Computing Quantiles in Markov Chains with Multi-Dimensional Costs

Christoph Haase
University of Oxford, UK

Stefan Kiefer
University of Oxford, UK

Markus Lohrey
Universität Siegen, Germany

***Abstract*—Probabilistic systems that accumulate quantities such as energy or cost are naturally modelled by cost chains, which are Markov chains whose transitions are labelled with a vector of numerical costs. Computing information on the probability distribution of the total accumulated cost is a fundamental problem in this model. In this paper, we study the so-called cost problem, which is to compute quantiles of the total cost, such as the median cost or the probability of large costs. While it is an open problem whether such probabilities are always computable or even rational, we present an algorithm that allows to approximate the probabilities with arbitrary precision. The algorithm is simple to state and implement, and exploits strong results from graph theory such as the so-called BEST theorem for efficiently computing the number of Eulerian circuits in a directed graph. Moreover, our algorithm enables us to show that a decision version of the cost problem lies in the counting hierarchy, a counting analogue to the polynomial-time hierarchy that contains the latter and is included in PSPACE. Finally, we demonstrate the applicability of our algorithm by evaluating it experimentally.**

I. Introduction

Stochastic uncertainty is an unavoidable feature of many systems and inherently present in probabilistic programs. As a consequence, their quantitative analysis, in particular performance analysis, requires formal models of probabilistic behaviour. *Markov chains*, a well-established and versatile mathematical concept, are at the core of such models. A Markov chain comprises a set of states with a transition function that assigns to every state a probability distribution over the set of successor states. A typical problem about a given Markov chain, with applications, e.g., to system reliability, is computing the probability with which a designated target state is reached starting from an initial state. This probability is computable in polynomial time [6] for explicitly given Markov chains. Polynomial-time decidability carries over to more complex specifications of probabilistic systems, such as properties specified in PCTL [6], a stochastic extension of the branching-time logic CTL. Probabilistic model checkers such as PRISM [25] and MRMC [23] can efficiently reason about such properties on large Markov chains in practice.

Cost chains. Reasoning about the performance of probabilistic programs and systems requires expressive modelling languages that incorporate not only stochasticity but further quantitative measures such as time, energy, profit, etc. We refer to those measures as costs. A cost chain is a Markov chain whose transitions are labelled by a numerical cost, or more generally by a vector of costs. We are primarily interested in the total cost accumulated during a run of the Markov chain. For an example, we consider the so-called coupon collector's problem, see e.g. [28]. Suppose a vendor of cereals launches a promotion campaign and adds a sticker with the photograph of a football player of a popular team into every box of cereals uniformly at random. A consumer wins a prize once she collects the stickers of every member of the team. For n team members, this process can be modelled by an n-dimensional cost chain, where the ith dimension models the ith distinct football player found in the cereal boxes.

Figure I illustrates this cost chain for $n = 4$. There, $\boldsymbol{e}_i$ is the ith unit vector. Starting in control state 1, the cost chain first transitions into control state 2 and adds the cost vector $\boldsymbol{e}_1$, since we certainly discover one new coupon at the beginning. Then, in control state 2, there are two possibilities: either with probability 0.25 we draw again the first coupon, i.e., loop in control state 2 while adding $\boldsymbol{e}_1$; otherwise we find the second coupon and transition to control state 3 while adding $\boldsymbol{e}_2$. Continuing analogously, we eventually reach the target control state 4 in which every component of the vector is at least one. The components indicate how often the n players have been discovered.

The cost problem. A tailor-made analysis of the coupon collector's problem shows that the expected termination time is $\Theta(n \cdot \log n)$, see e.g. [22], [28]. Given an arbitrary cost chain, one can compute the expected cost (or the vector of expected costs in the multi-dimensional case) in polynomial time [6]. However, this provides only little information about the (joint) probability distribution of the costs. For example, computing expectations does not suffice to answer questions such as "what is the probability that after collecting all coupons, there is some coupon that has been drawn at least three times?" The problem considered in the previous example is an instance of the more general *cost problem*: Given a cost chain, a probability threshold τ, and a *cost formula* which is a Boolean combination $\varphi(x_1, \ldots, x_n)$ of linear inequalities over n variables, the cost problem asks whether the accumulated probability of all paths achieving a value consistent with φ when reaching a target state t is at least τ. With an algorithm for the cost problem at hand, one can compute further information on the probability distribution of the costs, specifically quantiles, see Section III.

978-1-5090-3018-7/17/$31.00

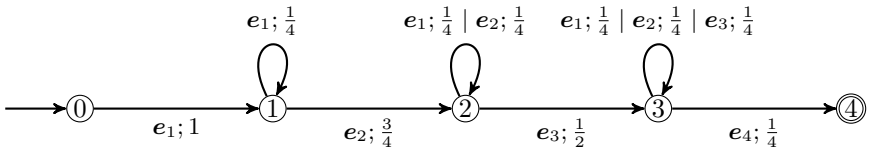

Fig. 1. Cost chain $\mathcal{C}_n$ modelling the coupon collector's problem for $n = 4$. The loop labelled with $\boldsymbol{e}_1; \frac{1}{4} \mid \boldsymbol{e}_2; \frac{1}{4}$ represents two loops, labelled with $\boldsymbol{e}_1; \frac{1}{4}$ and $\boldsymbol{e}_2; \frac{1}{4}$, respectively (and similarly for the right loop).

It has been shown in [18] that the *non-negative* cost problem (where all costs are non-negative) can be decided in PSPACE for one-dimensional cost chains. This follows from an application of a general result from [26], [33] that states equality of the complexity classes PSPACE and probabilistic PSPACE. From this, one could in principle glean a PSPACE algorithm for the non-negative cost problem by counting the number of accepting computations of a polynomial-space Turing-machine simulating the cost chain. It is questionable whether such an algorithm would be applicable in practice. The computational difficulty of the non-negative cost problem can be demonstrated by a lower complexity bound: it was shown in [18] that it is hard for the complexity class PP, the counting analogue of NP. In fact, PP is substantially harder than NP: it follows from Toda's theorem [34] that, with the help of an oracle for the cost problem, one can decide *any* problem in the polynomial-time hierarchy in deterministic polynomial time.

Our contribution. While the decidability of the general cost problem remains an open problem of this paper, we make the following contributions:

(1) We show that the probability of all paths reaching the target state with accumulated cost consistent with a cost formula can be approximated arbitrarily closely. This leads us to study the *finitary* cost problem, which is a subclass of the cost problem and a superclass of the non-negative cost problem.
(2) We provide a graph-based algorithm for deciding the finitary cost problem.
(3) Building upon this algorithm we show that the finitary cost problem can be decided in the counting hierarchy.
(4) We show that a prototypical implementation of our algorithm can outperform the state-of-the-art model checker PRISM [25].

The *counting hierarchy (*CH*)* is the counting analogue of (and actually contains) the polynomial-time hierarchy, see e.g. [4], [36] or Section IV. It is contained in PSPACE. Thus, our contribution (3) implies that this version of the cost problem is not PSPACE-hard unless PSPACE = CH. In recent years, several numerical problems, for which only PSPACE upper bounds had been known, have been shown to be in CH. Two of the most important and fundamental problems of this kind are PosSLP and BitSLP: PosSLP is the problem whether a given arithmetic circuit over the operations $+$, $-$ and $\times$ evaluates to a positive number, and BitSLP asks whether a certain bit of the computed number is equal to 1. Even placing those problems in PSPACE is non-trivial, since an arithmetic circuit with n gates can evaluate to a number in the order of 2^{2^n}; hence the number of output bits can be exponential and a certain bit of the output number can be specified with polynomially many bits. Placing the finitary cost problem in CH additionally requires us to make the following technical contributions:

(5) We develop a toolbox for showing membership of numerical problems in the counting hierarchy, based on techniques introduced in [3], [21].
(6) We show that the number of words with a given Parikh image accepted by a deterministic finite-state automaton can be computed in CH.

The latter result makes use of the so-called BEST theorem for efficiently counting the number of Eulerian paths in directed graphs [1, p. 445]. As a corollary, we derive from those results that the non-negative cost problem considered in [18] is in CH, improving the PSPACE upper bound from [18].

The BEST theorem enables us to summarise a potentially exponential number of paths that lead to the same cost vector. One can thus view the techniques developed in our paper as a partial order reduction technique for probabilistic systems. We elaborate on this aspect in some more detail in Section V-A.

Related work. The problems studied in this paper lie at the intersection of probabilistic verification, automata theory, enumerative combinatorics and computational complexity. Hence, there is a large body of related work that we can only briefly discuss here.

Over the last decade and in particular in recent years, there has been strong interest in extensions of Markov chains and Markov decision processes (MDPs) with (multi-dimensional) weights. Laroussinie and Sproston were the first to show that model checking PCTL on cost chains with non-negative costs in dimension one is NP-hard and in EXPTIME [27]; the lower bound has recently been improved to PP in [19]. Qualitative aspects of quantile problems in weighted MDPs where the probability threshold τ is either 0 or 1 have been studied by Ummels and Baier in [37], and iterative linear programming based approaches for solving reward quantiles have been described in [5], [24]. There is also a large body of work on synthesising strategies for weighted MDPs that ensure both worst case as well as expected value guarantees [12], [14], [15], [30]; see [31] for a survey on such beyond-worst-case-analysis problems. Another branch of related work deals

with extension of classical temporal logics that allow to reason about accumulated costs along runs, see e.g. [7], [10], [16], [35].

We use the BEST theorem in our proof for membership of the finitary cost problem in CH in order to count the number of Eulerian circuits in a directed edge-weighted multi-graph whose edge weights are encoded in binary. For an unweighted directed graph, the BEST theorem allows one to compute the number of Eulerian circuits in polynomial time (even in NC^2), since it basically reduces the computation to a determinant. On the other hand, computing the number of Eulerian circuits in an undirected graph is #P-complete [11]. For graphs of bounded tree width, the number of Eulerian circuits can be computed very efficiently: in logspace for directed graphs [8], and in NC^2 for undirected graphs [9].

Outline of the Paper. Section II contains all basic definitions. Subsequently, Section III establishes our contributions (1) and (2): we define the (finitary) cost problem and describe our basic algorithm for deciding it by a reduction to counting problems on edge-weighted graphs. We also explain how the cost problem relates to the computation of quantiles. In Section IV we deliver contribution (3) by proving that the finitary cost problem can be decided in the counting hierarchy. The technical contributions (5) and (6) are contained in Sections IV-C and IV-D. Finally, in Section V we evaluate our basic algorithm experimentally (contribution (4)). We conclude in Section VI.

II. Preliminaries

By $\mathbb{Z}$ and $\mathbb{N}$ we denote the set of integers and non-negative integers, respectively. Let $\Sigma = \{a_1, \dots, a_m\}$ be a finite alphabet. A Parikh vector is a vector of m non-negative integers, i.e., an element of $\mathbb{N}^m$ or alternatively $\mathbb{N}^\Sigma$. Let $u \in \Sigma^*$ be a word. For $a \in \Sigma$, we denote by $|u|_a$ the number of occurrences of a in u. The Parikh image $\Psi(u) \in \mathbb{N}^\Sigma$ of u is the Parikh vector counting how often every alphabet symbol of Σ occurs in u, i.e., $\Psi(u) := (|u|_{a_1}, \dots, |u|_{a_m})$. The Parikh image of a language $L \subseteq \Sigma^*$ is defined as $\Psi(L) := \{\Psi(u) : u \in L\} \subseteq \mathbb{N}^\Sigma$.

A deterministic finite-state automaton (DFA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$, where Q is a finite set of control states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\delta \subseteq Q \times \Sigma \times Q$ is a finite set of transitions such that $(q, a, q_1), (q, a, q_2) \in \delta$ implies $q_1 = q_2$. Given $u = a_1 a_2 \cdots a_n \in \Sigma^*$, a run ϱ of $\mathcal{A}$ on u is a finite sequence of control states $\varrho = p_0 p_1 \cdots p_n$ such that $p_0 = q_0$ and $(p_i, a_i, p_{i+1}) \in \delta$ for all $0 \leq i < n$. We call ϱ accepting whenever $p_n \in F$ and define the language accepted by $\mathcal{A}$ as $L(\mathcal{A}) := \{u \in \Sigma^* : \mathcal{A} \text{ has an accepting run on } u\}$.

Given a vector $\boldsymbol{z} = (z_1, \dots, z_d) \in \mathbb{Z}^d$, we denote by $\|\boldsymbol{z}\|_1$ and $\|\boldsymbol{z}\|_\infty$ the sum and the maximum norm, which are defined as usual, i.e., $\|\boldsymbol{z}\|_1 := \sum_{1 \leq i \leq d} |z_i|$ and $\|\boldsymbol{z}\|_\infty := \max_{1 \leq i \leq d} |z_i|$. Given a $d \times n$ integer matrix A consisting of row vectors $\boldsymbol{z}_1, \dots, \boldsymbol{z}_d$, we define $\|A\|_{1,\infty} := \max_{1 \leq i \leq d} \|\boldsymbol{z}_i\|_1$.

III. Cost Problems in Markov Chains

A *Markov chain* is a triple $\mathcal{M} = (S, s_0, \delta)$, where S is a countable (finite or infinite) set of states, $s_0 \in S$ is the initial state, and $\delta \colon S \to \mathit{dist}(S)$ is a probabilistic transition function that maps a state to a probability distribution over the successor states. Given a Markov chain we also write $s \xrightarrow{p} t$ or $s \to t$ to indicate that $p = \delta(s)(t) > 0$. A *run* is an infinite sequence $s_0 s_1 s_2 \cdots$ where $s_i \in S$ and $s_{i-1} \to s_i$ for all $i \geq 1$. We write $\mathit{Run}(s_0 \cdots s_k)$ for the set of runs that start with $s_0 \cdots s_k$. We associate to $\mathcal{M}$ the standard probability space $(\mathit{Run}(s_0), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-algebra generated by all basic cylinders $\mathit{Run}(s_0 \cdots s_k)$ with $s_0 \cdots s_k \in s_0 S^*$, and $\mathcal{P} \colon \mathcal{F} \to [0, 1]$ is the unique probability measure such that

$$\mathcal{P}(\mathit{Run}(s_0 \cdots s_k)) = \prod_{i=1}^{k} \delta(s_{i-1})(s_i).$$

A *cost chain in dimension* d is a tuple $\mathcal{C} = (Q, q_0, t, \Delta)$, where Q is a finite set of control states, $q_0 \in Q$ is the initial control state, $t \in Q \setminus \{q_0\}$ is the target control state, and $\Delta \colon Q \to \mathit{dist}(Q \times \mathbb{Z}^d)$ is a probabilistic transition function. Here, for $q, q' \in Q$ and $\boldsymbol{z} \in \mathbb{Z}^d$, when q is the current control state, the value $\Delta(q)(q', \boldsymbol{z}) \in [0, 1]$ is the probability that the cost chain transitions to control state q' and cost $\boldsymbol{z}$ is incurred. (Rational numbers as costs could be made integer by multiplying with the least common multiple of the denominators.) For complexity results we define the *size* of $\mathcal{C}$ as the size of a succinct description, i.e., the costs are encoded in binary, the probabilities are encoded as fractions of integers in binary (so the probabilities are assumed to be rational in order to get a finite representation), and for each $q \in Q$, the distribution $\Delta(q)$ is described by the list of triples $(q', \boldsymbol{z}, p)$ with $\Delta(q)(q', \boldsymbol{z}) = p > 0$ (so we assume this list to be finite). We define the set E of edges of $\mathcal{C}$ as $E := \{(q, \boldsymbol{z}, q') : \Delta(q)(q', \boldsymbol{z}) > 0\}$, and write $\Delta(e)$ for $\Delta(q)(q', \boldsymbol{z})$ when $e = (q, \boldsymbol{z}, q') \in E$. A cost chain $\mathcal{C}$ induces a Markov chain $\mathcal{M}_\mathcal{C} = (Q \times \mathbb{Z}^d, (q_0, \mathbf{0}), \delta)$ with $\delta(q, \boldsymbol{c})(q', \boldsymbol{c}') = \Delta(q)(q', \boldsymbol{c}' - \boldsymbol{c})$ for all $q, q' \in Q$ and $\boldsymbol{c}, \boldsymbol{c}' \in \mathbb{Z}^d$. For a state $(q, \boldsymbol{c}) \in Q \times \mathbb{Z}^d$ in $\mathcal{M}_\mathcal{C}$ we view q as the current control state and $\boldsymbol{c}$ as the current cost, i.e., the cost accumulated so far. We sometimes call the components of $\boldsymbol{c}$ and their values counters and counter values, respectively.

In this section, we will be interested in the cost accumulated during a run when reaching the target state t. Following [18], we assume (i) that the target state t is almost surely reached, and (ii) that $\Delta(t)(t, \mathbf{0}) = 1$. Hence runs that visit t do not leave t and accumulate only a finite cost. Those assumptions are needed for the following definition to be sound[1]: Given a cost chain $\mathcal{C}$, we define a random variable $K_\mathcal{C}$ that maps a run of $\mathcal{M}_\mathcal{C}$ to its accumulated cost. Formally, we define $K_\mathcal{C} \colon \mathit{Run}((q_0, \mathbf{0})) \to \mathbb{Z}^d$ such that $K_\mathcal{C}((q_0, \mathbf{0})\,(q_1, \boldsymbol{c}_1) \cdots) = \boldsymbol{c}$ if there exists $i \in \mathbb{N}$ with $(q_i, \boldsymbol{c}_i) = (t, \boldsymbol{c})$. From the aforementioned assumptions on t, it follows that the random variable $K_\mathcal{C}$ is almost surely defined.

[1] See [18] for a discussion on why those assumptions can be made.

A cost chain $\mathcal{C}$ induces a DFA $\mathcal{A}_\mathcal{C} := (Q, E, q_0, \{t\}, \delta)$, which we define such that $(q, e, q') \in \delta$ whenever $e = (q, \boldsymbol{z}, q') \in E$ for some $\boldsymbol{z} \in \mathbb{Z}^d$, except for t which has no outgoing transitions. Note that different transitions are labelled with different alphabet symbols. For $u \in L(\mathcal{A}_\mathcal{C})$, we define $\widehat{Run}(u)$ to be the unique run of $\mathcal{C}$ that coincides with u in the order of the traversed edges in its initial segment.

Let $x_1, \ldots, x_d$ be d integer-valued variables. An *atomic cost formula* is an inequality of the form $a_1 \cdot x_1 + \cdots + a_d \cdot x_d \leq b$, where $a_1, \ldots, a_d, b \in \mathbb{Z}$ are encoded in binary, and a *cost formula* is an arbitrary Boolean combination of atomic cost formulas, i.e., a formula in quantifier-free Presburger arithmetic. We say that a tuple $\boldsymbol{c} = (c_1, \ldots, c_d) \in \mathbb{Z}^d$ *satisfies* a cost formula φ, in symbols $\boldsymbol{c} \models \varphi$, if φ is true when every x_i is replaced by c_i. Let

$$[\![\varphi]\!] := \{\boldsymbol{c} \in \mathbb{Z}^d : \boldsymbol{c} \models \varphi\}$$

be the set of all satisfying assignments of φ.

We can now formally define the problem that we study in this paper:

COST PROBLEM

INPUT: A cost chain $\mathcal{C}$, a cost formula φ and a probability threshold $\tau \in [0, 1]$ given as a fraction of integers encoded in binary.

QUESTION: Does $\mathcal{P}(K_\mathcal{C} \models \varphi) \geq \tau$ hold?

This generalises the definition from [18], where the cost problem for one-dimensional cost chains with non-negative costs is studied. Consequently φ is required to only define a subset of $\mathbb{N}$. It is shown in [18] that this version of the cost problem belongs to PSPACE. This result is not algorithmic in its nature and relies on a nontrivial result from computational complexity, namely the equality PSPACE = #PSPACE [26]. Moreover, the decidability result from [18] depends on the fact that due to the imposed restrictions, $[\![\varphi]\!]$ is either finite or co-finite, and hence it suffices to consider only a finite number of paths in the induced Markov chain in order to decide a cost problem. If we allow negative costs to occur in cost chains, it is not obvious (and the authors do not know) whether the cost problem is decidable, even in dimension one. It is not even clear whether the probability of reaching a particular target configuration is rational. Therefore, we consider a special case of the cost problem to obtain decidability and good complexity bounds.

Computation of Quantiles. The cost problem generalises the computation of quantiles of the probability distribution of accumulated costs. In more detail, in the one-dimensional case, one might be interested in finding the smallest budget b so that the probability of the cost being at most b is at least τ. Given b, it is an instance of the cost problem to check whether $\mathcal{P}(K_\mathcal{C} \models x_1 \leq b) \geq \tau$. Binary search can be used to determine the least b that satisfies this probability bound. In the multidimensional case one might fix $b_1, \ldots, b_{n-1}$ and use binary search on b_n to find the least b_n such that $\mathcal{P}(K_\mathcal{C} \models \bigwedge_{i=1}^n x_i \leq b_i) \geq \tau$.

A. The Finitary Cost Problem

Let $\mathcal{C}$, φ, τ be an input instance of the cost problem with its induced DFA $\mathcal{A}_\mathcal{C}$. This instance is *finitary* if the set

$$L_\varphi(\mathcal{A}_\mathcal{C}) := \{u \in L(\mathcal{A}_\mathcal{C}) : K_\mathcal{C}(\widehat{Run}(u)) \models \varphi\}$$

is finite. If $L_{\neg\varphi}(\mathcal{A}_\mathcal{C})$ is finite, we call the instance *co-finitary*. We will later argue that one can decide whether an instance of the cost problem is finitary (or co-finitary), see Proposition 3 below. Clearly, an algorithm for the cost problem restricted to finitary instances also solves the co-finitary case, since $\mathcal{P}(K_\mathcal{C} \models \varphi) = 1 - \mathcal{P}(K_\mathcal{C} \models \neg\varphi)$. Moreover, an arbitrary instance of a cost problem in dimension d can be turned into a finitary cost problem given by $\hat{\mathcal{C}}$, $\hat{\varphi}$ and τ by (i) adding to $\mathcal{C}$ a new counter that gets incremented along every transition, except for the self-loop at the target state t, in order to obtain a cost chain $\hat{\mathcal{C}}$; and (ii) replacing φ with $\hat{\varphi} := \varphi \wedge 0 \leq x_{d+1} \leq m$ for some arbitrary but fixed *length threshold* $m \in \mathbb{N}$ representing the maximum length of paths allowed before reaching the target t. This transformation gives up the infinite-state nature of the cost problem, but it allows for an arbitrarily close approximation:

Proposition 1. *Let $\hat{\mathcal{C}}$, $\hat{\varphi}$, τ be the instance of the finitary cost problem obtained from restricting the instance $\mathcal{C}$, φ, τ of the cost problem to some length threshold m, and let $p = \mathcal{P}(K_\mathcal{C} \models \varphi)$. Then $p - \epsilon \leq \mathcal{P}(K_{\hat{\mathcal{C}}} \models \hat{\varphi}) \leq p$ for*

$$\epsilon = \exp\left(p_{\min}^{|Q|} \cdot \left(-\frac{m-1}{|Q|} + 1\right)\right),$$

where $p_{\min}$ is the smallest nonzero probability appearing in the description of $\mathcal{C}$.

Proof. We have:

$$\begin{aligned}
\mathcal{P}(K_{\hat{\mathcal{C}}} \models \hat{\varphi}) &\leq \mathcal{P}(K_{\hat{\mathcal{C}}} \models \varphi) && \text{def. of } \hat{\varphi} \\
&= \mathcal{P}(K_\mathcal{C} \models \varphi) && \text{def. of } \hat{\mathcal{C}} \\
&= p && \text{def. of } p
\end{aligned}$$

This proves the upper bound. Towards the lower bound, from the definition of ϵ we obtain:

$$m = |Q| \cdot \left(-\frac{\ln \epsilon}{p_{\min}^{|Q|}} + 1\right) + 1$$

From [18, Prop. 2] we have:

$$\mathcal{P}(K_{\hat{\mathcal{C}}} \models x_{d+1} > m) \leq \epsilon \tag{1}$$

Hence:

$$\begin{aligned}
p &= \mathcal{P}(K_\mathcal{C} \models \varphi) && \text{def. of } p \\
&= \mathcal{P}(K_{\hat{\mathcal{C}}} \models \varphi) && \text{def. of } \hat{\mathcal{C}} \\
&= \mathcal{P}(K_{\hat{\mathcal{C}}} \models \underbrace{\varphi \wedge 0 \leq x_{d+1} \leq m}_{\hat{\varphi}}) \\
&\quad + \mathcal{P}(K_{\hat{\mathcal{C}}} \models \varphi \wedge x_{d+1} > m) \\
&\leq \mathcal{P}(K_{\hat{\mathcal{C}}} \models \hat{\varphi}) + \mathcal{P}(K_{\hat{\mathcal{C}}} \models x_{d+1} > m) \\
&\leq \mathcal{P}(K_{\hat{\mathcal{C}}} \models \hat{\varphi}) + \epsilon && \text{by (1)}
\end{aligned}$$

This proves the proposition. □

Note that ϵ in Proposition 1 is independent from the integer weights occurring in $\mathcal{C}$.

B. A Parikh-Image Based Approach to the Cost Problem

We can use Parikh images as an abstraction for runs of a cost chain $\mathcal{C}$ and words in $\mathcal{A}_\mathcal{C}$. This in turn enables us to prove an upper bound on the length of the relevant part of runs satisfying φ. Let $E = \{e_1, \ldots, e_m\}$ be the set of edges of $\mathcal{C}$ and let $\boldsymbol{p} = (p_1, \ldots, p_m)$ be first-order variables representing a Parikh vector. It is shown in [32, Thm. 1] that from a given NFA $\mathcal{A}$ one can construct in linear time an existential Presburger formula of size $O(|\mathcal{A}|)$ for the Parikh image $\Psi(L(\mathcal{A}))$. It follows that there exists an existential Presburger formula $\gamma_{\mathcal{C},\varphi}(\boldsymbol{p})$ of size $O(|\mathcal{C}| + |\varphi|)$ that defines the Parikh image of $L_\varphi(\mathcal{A}_\mathcal{C})$:

$$[\![\gamma_{\mathcal{C},\varphi}]\!] = \Psi(L_\varphi(\mathcal{A}_\mathcal{C})). \tag{2}$$

While the formula $\gamma_{\mathcal{C},\varphi}$ is not explicitly presented in [32], it can easily be explained on an informal level since it essentially encodes the classical Euler-Hierholzer theorem in Presburger arithmetic. This theorem states that there exists a Eulerian path (a path traversing every edge exactly once) in a directed graph if and only if

(a) for every node, the number of incoming edges equals the number of outgoing edges, and
(b) the graph is connected.

Condition (a) is easily encoded as a system of linear equalities, and for Condition (b) the authors use a neat trick that simulates a breadth-first search in Presburger arithmetic. By recording for a run in $\mathcal{A}_\mathcal{C}$ the number of times it traverses an edge, the Euler-Hierholzer theorem gives us sufficient and necessary conditions for ensuring that a Parikh vector has some corresponding run in $\mathcal{A}_\mathcal{C}$. See also [20] for a more elaborate discussion on the construction from [32].

The following lemma makes use of (2) and in particular implies that the finitary cost problem is decidable.

Lemma 2. *Let $\mathcal{C}$, φ, τ be an instance of the finitary cost problem. Then for every $u \in L_\varphi(\mathcal{A}_\mathcal{C})$ we have $|u| \leq 2^{O((|\mathcal{C}|+|\varphi|)^3)}$.*

Proof. Let $\gamma(\boldsymbol{p})$ be the formula $\gamma_{\mathcal{C},\varphi}$ from (2). Bringing $\gamma(\boldsymbol{p})$ into disjunctive normal form, we have that $\gamma(\boldsymbol{p})$ is equivalent to a disjunction of polyhedra each given by the Presburger formula $\theta(\boldsymbol{p}) = \exists \boldsymbol{x} : A \cdot \binom{\boldsymbol{p}}{\boldsymbol{x}} \geq \boldsymbol{b}$, where A is an $n_1 \times n_2$ matrix for some $n_1, n_2 \leq |\gamma|$ and $\boldsymbol{x}$ is an integer vector of length $n_2 - m$. It follows from a result of Pottier [29, Cor. 1] and the fact that we are dealing with a finitary instance of the cost problem that for any $\boldsymbol{p}$ with $\theta(\boldsymbol{p})$

$$\|\boldsymbol{p}\|_1 \leq (2 + \|A\|_{1,\infty} + \|\boldsymbol{b}\|_\infty)^{|\gamma|}.$$

Estimating $\|A\|_{1,\infty} \leq 2^{|\gamma|} \cdot |\gamma|$ and $\|\boldsymbol{b}\|_\infty \leq 2^{|\gamma|}$, we conclude that every $\boldsymbol{p} \in \mathbb{N}^m$ with $\gamma(\boldsymbol{p})$ satisfies

$$\|\boldsymbol{p}\|_1 \leq (2 + 2^{|\gamma|} \cdot |\gamma| + 2^{|\gamma|})^{|\gamma|} \leq 2^{O((|\mathcal{C}|+|\varphi|)^3)}.$$

Algorithm 1 Deciding an instance of a finitary cost problem $\mathcal{C}$, φ and τ.

```
p := 0
let γ_{C,φ}(p) be the formula from (2)
while γ_{C,φ}(p) is satisfiable and p < τ do
    choose p* ∈ [[γ_{C,φ}]]
    p := p + N(A_C, p*) · ∏_{e∈E} Δ(e)^{p*(e)}
    γ_{C,φ}(p) := γ_{C,φ}(p) ∧ p ≠ p*
end while
return p ≥ τ
```

Since $\|\boldsymbol{p}\|_1$ equals the length of all paths with Parikh image $\boldsymbol{p}$, the length of every $u \in L_\varphi(\mathcal{A}_\mathcal{C})$ is bounded by $\|\boldsymbol{p}\|_1$ and the statement follows. □

Moreover, Lemma 2 enables us to decide whether a given instance of the cost problem is finitary:

Proposition 3. *Given a cost chain $\mathcal{C}$ and a cost formula φ, one can check in* coNP *whether $L_\varphi(\mathcal{A}_\mathcal{C})$ is finite.*

Proof. Lemma 2 implies that there is a constant $c > 0$ such that the given instance $\mathcal{C}$, φ is finitary if and only if $|u| \leq 2^{c \cdot (|\mathcal{C}|+|\varphi|)^3}$ for every $u \in L_\varphi(\mathcal{A}_\mathcal{C})$. Consequently, in order to check whether $\mathcal{C}$, φ is not finitary it suffices to decide whether $\gamma_{\mathcal{C},\varphi}(\boldsymbol{p})$ has a solution such that $\|\boldsymbol{p}\|_1 > 2^{c \cdot (|\mathcal{C}|+|\varphi|)^3}$. For $\boldsymbol{p} = (p_1, \ldots, p_n)$, this is equivalent to deciding whether

$$\gamma_{\mathcal{C},\varphi} \wedge \sum_{1 \leq i \leq n} p_i > 2^{c \cdot (|\mathcal{C}|+|\varphi|)^3}$$

is satisfiable, which can be done in NP. □

Note that due to (2), we can use an SMT-solver to enumerate $\Psi(L_\varphi(\mathcal{A}_\mathcal{C}))$. However, a single Parikh vector in $[\![\gamma_{\mathcal{C},\varphi}]\!]$ may correspond to an exponential number of paths through the Markov chain. Here we use the following observation: we only need the *number* of those paths. Suppose we had access to an oracle $N(\mathcal{A}_\mathcal{C}, \boldsymbol{p})$ returning the number of words in $L(\mathcal{A}_\mathcal{C})$ with Parikh image $\boldsymbol{p}$. Then we could calculate the probability $\mathcal{P}(K_\mathcal{C} \models \varphi)$ without enumerating all paths:

$$\begin{aligned} \mathcal{P}(K_\mathcal{C} \models \varphi) &= \sum_{u \in L_\varphi(\mathcal{A}_\mathcal{C})} \mathcal{P}(\widehat{Run}(u)) \\ &= \sum_{\boldsymbol{p} \in \Psi(L_\varphi(\mathcal{A}_\mathcal{C}))} N(\mathcal{A}_\mathcal{C}, \boldsymbol{p}) \cdot \prod_{e \in E} \Delta(e)^{\boldsymbol{p}(e)}. \end{aligned} \tag{3}$$

This equation gives rise to Algorithm 1. It enumerates all Parikh images of words in $L_\varphi(\mathcal{A}_\mathcal{C})$ using the formula $\gamma_{\mathcal{C},\varphi}$ from (2), and accumulates the probability of all paths with every such Parikh image. The while-loop is guaranteed to terminate as the cost problem is finitary.

In the remainder of the section we show how to efficiently compute the number $N(\mathcal{A}_\mathcal{C}, \boldsymbol{p})$. To this end, we give a graph-theoretical expression for the number $N(\mathcal{A}_\mathcal{C}, \boldsymbol{p})$: we relate it to the number of Eulerian circuits in an edge-weighted multi-graph associated with $\mathcal{A}_\mathcal{C}$ and $\boldsymbol{p}$.

C. Multi-Graphs

A (finite directed) multi-graph is a tuple $G = (V, E, s, t)$, where V is a finite set of nodes, E is a finite set of edges, and the mapping $s\colon E \to V$ (resp., $t\colon E \to V$) assigns to each edge its source node (resp., target node). We always assume that there are no isolated nodes, i.e., $V = s(E) \cup t(E)$. A path (of length n) in G from u to v is a sequence of edges $e_1, e_2, \ldots, e_n$ such that $s(e_1) = u$, $t(e_n) = v$, and $t(e_i) = s(e_{i+1})$ for all $1 \leq i \leq n-1$. We say that G is connected if for all nodes $u, v \in V$ there exists a path in G from u to v. We call an edge $e \in E$ with $s(e) = t(e)$ a loop. The in-degree of v is $d^+_G(v) := \#t^{-1}(v)$ (the preimage $t^{-1}(v)$ is the set of all incoming edges for node v) and the out-degree of v is $d^-_G(v) := \#s^{-1}(v)$. We call G Eulerian if $d^-_G(v) = d^+_G(v)$ for all $v \in V$. A Eulerian circuit is a path $e_1, e_2, \ldots, e_n$ such that $E = \{e_1, e_2, \ldots, e_n\}$, $e_i \neq e_j$ for $i \neq j$, and $t(e_n) = s(e_1)$. Let us denote by $e(G)$ the number of Eulerian circuits of G, where we do not distinguish between the Eulerian circuits $e_1, e_2, \ldots, e_n$ and $e_i, e_{i+1}, \ldots, e_n, e_1, \ldots, e_{i-1}$. Alternatively, $e(G)$ is the number of Eulerian circuits that start with a distinguished edge. In other words, $e(G)$ counts the number of Eulerian circuits of G modulo cyclic rotations. Since we have no isolated nodes, we have $e(G) > 0$ if and only if G is Eulerian and connected.

D. The BEST Theorem and Tutte's Matrix-Tree Theorem

In this section we explain the two main tools from combinatorics that we use for the computation of $N(\mathcal{A}_\mathcal{C}, \boldsymbol{p})$: the BEST theorem and Tutte's matrix-tree theorem.

Let $G = (V, E, s, t)$ be a multi-graph. For $v \in V$, let $t(G, v)$ denote the number of directed spanning trees oriented towards v, i.e., the number of sub-graphs T of G such that

(i) T contains all nodes of G,
(ii) v has out-degree 0 in T, and
(iii) for every other vertex $u \in V \setminus \{v\}$ there is a unique path in T from u to v.

If G is Eulerian then $t(G, v) = t(G, v')$ for all $v, v' \in V$ [1, p. 236], and we denote this number with $t(G)$. The BEST theorem [1, p. 445], named after de Bruijn, van Aardenne-Ehrenfest, Smith and Tutte, relates the number $e(G)$ of Eulerian circuits with $t(G)$:

Theorem 4 (BEST theorem)**.** *If G is a connected Eulerian multi-graph, then $e(G) = t(G) \cdot \prod_{v \in V} (d^-_G(v) - 1)!$.*

The number $t(G)$ can be computed using Tutte's matrix-tree theorem, see e.g. [1, p. 231]. First, note that removing all loops from G does not influence the number of spanning trees. We can also assume that $V = \{1, \ldots, n\}$. The adjacency matrix of G is $A(G) = (a_{i,j})_{1 \leq i,j \leq n}$, where $a_{i,j} := \#\{e \in E : s(e) = i, t(e) = j\}$ is the number of edges from i to j. The out-degree matrix $D^-(G) = (d_{i,j})_{1 \leq i,j \leq n}$ is defined by $d_{i,j} := 0$ for $i \neq j$ and $d_{i,i} := d^-_G(i)$. The Laplacian $L(G)$ of G is $L(G) := D^-(G) - A(G)$. If M is a matrix then we denote by $M^{i,j}$ the (i, j)-minor of M, i.e., the matrix obtained from M by deleting its ith row and jth column.

Theorem 5 (Tutte's matrix-tree theorem)**.** *For a loop-free multigraph G we have $t(G, i) = (-1)^{i+j} \cdot \det(L(G)^{i,j})$, where $j \in V$ is arbitrary. In particular, $t(G, i) = \det(L(G)^{i,i})$.*

As mentioned previously, in the Eulerian case we have $t(G) = t(G, i) = t(G, j)$ for all $i, j \in V$, and thus $t(G) = \det(L(G)^{1,1})$.

E. Edge-Weighted Multi-Graphs

An edge-weighted multi-graph is a tuple $G = (V, E, s, t, w)$, where (V, E, s, t) is a multi-graph and $w\colon E \to \mathbb{N}$ assigns a weight to every edge. We can define the ordinary multi-graph $\tilde{G}$ induced by G by replacing every edge $e \in E$ by $k = w(e)$ many edges $e_1, \ldots, e_k$ with $s(e_i) = s(e)$ and $t(e_i) = t(e)$. For two nodes $u, v \in V$ with $u \neq v$ we denote by $e(G, u, v)$ the number of paths $e_1, e_2, \ldots, e_n$ in (V, E, s, t) such that $s(e_1) = u$, $t(e_n) = v$, and for every edge e, $w(e) = \#\{i : 1 \leq i \leq n, e = e_i\}$. To compute this number, let us define $\tilde{G}_{v,u}$ by adding to $\tilde{G}$ a new edge e with $s(e) = v$ and $t(e) = v$. With these definitions, we can show the following simple statement:

Lemma 6. *For an edge-weighted multi-graph $G = (V, E, s, t, w)$ and two nodes $u, v \in V$ with $u \neq v$ we have*

$$e(G, u, v) = \frac{e(\tilde{G}_{v,u})}{\prod_{e \in E} w(e)!}$$

Proof. Note that every path in G from u to v that uses every edge e exactly $w(e)$ times corresponds to exactly $\prod_{e \in E} w(e)!$ many Eulerian circuits of $\tilde{G}_{v,u}$. These Eulerian circuits are obtained by fixing for every $e \in E$ an arbitrary permutation of the $k = w(e)$ many edges $e_1, \ldots, e_k$ in $\tilde{G}$ that replace e. □

F. From Automata to Graphs

Let Σ be an alphabet and $\boldsymbol{p} \in \mathbb{N}^\Sigma$ be a Parikh vector. For a DFA $\mathcal{A}$, we denote by $N(\mathcal{A}, \boldsymbol{p})$ the number of words in $L(\mathcal{A})$ with Parikh image $\boldsymbol{p}$, i.e.,

$$N(\mathcal{A}, \boldsymbol{p}) := \#\{u \in L(\mathcal{A}) : \Psi(u) = \boldsymbol{p}\}.$$

Let us call a DFA well-formed if it has a unique final state that is different from the initial state and has no outgoing transitions. We can reduce the computation of $N(\mathcal{A}, \boldsymbol{p})$ to well-formed DFAs as follows: Let $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ be a DFA, and $\boldsymbol{p}$ a Parikh vector. We define a new DFA $\mathcal{A}'$ as follows: Add a fresh state $q_f \notin Q$ and a fresh symbol $b \notin \Sigma$ to the alphabet together with all transitions (q, b, q_f) for $q \in F$. Moreover, q_f becomes the only final state of $\mathcal{A}'$. Note that $L(\mathcal{A}') = L(\mathcal{A})b$. We extend the Parikh vector $\boldsymbol{p}$ to a Parikh vector $\boldsymbol{p}'$ by $\boldsymbol{p}'(b) = 1$ and $\boldsymbol{p}'(a) = \boldsymbol{p}(a)$ for all $a \neq b$. It follows that the mapping $x \mapsto xb$ ($x \in \Sigma^*$) induces a bijection between words $u \in L(\mathcal{A})$ with $\Psi(w) = \boldsymbol{p}$ and words $v \in L(\mathcal{A}')$ with $\Psi(v) = \boldsymbol{p}'$. Thus, we have $N(\mathcal{A}, \boldsymbol{p}) = N(\mathcal{A}', \boldsymbol{p}')$.

For a well-formed DFA $\mathcal{A} = (Q, \Sigma, q_0, \{q_f\}, \delta)$ and $\boldsymbol{p}\colon \Sigma \to \mathbb{N}$ let $W(\mathcal{A}, \boldsymbol{p})$ be the set of all mappings $w\colon \delta \to \mathbb{N}$ such that for every $a \in \Sigma$, we have $\boldsymbol{p}(a) = \sum_{(p,a,q) \in \delta} w(p, a, q)$. Moreover, for $w\colon \delta \to \mathbb{N}$ we define

the edge-weighted multi-graph $\mathcal{A}^w = (Q, \delta, s, t, w)$, where $s(p, a, q) = p$ and $t(p, a, q) = q$. With these definitions, we clearly get:

Lemma 7. *Let $\mathcal{A} = (Q, \Sigma, q_0, \{q_f\}, \delta)$ be a well-formed DFA. Then, $N(\mathcal{A}, \boldsymbol{p}) = \sum_{w \in W(\mathcal{A}, \boldsymbol{p})} e(\mathcal{A}^w, q_0, q_f)$.*

Let us summarise our reduction chain for the DFA $\mathcal{A}_\mathcal{C} = (Q, E, q_0, \{t\}, \delta)$ obtained from a cost chain $\mathcal{C} = (Q, q_0, t, \Delta)$. Note that $\mathcal{A}_\mathcal{C}$ is already well-formed and all transitions are labelled by pairwise different letters. The latter implies $W(\mathcal{A}_\mathcal{C}, \boldsymbol{p}) = \{w\}$, where $w(p, e, q) = \boldsymbol{p}(e)$ for all $(p, e, q) \in \delta$, and we get:

$$N(\mathcal{A}_\mathcal{C}, \boldsymbol{p}) = e(\mathcal{A}_\mathcal{C}^w, q_0, t). \tag{4}$$

From the edge-weighted multi-graph $G = \mathcal{A}_\mathcal{C}^w$, we form the multi-graph $G' = \tilde{G}_{t,q_0}$. W.l.o.g. we can assume that it has no isolated nodes. If G' is not connected or not Eulerian, then $e(G, q_0, t) = 0$, otherwise we get with Lemma 6

$$e(G, q_0, t) = \frac{e(G')}{\prod_{t \in \delta} w(t)!} = \frac{e(G')}{\prod_{e \in E} \boldsymbol{p}(e)!}.$$

Finally, for $e(G')$ we get with Theorems 4 and 5 the formula

$$\begin{aligned} e(G') &= t(G') \cdot \prod_{q \in Q} (d^-_{G'}(q) - 1)! \\ &= \det(L(G'')^{1,1}) \cdot \prod_{q \in Q} (d^-_{G'}(q) - 1)! \end{aligned}$$

(note that the new state q_f has outdegree one, and $0! = 1$). Here, G'' is obtained from G' by removing all loops. Our tool QUANT, see Section V, uses the above formulas to evaluate $N(\mathcal{A}_\mathcal{C}, \boldsymbol{p})$ in Algorithm 1.

IV. The Finitary Cost Problem Belongs to the Counting Hierarchy

In this section we use our approach based on the BEST theorem to show that the finitary cost problem is in the counting hierarchy (CH), which is defined in Section IV-A. Section IV-B contains further definitions and results used in the proofs. In Section IV-C we develop a toolbox for functions mapping bit strings to natural numbers that can be evaluated in CH. Here, evaluating a function refers to deciding whether a certain bit of the function value is equal to one. Our toolbox enables us to add, multiply and divide functions while staying inside CH. We apply these closure properties in Section IV-D to show that the previously defined function $N(\mathcal{A}, \boldsymbol{p})$ that counts the number of words with Parikh image $\boldsymbol{p}$ accepted by a DFA $\mathcal{A}$ can be evaluated in CH. This enables us to prove membership of the finitary cost problem in CH.

A. Definitions From Computational Complexity

We assume familiarity with basic complexity classes. The class PP (probabilistic polynomial time) contains all problems A for which there exists a non-deterministic polynomial-time Turing machine M such that for every input x, $x \in A$ if and only if more than half of all computation paths of M on input x are accepting. By a result of Toda [34], the polynomial time hierarchy (PH) is contained in P^{PP}, which is the class of all languages that can be decided in deterministic polynomial time with the help of an oracle from PP. Hence, if a problem is PP-hard, then this can be seen as a strong indication that the problem does not belong to PH (otherwise PH would collapse). The levels of the *counting hierarchy* C_i^p ($i \geq 0$) are inductively defined as follows: $\mathsf{C}_0^p = \mathsf{P}$ and $\mathsf{C}_{i+1}^p = \mathsf{PP}^{\mathsf{C}_i^p}$ (the set of languages accepted by a PP-machine as above with an oracle from C_i^p) for all $i \geq 0$. Let $\mathsf{CH} = \bigcup_{i \geq 0} \mathsf{C}_i^p$ be the counting hierarchy. It is not hard to show that $\mathsf{CH} \subseteq \mathsf{PSPACE}$, but it is open whether this inclusion is strict, see [4], [36] for more details.

B. Auxiliary Definitions and Results About Circuit Complexity

The circuit complexity class TC^0 is the class of all languages $L \subseteq \{0,1\}^*$ that can be decided by a family of circuits $(C_n)_{n \geq 0}$ with the following properties:

- The circuits C_n are built up from Boolean gates (AND, OR and NOT) and threshold gates. A threshold gate outputs a 1 if at least half of its inputs are 1, otherwise it outputs a 0.
- The circuit C_n has n input gates $x_1, \ldots, x_n$ and a single output gate, and for an input assignment $\alpha_n : \{x_1, \ldots, x_n\} \to \{0,1\}$ the output gate evaluates to 1 if and only if the bit string $\alpha_n(x_1)\alpha_n(x_2)\cdots\alpha_n(x_n)$ belongs to L.
- There is a constant c such that every circuit C_n has depth at most c, where the depth of a circuit is the length of a longest path from an input gate to the output gate.
- There is a polynomial $p(n)$ such that the circuit C_n has at most $p(n)$ many gates.

Note that this is a non-uniform computation model in the sense that the circuits C_n do not have to follow a common pattern, and this implies that TC^0 also contains undecidable languages. Therefore, TC^0 is often restricted to uniform variants. The most restricted but still non-trivial such variant is DLOGTIME-uniform TC^0. Here, DLOGTIME-uniformity means that one can compute in time $O(\log n)$ (i) the type of a given gate of the nth circuit C_n, and (ii) whether two given gates of the nth circuit are connected by a wire. Here, gates of the nth circuit are encoded by bit strings of length $O(\log n)$. If we do not allow threshold gates in this definition, we obtain DLOGTIME-uniform AC^0.

There are obvious generalisation of the above language classes AC^0 and TC^0 to function classes. Given two n-bit numbers $x, y \in \mathbb{Z}$, $x + y$ (resp., $x \cdot y$) can be computed in DLOGTIME-uniform AC^0 (resp., DLOGTIME-uniform TC^0) [38]. Even the product of n numbers $x_1, \ldots, x_n \in \mathbb{Z}$, each of bit-size at most n, and the integer quotient $\lfloor x/y \rfloor$ ($x, y \in \mathbb{Z}$) can be computed in DLOGTIME-uniform TC^0 [2], [21]. More details on the counting hierarchy (resp., circuit complexity) can be found in [4] (resp., [38]).

C. A Toolbox for the Counting Hierarchy

To place the finitary cost problem into CH, we need some closure results for the counting hierarchy. These results are

based on ideas and results developed in [3], [13], which are, however, not sufficiently general for our purposes.

Let $\mathbb{B} := \{0,1\}^*$. For a k-tuple $\overline{x} = (x_1, \ldots, x_k) \in \mathbb{B}^k$ let $|\overline{x}| = \sum_{i=1}^{k} |x_i|$. The following definition is a slight variant of the definition in [13] that suits our purposes better. Consider a function $f\colon \mathbb{B}^k \to \mathbb{N}$ that maps a k-tuple of binary words to a natural number. We say that f belongs to CH if there exists a polynomial $p(n)$ such that

- for all $\overline{x} \in \mathbb{B}^k$ we have $f(\overline{x}) \leq 2^{2^{p(|\overline{x}|)}}$, and
- the set of tuples

$$L_f := \{(\overline{x}, i) \in \mathbb{B}^k \times \mathbb{N} : \text{the } i\text{th bit of } f(\overline{x}) \text{ is } 1\}$$

belongs to CH (here, we assume that i is given in binary representation).

We also consider mappings $f\colon D_1 \times \cdots \times D_k \to \mathbb{N}$ in CH, where the domains $D_1, \ldots, D_k$ are not $\mathbb{B}$ (e.g., in Proposition 13 below). In this case, we assume some standard encoding of the elements from $D_1, \ldots, D_k$ as words over a binary alphabet.

Lemma 8. *If the function $f\colon \mathbb{B}^{k+1} \to \mathbb{N}$ belongs to CH and $p(n)$ is a polynomial, then also the functions $g\colon \mathbb{B}^k \to \mathbb{N}$ and $h\colon \mathbb{B}^k \to \mathbb{N}$ belong to CH, where*

$$g(\overline{x}) = \sum_{y \in \{0,1\}^{p(|\overline{x}|)}} f(\overline{x}, y) \quad \text{and} \quad h(\overline{x}) = \prod_{y \in \{0,1\}^{p(|\overline{x}|)}} f(\overline{x}, y).$$

Proof. We only prove the statement for products, the proof for sums is the same. To this end, we follow the arguments from [3] showing that BITSLP belongs to the counting hierarchy. As mentioned in Section IV-B, iterated product, i.e., the problem of computing the product of a sequence of binary encoded integers, belongs to DLOGTIME-uniform TC^0. More precisely, there is a DLOGTIME-uniform TC^0 circuit family, where the nth circuit C_n has n^2 many input gates, which are interpreted as n-bit integers $x_1, \ldots, x_n$, and n^2 output gates, which evaluate to the bits in the product $\prod_{i=1}^{n} x_i$. Let c be the depth of the circuits C_n, which is a fixed constant. One can assume that the gates are partitioned into c levels, where level 1 consists of the input gates, level c consists of the output gate and all wires go from a gate on level i to a gate on level $i+1$ for some $1 \leq i \leq c-1$.

Since the function f belongs to CH, there is a polynomial $q(n)$ such that for all $\overline{x} \in \mathbb{B}^k$ and $y \in \mathbb{B}$ we have $f(\overline{x}, y) \leq 2^{2^{q(|\overline{x}|+|y|)}}$. Let $r(n)$ be the polynomial with $r(n) = q(n + p(n))$. Hence, for all $\overline{x} \in \mathbb{B}^k$ and $y \in \{0,1\}^{p(|\overline{x}|)}$ we have $f(\overline{x}, y) \leq 2^{2^{r(|\overline{x}|)}}$. We can assume that $r(n) \geq p(n)$ for all n (simply assume that $q(n) \geq n$).

For an input tuple $\overline{x} \in \mathbb{B}^k$ with $|\overline{x}| = n$ we consider the circuit $D_n := C_{2^{r(n)}}$. It takes $2^{r(n)} \geq 2^{p(n)}$ integers with $2^{r(|\overline{x})}$ bits as input. Hence, we can consider the input tuple

$$\overline{z} = (f(\overline{x}, y_1), f(\overline{x}, y_2), \ldots, f(\overline{x}, y_{2^{p(n)}}), 1, \ldots, 1) \quad (5)$$

for D_n. Here, $y_1, \ldots, y_{2^{p(n)}}$ is the lexicographic enumeration of all binary words of length $p(n)$. We pad the tuple with a sufficient number of ones so that the total length of the tuple is $2^{r(n)}$.

There is a polynomial $s(n)$ such that D_n has at most $2^{s(n)}$ many gates. Hence, a gate of D_n can be identified with a bitstring of length $s(n)$. Then, one shows that for every level $1 \leq i \leq c$ the following set belongs to CH:

$$\{(\overline{x}, u) : \overline{x} \in \mathbb{B}^k, u \in \{0,1\}^{s(|\overline{x}|)}, \text{ gate } u \text{ is on level } i \text{ of } D_{|\overline{x}|} \text{ and evaluates to 1 if } \overline{z} \text{ from (5) is input for } D_{|\overline{x}|}\}.$$

This is shown by a straightforward induction on i as in [3]. For the induction base $i = 1$ one uses the fact that f belongs to CH.

For the statement about sums, the proof is the same using the result that iterated sum belongs to DLOGTIME-uniform TC^0 as well (which is much easier to show than the corresponding result for iterated products). □

Remark 9. *A particular application of Lemma 8 that we will use subsequently is the following: Assume that $f, g\colon \mathbb{B}^k \to \mathbb{N}$ are functions such that for a given tuple $\overline{x} \in \mathbb{B}^k$ the values $f(\overline{x})$ and $g(\overline{x})$ are bounded by $2^{poly(|\overline{x}|)}$ and the binary representations of these numbers can be computed in polynomial time. Then, the mappings defined by $f(\overline{x})!$ and $f(\overline{x})^{g(\overline{x})}$ belong to CH.*

Remark 10. *In our subsequent applications of Lemma 8 we have to consider the case that g is given as*

$$g(\overline{x}) = \sum_{y \in S(\overline{x}) \cap \{0,1\}^{p(|\overline{x}|)}} f(\overline{x}, y),$$

such that for a given tuple $\overline{x} \in \mathbb{B}^k$ and a binary word $y \in \{0,1\}^{p(|\overline{x}|)}$ one can decide in polynomial time whether $y \in S(\overline{x})$. This case can be easily reduced to Lemma 8, since $g(\overline{x}) = \sum_{y \in \{0,1\}^{p(|\overline{x}|)}} f'(\overline{x}, y)$, where f' is defined as

$$f'(\overline{x}, y) := \begin{cases} f(\overline{x}, y) & \text{if } y \in S(\overline{x}) \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, if f belongs to CH, then also f' belongs to CH. The same remark applies to products instead of sums.

Lemma 11. *If the functions $f\colon \mathbb{B}^k \to \mathbb{N}$ and $g\colon \mathbb{B}^k \to \mathbb{N}$ belong to CH, then also the following functions $q\colon \mathbb{B}^k \to \mathbb{N}$ (quotient) and $d\colon \mathbb{B}^k \to \mathbb{N}$ (modified difference) belong to CH:*

$$q(\overline{x}) := \left\lfloor \frac{f(\overline{x})}{g(\overline{x})} \right\rfloor \quad \text{and} \quad d(\overline{x}) := \max\{0, f(\overline{x}) - g(\overline{x})\}$$

Proof. The proof is the same as for Lemma 8, using the result that division (resp., subtraction) of integers encoded in binary is in DLOGTIME-uniform TC^0 [21] (resp., $\mathsf{AC}^0 \subseteq \mathsf{TC}^0$ [38]). □

Lemma 11 implies:

Lemma 12. *If the functions $f\colon \mathbb{B}^k \to \mathbb{N}$ and $g\colon \mathbb{B}^k \to \mathbb{N}$ belong to CH, then also the following function $h\colon \mathbb{B}^k \to \mathbb{N}$ belongs to CH:*

$$h(\overline{x}) := \begin{cases} 1 & \text{if } f(\overline{x}) > g(\overline{x}) \\ 0 & \text{otherwise} \end{cases}$$

D. Applications to the Cost Problem

We apply our toolkit for the counting hierarchy to show that any bit of the number $N(\mathcal{A}, \boldsymbol{p})$ can be evaluated in the counting hierarchy, see Proposition 13 below. This, in turn, enables us to place the cost problem in the counting hierarchy. Formally:

BITPARIKH

INPUT: A DFA $\mathcal{A}$ over an alphabet Σ, a Parikh vector $\boldsymbol{p} \in \mathbb{N}^{\Sigma}$, and a number $i \in \mathbb{N}$ encoded binary.

QUESTION: Is the ith bit of $N(\mathcal{A}, \boldsymbol{p})$ equal to one?

Proposition 13. BITPARIKH *is in* CH.

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \{q_f\}, \delta)$ be a DFA which we may assume to be well-formed, cf. Section III-F, and let $\boldsymbol{p} \in \mathbb{N}^{\Sigma}$ be a Parikh vector. By Lemma 7 we have $N(\mathcal{A}, \boldsymbol{p}) = \sum_{w \in W(\mathcal{A}, \boldsymbol{p})} e(\mathcal{A}^w, q_0, q_f)$, where $w \colon \delta \to \mathbb{N}$ and $\mathcal{A}^w$ is the edge-weighted multi-graph obtained from $\mathcal{A}$ by assigning to every transition $t \in \delta$ the weight $w(t)$. Note that for a given mapping $w \colon \delta \to \mathbb{N}$ one can check in polynomial time whether $w \in W(\mathcal{A}, \boldsymbol{p})$, see Section III-F. Moreover, from w and $\mathcal{A}$ one can construct the edge-weighted multi-graph $\mathcal{A}^w$ in polynomial time. Together with Lemma 8 and Remark 10, this implies that it suffices to show that the mapping $(G, u, v) \mapsto e(G, u, v)$, where $G = (V, E, s, t, w)$ is an edge-weighted multi-graph and $u, v \in V$ are different node, belongs to CH. By Lemma 6 we have $e(G, u, v) = e(G')/\prod_{e \in E} w(e)!$, where $G' = \tilde{G}_{v,u}$. Given G, u and v, we can check in polynomial time whether $\tilde{G}_{v,u}$ is connected and Eulerian. If this is not the case, then $e(G, u, v) = 0$. Otherwise, Theorems 4 and 5 yield

$$e(G, u, v) = \det(L(G'')^{1,1}) \cdot \prod_{v \in V} (d^-_{G'}(v) - 1)! / \prod_{e \in E} w(e)!,$$

where G'' is obtained from G' by removing all loops. The determinant $\det(L(G'')^{1,1})$ can be computed in polynomial time from the edge weights of G. Finally, Lemma 8 and 11 imply that the mapping $(G, u, v) \mapsto e(G, u, v)$ belongs to CH. □

Proposition 13 can be used to show that a bit of the probability $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$ can be computed in CH. Formally, we consider the following problem:

BITCOST PROBLEM

INPUT: An instance $\mathcal{C}$, φ, τ of the cost problem and an integer $j \geq 0$ encoded in binary.

QUESTION: Is the jth bit of $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$ equal to one?

We call an instance of the BITCOST problem finitary (resp. co-finitary) if the underlying cost problem is finitary (resp. co-finitary). We have:

Proposition 14. *The finitary and co-finitary* BITCOST *problems belong to* CH.

Proof. The proof is a convolution of all results and concepts introduced in this paper so far. Let $\mathcal{C}$, φ, τ and j be an instance of the finitary BITCOST problem. Moreover, let E be the set of edges of $\mathcal{C}$, and for every edge $e \in E$, let $\Delta(e) = m_e/d_e$ be the probability of e. Recall that the numbers m_e and d_e are given in binary notation. By Lemma 2, there is some $c \leq 2^{O((|\mathcal{C}|+|\varphi|)^3)}$ such that $|u| \leq c$ for all $u \in L_{\varphi}(\mathcal{A}_{\mathcal{C}})$, and in particular $\|\Psi(u)\|_1 \leq c$. From Equation (3), it then follows that

$$\mathcal{P}(K_{\mathcal{C}} \models \varphi) = \frac{\displaystyle\sum_{\boldsymbol{p} \in \Psi(L_{\varphi}(\mathcal{A}_{\mathcal{C}}))} N(\mathcal{A}_{\mathcal{C}}, \boldsymbol{p}) \cdot \prod_{e \in E} m_e^{\boldsymbol{p}(e)} d_e^{c+1-\boldsymbol{p}(e)}}{\displaystyle\prod_{e \in E} d_e^{c+1}}.$$

From the above formula for $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$, it follows that the jth bit of $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$ is the least significant bit of the following integer:

$$P(\mathcal{C}, \varphi, j) := \left\lfloor \frac{\displaystyle\sum_{\boldsymbol{p} \in \Psi(L_{\varphi}(\mathcal{A}_{\mathcal{C}}))} 2^j \, N(\mathcal{A}_{\mathcal{C}}, \boldsymbol{p}) \prod_{e \in E} m_e^{\boldsymbol{p}(e)} d_e^{c+1-\boldsymbol{p}(e)}}{\displaystyle\prod_{e \in E} d_e^{c+1}} \right\rfloor$$

By Proposition 13, the function $N(\mathcal{A}_{\mathcal{C}}, \boldsymbol{p})$ belong to CH. Hence, Lemma 8 and 11 (see also Remark 9 and 10 after Lemma 8) imply that the function $P(\mathcal{C}, \varphi, j)$ belongs to CH (which implies that the finitary BITCOST problem belongs to CH). For this, note that for given $\mathcal{C}$, φ and $\boldsymbol{p}$ with $\|\boldsymbol{p}\|_1 \leq c$, one can decide in polynomial time whether $\boldsymbol{p} \in L_{\varphi}(\mathcal{A}_{\mathcal{C}})$, which allows to apply Remark 10.

For a co-finite instance $\mathcal{P}(K_{\mathcal{C}} \models \varphi) = 1 - \mathcal{P}(K_{\mathcal{C}} \models \neg\varphi)$ holds. Hence, we have to compute the least significant bit of the number $\lfloor 2^j - 2^j \cdot \mathcal{P}(K_{\mathcal{C}} \models \neg\varphi) \rfloor$. This can be done in CH by using again the above formula for $\mathcal{P}(K_{\mathcal{C}} \models \neg\varphi)$ and the lemmas from Section IV. □

Now we can prove our main result:

Theorem 15. *The finitary and co-finitary cost problems belong to* CH.

Proof. Let $\mathcal{C}$, φ, τ be an instance of the finitary cost problem, and let $\tau = m/d$. Following the proof of Proposition 14, deciding $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$ reduces to deciding whether

$$\sum_{\boldsymbol{p} \in \Psi(L_{\varphi}(\mathcal{A}_{\mathcal{C}}))} d \cdot N(\mathcal{C}, \boldsymbol{p}) \cdot \prod_{e \in E} m_e^{\boldsymbol{p}(e)} d_e^{c+1-\boldsymbol{p}(e)} \geq m \cdot \prod_{e \in E} d_e^{c+1}.$$

This can be decided in CH as in the proof of Proposition 14, where in addition we have to use Lemma 12. Finally, in the co-finite case we check as above whether $\mathcal{P}(K_{\mathcal{C}} \models \neg\varphi) \leq 1 - \tau$. □

V. EXPERIMENTAL EVALUATION

While so far the main focus of this paper has been on finding good complexity-theoretic upper bounds for the cost problem, in this section we briefly discuss on which instances we can expect Algorithm 1 to perform well. We first describe how our approach can be seen as a partial-order reduction technique, and then compare an implementation of Algorithm 1 with the state-of-the-art probabilistic model checker PRISM [25].

A. Counting Paths as a Partial Order Reduction Technique

As already briefly mentioned in the introduction, our application of the BEST theorem can be viewed as a partial order reduction technique for cost chains. For finitary instances, by Lemma 2 it is always possible to obtain an explicit Markov chain by hard-coding the relevant cost vectors into the control structure. This is partly the basis of the approaches described for instance in [5], [24] for lower-dimensional cost chains. Leaving aside the possible exponential blow-up, a potential drawback is that the resulting Markov chain does not allow for subsuming paths that accumulate the same cost vectors due to commutativity of the transitions traversed. This is in stark contrast to our approach: a Parikh vector can compactly represent and subsume an exponential number of such paths on which an exponential states may occur. For an n-dimensional cost chain and a Parikh vector $\boldsymbol{p}$, in the best case $\boldsymbol{p}$ subsumes $n^{\|\boldsymbol{p}\|_1}$ many states along paths leading to the control state represented by $\boldsymbol{p}$.

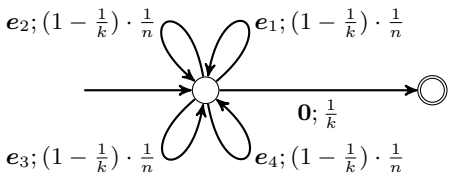

Fig. 2. Cost chain $\mathcal{G}_{n,k}$ modelling the geometric coupon collector's problem for $n = 4$.

For an illuminating example, consider a variant of the coupon collector's problem from the introduction presented in Figure 2. In the cost chain $\mathcal{G}_{n,k}$ it is no longer the case that coupons are bought until every coupon type (out of n types) has been discovered. Instead, in $\mathcal{G}_{n,k}$ the total number of coupons bought is distributed geometrically with expectation k. Suppose we wish to compute the probability of reaching the target state with a cost vector whose components all lie in an interval $[\ell, u]$. A Markov chain hard-coding the cost vectors consists of n^u many states that need to be explicitly constructed. Our approach avoids this construction and only needs to consider $n^{u-\ell}$ many Parikh images of paths leading into this interval. Thus, as long as $u - \ell$ is not too large, the size of the total explicit finite state space is almost irrelevant for the empirical performance of Algorithm 1 on this example.

As a general rule of thumb, the situations in which Algorithm 1 becomes advantageous occur when $\#\Psi(L_\varphi(\mathcal{A}_\mathcal{C}))$ is relatively small, but the overall state space is huge. Thus, Algorithm 1 is particularly well-suited for exactly computing probabilities of rare events of highly succinct probabilistic systems.

B. A Comparison with PRISM

The goal of this section is to empirically compare the algorithmic approach to the cost problem developed in this paper with the PRISM model checker [25]. In particular, our experiments show that SMT solvers are sufficiently powerful in order to enumerate all Parikh vectors fulfilling a cost formula, which could at first sight be seen as a limiting factor of our approach.

To this end, we implemented Algorithm 1 in a tool called QUANT[2]. It is written in the PYTHON programming language and consists of about 200 lines of code. In order to enumerate Parikh vectors, we use the SMT-solver Z3 [17]. The benchmarks were run on a Samsung Series 9 ultrabook with an Intel® Core™ i5-2467M 1.60 GHz processor with 4 GB DDR3 1066 MHz under Ubuntu Linux 16.04. For the run times, we used the `user` time reported by the Linux tool `time`. The version of PRISM that we compared to is 4.3.1. We set the timeout to 300s and allowed for a maximal memory usage of 1Gb.

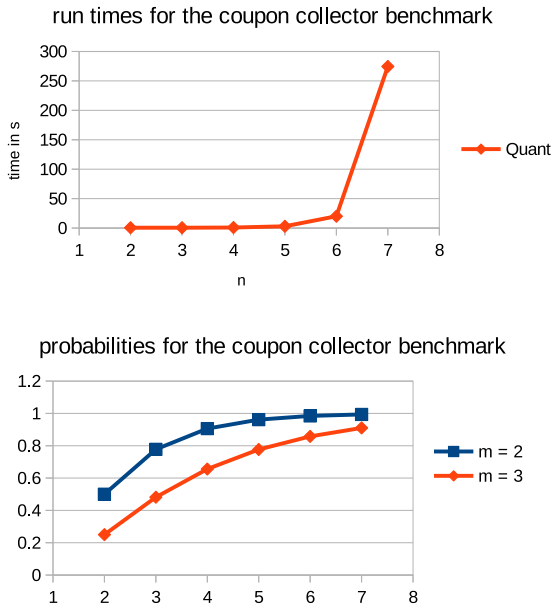

Fig. 3. Run times and probabilities for the first experiment.

We first evaluated QUANT on the classical coupon collector's problem that we presented in the introduction by computing the probability of the event that after collecting all coupons, there is some coupon that has been drawn at least two or three times, respectively, i.e., the probability $\mathcal{P}(K_{\mathcal{C}_n} \models \varphi_{n,m})$, where $\varphi_{n,m} := x_1 \geq m \vee x_2 \geq m \vee \cdots \vee x_n \geq m$, $2 \leq n \leq 7$ and $m \in \{2, 3\}$. For $n = 7$ and $m = 3$, QUANT can compute this probability, which is $\approx$0.91, in less than 274s. The run time of QUANT and the calculated probabilities are illustrated in Figure 3. As expected, $\mathcal{P}(K_{\mathcal{C}_n} \models \varphi_{n,m})$ increases with larger n and is almost 1 for $n = 7$ and $m = 2$. On this instance, PRISM outperforms QUANT significantly and discharges every instance within a couple of seconds. This is not surprising: for $n = 7$ and $m = 3$, the generated state

[2]The tool QUANT and the experiments performed in this section can be obtained from the authors upon request.

space is with 1822 states very modest, while there are many different Parikh images leading to the same configuration that have to be inspected by QUANT.

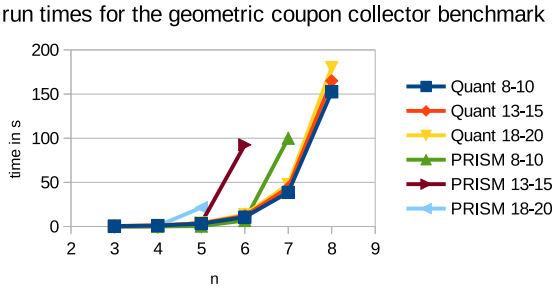

Fig. 4. Run times for the second experiment. Missing data points indicate a timeout or out-of-memory error.

As discussed in the previous section, the situation should become more favourable for QUANT when we consider the geometric variant of the coupon collector's problem from Figure 2. In our benchmarks, we compute the probabilities that at the end every coupon has been drawn between 8 and 10 times, 13 and 15 times, and 18 and 20 times. Formally, we compute $\mathcal{P}(K_{\mathcal{G}_{n,k}} \models \psi_n)$, where $\psi_n := \bigwedge_{1 \le i \le n} \ell \le x_i \le u$ for $(\ell, u) \in \{(8,10),(13,15),(18,20)\}$, and $k = n \cdot (\ell + 1)$. In Figure 4, we see that the run time of QUANT is almost independent from the choices of ℓ and u. This is in stark contrast to PRISM. For $(\ell, u) = (8,10),(13,15),(18,20)$, PRISM cannot handle the cases $n = 8$, $n = 7$ and $n = 6$ any longer. In the respective cases, the number of states generated by PRISM before failing to discharge the next n are $\approx 7.1 \cdot 10^7$, $\approx 4.8 \cdot 10^7$ and $\approx 3.6 \cdot 10^7$.

VI. CONCLUSION

Computing quantiles in cost chains is a natural problem in the quantitative analysis of stochastic systems. Existing complexity results show that polynomial-time algorithms for the cost problem are unlikely to exist. Nevertheless, in this paper we have devised an algorithm that can both be implemented efficiently and proves that the finitary cost problem is in CH. This complements a result from [18] stating hardness for PP and POSSLP. Our algorithm and its implementation employ a variety of techniques and concepts, including formal language theory, Presburger arithmetic, the BEST theorem, and SMT-solvers.

There are several further research directions that we plan to explore. Decidability of the infinitary cost problem is open. A more precise complexity analysis of the finitary cost problem might place it in a low level within CH. One might also improve the POSSLP and PP lower bounds which hold even in the non-negative one-dimensional case [18]. In a forthcoming paper we plan an investigation of the complexity of problems related to counting the number of words that both have a given Parikh image and are accepted by a given language acceptor.

While QUANT shows promising performance in our benchmarks, more engineering efforts might conceivably lead to better scalability. We would like to explore how our methods can be combined with iterative linear programming approaches that have been described in [5].

ACKNOWLEDGMENT

Parts of this research were supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS when the first author was affiliated with LSV, CNRS & ENS Cachan, Université Paris-Saclay, France.

REFERENCES

[1] M. Aigner. *A Course in Enumeration*, volume 238 of *Graduate Texts in Mathematics*. Springer, 2007.
[2] E. Allender, N. Balaji, and S. Datta. Low-depth uniform threshold circuits and the bit-complexity of straight line programs. In *Proc. MFCS 2014, Part II*, volume 8635 of *LNCS*, pages 13–24. Springer, 2014.
[3] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.
[4] E. Allender and K. W. Wagner. Counting hierarchies: Polynomial time and constant depth circuits. *Bulletin of the EATCS*, 40:182–194, 1990.
[5] C. Baier, M. Daum, C. Dubslaff, J. Klein, and S. Klüppelholz. Energy-utility quantiles. In *Proc. NFM 2014*, volume 8430 of *LNCS*, pages 285–299. Springer, 2014.
[6] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
[7] C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: complexity and decidability. In *Proc. CSL-LICS 2014*, pages 11:1–11:10. ACM, 2014.
[8] N. Balaji and S. Datta. Bounded treewidth and space-efficient linear algebra. In *Proc. TAMC 2015*, volume 9076 of *LNCS*, pages 297–308. Springer, 2015.
[9] N. Balaji, S. Datta, and V. Ganesan. Counting Euler tours in undirected bounded treewidth graphs. In *Proc. FSTTCS 2015*, volume 45 of *LIPIcs*, pages 246–260, 2015.
[10] U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014.
[11] G. Brightwell and P. Winkler. Counting eulerian circuits is #P-complete. In *Proc. ALENEX / ANALCO 2005*, pages 259–262. SIAM, 2005.
[12] V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games. In *Proc. STACS 2014*, volume 25 of *LIPIcs*, pages 199–213, 2014.
[13] P. Bürgisser. On defining integers and proving arithmetic circuit lower bounds. *Comput. Complex.*, 18(1):81–103, 2009.
[14] K. Chatterjee, Z. Komárková, and J. Kretínský. Unifying two views on multiple mean-payoff objectives in Markov decision processes. In *Proc. LICS 2015*, pages 244–256. IEEE, 2015.
[15] L. Clemente and J.-F. Raskin. Multidimensional beyond worst-case and almost-sure problems for mean-payoff objectives. In *Proc. LICS 2015*, pages 257–268. IEEE, 2015.
[16] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proc. LICS 1998*, pages 454–465. IEEE Computer Society, 1998.
[17] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proc. TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
[18] C. Haase and S. Kiefer. The odds of staying on budget. In *Proc. ICALP 2015, Part II*, volume 9135 of *LNCS*, pages 234–246. Springer, 2015.
[19] C. Haase and S. Kiefer. The complexity of the Kth largest subset problem and related problems. *Inf. Process. Lett.*, 116(2):111–115, 2016.
[20] M. Hague and A. W. Lin. Model checking recursive programs with numeric data types. In *Proc. CAV 2011*, volume 6806 of *LNCS*, pages 743–759. Springer, 2011.
[21] W. Hesse, E. Allender, and D. A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.
[22] B.L. Kaminski, J.-P. Katoen, C. Matheja, and F. Olmedo. Weakest precondition reasoning for expected run-times of probabilistic programs. In *Proc. ESOP 2016*, volume 9632 of *LNCS*, pages 364–389. Springer, 2016.

[23] J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Perform. Eval.*, 68(2):90–104, 2011.

[24] J. Klein, C. Baier, P. Chrszon, M. Daum, C. Dubslaff, S. Klüppelholz, S. Märcker, and S. Müller. Advances in symbolic probabilistic model checking with PRISM. In *Proc. TACAS 2016*, volume 9636 of *LNCS*, pages 349–366. Springer, 2016.

[25] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. CAV 2011*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[26] R. E. Ladner. Polynomial space counting problems. *SIAM J. Comput.*, 18(6):1087–1097, 1989.

[27] F. Laroussinie and J. Sproston. Model checking durational probabilistic systems. In *Proc. FOSSACS 2005*, volume 3441 of *LNCS*, pages 140–154. Springer, 2005.

[28] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[29] L. Pottier. Minimal solutions of linear Diophantine systems : bounds and algorithms. In *Proc. RTA 1991*, volume 488 of *LNCS*, pages 162–173. Springer, 1991.

[30] M. Randour, J.-F. Raskin, and O. Sankur. Percentile queries in multi-dimensional Markov decision processes. In *Proc. CAV 2015, Part I*, volume 9206 of *LNCS*, pages 123–139. Springer, 2015.

[31] M. Randour, J.-F. Raskin, and O. Sankur. Variations on the stochastic shortest path problem. In *Proc. VMCAI 2015*, volume 8931 of *LNCS*, pages 1–18. Springer, 2015.

[32] H. Seidl, T. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *Proc. ICALP 2004*, volume 3142 of *LNCS*, pages 1136–1149. Springer, 2004.

[33] J. Simon. On the difference between one and many. In *Proc. ICALP 1977*, volume 52 of *LNCS*, pages 480–491. Springer, 1977.

[34] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[35] T. Tomita, S. Hiura, S. Hagihara, and N. Yonezaki. A temporal logic with mean-payoff constraints. In *Proc. ICFEM 2012*, volume 7635 of *LNCS*, pages 249–265. Springer, 2012.

[36] J. Torán. Complexity classes defined by counting quantifiers. *J. ACM*, 38(3):753–774, 1991.

[37] M. Ummels and C. Baier. Computing quantiles in Markov reward models. In *Proc. FOSSACS 2013*, volume 7794 of *LNCS*, pages 353–368. Springer, 2013.

[38] H. Vollmer. *Introduction to Circuit Complexity*. Springer, 1999.