}(s, y, x) = \sum_{s' \in S} f_{\mathsf{J}}(s, s', y, x).$$

By INTEGRATE(K, x) the integrals $\int_0^{x_J} f_{\mathsf{K}}(s, x_J, x)dx$ are approximated by $\sum_i \alpha_i^{[0,J]} \cdot f_{\mathsf{K}}(s, x_J, x_i)$. Here, $\alpha_i^{[0,J]} = wt_i^{[0,J]}(\mathsf{x})$ are the weights of the chosen type of quadrature formulas for computing integrals over the interval $[0, x_J]$ with equally-spaced abscissas and step size $h = t/N$. For generating the MTDD F_{k+1} that represents function $(s, x) \mapsto 1$ if $s \models \Phi_2$, $(s, x) \mapsto f_{\mathsf{L}}(s, x)$ if $s \models \Phi_1 \wedge \neg\Phi_2$ and $(s, x) \mapsto 0$ otherwise, we use the fact that

$$F_{k+1}(s, x) \ = \ \max\{\ \min\{\ f_{\mathsf{Sat}[\![\Phi_1]\!]}(s), f_{\mathsf{L}}(s, x)\ \}, f_{\mathsf{Sat}[\![\Phi_2]\!]}(s)\ \}.$$

Finally, after the calculation of F_{k+1}, the result is compared with the result of the previous iteration, by an inspection of the terminal nodes of D_{k+1} which represents the difference between F_k and F_{k+1}. The iteration is finished if either the indicated maximum number of iterations is reached, or the tolerance of an "acceptable" approximation results.

Example 5. Consider our running CTMC-example and check $s_1 \models \mathcal{P}_{\leqslant 0.8}(a\,\mathcal{U}^{\leqslant 2}\,b)$. We assume that N equals 4 and adopt the trapezoidal method for numerical integration. The MTBDD Q is the same as in Example 4. In the first iteration we obtain for F_1 a single state vertex v labelled s_1 with $child_1(v) = \mathbf{1}$, the terminal vertex labelled 1. In the second iteration we obtain the MTDD F_2:



where $n_1 = \frac{1}{2} \cdot e^{-0} + \frac{1}{2} \cdot e^{-0.5}$, $n_2 = \frac{1}{4} \cdot e^{-0} + \frac{1}{2} \cdot e^{-0.5} + \frac{1}{4} \cdot e^{-1}$, $n_3 = \frac{1}{4} \cdot e^{-0} + \frac{1}{2} \cdot (e^{-0.5} + e^{-1}) + \frac{1}{4} \cdot e^{-1.5}$, and $n_4 = \frac{1}{4} \cdot e^{-0} + \frac{1}{2} \cdot (e^{-0.5} + e^{-1} + e^{-1.5}) + \frac{1}{4} \cdot e^{-2}$ The third iteration reveals that $\mathsf{F}_3 = \mathsf{F}_2$, so the algorithm finishes and returns $\mathsf{R} = \textsc{Restrict}(\mathsf{F}_3, \mathsf{x}, 2)$, obtained from F_3 by replacing the subgraph starting in vertex x by the terminal vertex with label $n_4 \approx 0.882604$. Finally, $\textsc{Compare}(\mathsf{R}, I_{\leqslant 0.8})$ reveals that $\mathcal{P}_{\leqslant 0.8}(a\,\mathcal{U}^{\leqslant 2}\,b)$ is indeed violated in s_1. Increasing the number of abscissas increases the accuracy: e.g. $N = 64$ leads to 0.8647350 as an approximation for $1 - e^{-2}$.

4 Concluding remarks

In this paper we have generalised MTBDDs to multi-terminal decision diagrams, in order to achieve a data structure for representing real-valued functions whose arguments are boolean or real variables. These structures are suitable for numerical integration using quadrature formulas, like trapezoidal, Simpson and Romberg integration schemes. The usefulness of MTDDs has been exemplified for symbolic model checking of continuous-time Markov chains.

Due to their suitability for numerical integration, the potential applicability of MTDDs is much wider than model checking of CTMCs, or more general stochastic processes. In order to provide evidence about the adequacy of our approach, an important direction for future research is the implementation of an MTDD-tool, together with the proposed verification algorithm for CTMCs that should .

Acknowledgement. We thank Markus Siegle for discussion about our initial ideas concerning MTDDs. Ed Brinksma has provided valuable comments on an earlier version of this paper.

References

1. R. Alur, C. Courcoubetis and D. Dill. Model-checking for probabilistic real-time systems. In *Automata, Languages and Programming*, LNCS 510, pp. 115–127, 1991.
2. A. Aziz, K. Sanwal, V. Singhal and R. Brayton. Verifying continuous time Markov chains. In *Computer-Aided Verification*, LNCS 1102, pp. 269–276, 1996.
3. A. Aziz, V. Singhal, F. Balarin, R. Brayton and A. Sangiovanni-Vincentelli. It usually works: the temporal logic of stochastic systems. In *Computer-Aided Verification*, LNCS 939, pp. 155–165, 1995.
4. I. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Padro and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in Systems Design*, **10**(2/3): 171–206, 1997.
5. C. Baier. On algorithmic verification methods for probabilistic systems. Habilitation thesis (submitted), Univ. Mannheim, 1998.
6. C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Automata, Languages and Programming*, LNCS 1256, pp. 430-440, 1997.
7. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching-time logic with fairness. *Distr. Comp.*, **11**(3), 1998.
8. D. Beauquier and A. Slissenko. Polytime model checking for timed probabilistic computation tree logic. *Acta Inf.*, **35**: 645–664, 1998.
9. J. Bern, C. Meinel, A. Slobodova: Global Rebuilding of OBDDs Avoiding Memory Requirement Maxima, Proc. CAV'95, LNCS 939, pp 4-15, 1995.
10. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Found. of Softw. Technology and Theor. Comp. Sci.*, LNCS 1026, pp. 499–513, 1995.
11. R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Comp.*, **C-35**(8): 677-691, 1986.
12. R. Bryant, Y. Chen: Verification of Arithmetic Functions with Binary Moment Diagrams, Proc. 32nd ACM/IEEE Design Automation Conference, pp 535-541, 1995.

13. W. Chan, R. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin and J.D. Reese. Model checking large software specifications. *IEEE Trans. on Softw. Eng.*, **24**(7): 498–519, 1998.
14. I. Christoff and L. Christoff. Reasoning about safety and liveness properties for probabilistic systems. In *Found. of Softw. Technology and Theor. Comp. Sci.*, LNCS 652, pp 342-355, 1992.
15. E. Clarke, M. Fujita, P. McGeer, J. Yang and X. Zhao. Multi-terminal binary decision diagrams: an efficient data structure for matrix representation. In *Proc. IEEE Int. Workshop on Logic Synthesis*, pp. 1–15, 1993.
16. E. Clarke, M. Fujita, X. Zhao: Multi-Terminal Binary Decision Diagrams and Hybrid Decision Diagrams, In *Representations of Discrete Functions*, T. Sasao and M. Fujita (eds.), Kluwer Academic Publishers, pp 93-108, 1996.
17. E. Clarke, O. Grumberg and D. Long. Verification tools for finite-state concurrent programs. In *A Decade of Concurrency*, LNCS 803, pp. 124–175, 1993.
18. C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. Symp. on Foundations of Computer Science*, pp. 338–345, 1988.
19. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, **42**(4): 857–907, 1995.
20. L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proc. Symp. on Logic in Computer Science*, 1998.
21. L. de Alfaro. Stochastic transition systems. In *Concurrency Theory*, LNCS 1466, pp. 423–438, 1998.
22. M. Fujita, Y. Matsunaga, T. Kakadu: On Variable Ordering of Binary Decision Diagrams for the Application of Multi-Valued Logic Synthesis, Proc. EDAC'91, pp 50-53, 1991.
23. M. Fujita, P. McGeer, J. Yang: Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation, *Formal Methods in System Design*, Vol. 10, No. 2/3, pp 149-170, 1997.
24. G. Hachtel, E. Macii, A. Padro and F. Somenzi. Markovian analysis of large finite-state machines. *IEEE Trans. on Comp. Aided Design of Integr. Circ. and Sys.*, **15**(12): 1479–1493, 1996.
25. H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Form. Asp. of Comp.*, **6**: 512–535, 1994.
26. V. Hartonas-Garmhausen. *Probabilistic Symbolic Model Checking with Engineering Models and Applications.* PhD. thesis, Carnegie Mellon University, 1998.
27. H. Hermanns. *Interactive Markov Chains.* Ph.D thesis, U. Erlangen-Nürnberg, 1998.
28. H. Hermanns and J.-P. Katoen. Automated compositional Markov chain generation for a plain-old telephone system. *Sci. of Comp. Programming*, 1999 (to appear).
29. Y. Lai, M. Pedram, B. Vrudhula: Edge-Valued Binary Decision Diagrams for Integer Linear Programming, Spectral Transformation, and Function Decomposition, *IEEE Transactions on CAD*, Vol. 13, No. 8, pp 959-975, 1994.
30. A. Pnueli, L. Zuck. Probabilistic verification. *Inf. and Comp.*, **103,**: 1–29, 1993.
31. W. Press, B. Flannery, S. Teukolsky and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge Univ. Press, 1989.
32. R. Rudell: Dynamic Variable Ordering for Ordered Binary Decision Diagrams, Proc. IEEE ICCAD'93, pp 42-47, 1993.
33. T. Sasao, M. Fujita: Representations of Discrete Functions, Kluwer Academic Publishers, 1996.
34. W. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.
35. K.S. Trivedi, J.K. Muppala, S.P. Woolet, and B.R. Haverkort. Composite Performance and Dependability Analysis. *Performance Evaluation*, **14**: 197–215, 1992.
36. M.Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. Symp. on Foundations of Computer Science*, pages 327–338, 1985.
37. X. Zhao: Verification of Arithmethic Circuits, Ph.D.Thesis, Carnegie Mellon University, 1996.

This article was processed using the LaTeX macro package with LLNCS style