The Odds of Staying on Budget

Christoph Haase[1]* and Stefan Kiefer[2]

[1] Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan, France
[2] Department of Computer Science, University of Oxford, UK

Abstract. Given Markov chains and Markov decision processes (MDPs) whose transitions are labelled with non-negative integer costs, we study the computational complexity of deciding whether the probability of paths whose accumulated cost satisfies a Boolean combination of inequalities exceeds a given threshold. For acyclic Markov chains, we show that this problem is PP-complete, whereas it is hard for the PosSLP problem and in PSpace for general Markov chains. Moreover, for acyclic and general MDPs, we prove PSpace- and EXP-completeness, respectively. Our results have direct implications on the complexity of computing reward quantiles in succinctly represented stochastic systems.

1 Introduction

Computing the shortest path from s to t in a directed graph is a ubiquitous problem in computer science, so shortest-path algorithms such as Dijkstra's algorithm are a staple for every computer scientist. These algorithms work in polynomial time even if the edges are weighted, so questions of the following kind are easy to answer:

(I) *Is it possible to travel from Copenhagen to Kyoto in less than 15 hours?*

From a complexity-theoretic point of view, even computing the length of the shortest path lies in NC, the class of problems with "efficiently parallelisable" algorithms.[3]

The shortest-path problem becomes more intricate as soon as uncertainties are taken into account. For example, additional information such as "*there might be congestion in Singapore, so the Singapore route will, with probability 10%, trigger a delay of 1 hour*" naturally leads to questions of the following kind:

(II) *Is there a travel plan avoiding trips longer than 15 hours with probability* ≥ 0.9*?*

Markov decision processes (MDPs) are the established model to formalise problems such as (II). In each *state* of an MDP some *actions* are enabled, each of which is associated with a probability distribution over outgoing *transitions*. Each transition, in turn, determines the successor state and is equipped with a non-negative "weight". The weight could be interpreted as time, distance, reward, or—as in this paper—as *cost*. For another example, imagine the plan of a research project whose workflow can be modelled by a directed weighted graph. In each project state the investigators can hire a programmer, travel to collaborators, acquire new equipment, etc., but each action costs money, and the result (i.e., the next project state) is probabilistic. The objective is to meet the goals of the project before exceeding its budget for the total accumulated cost. This leads to questions such as:

* Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.

[3] The NC algorithm performs "repeated squaring" of the weight matrix in the $(\max, +)$-algebra.

(III) *Is there a strategy to stay on budget with probability* ≥ 0.85*?*

MDP problems like (II) and (III) become even more challenging when each transition is equipped with both a cost and a *utility*, e.g. in order to model problems that aim at maximising the probability that both a given budget is kept *and* a minimum total utility is achieved. Such *cost-utility trade-offs* have recently been studied in [3].

The problems (II) and (III) may become easier if there is no non-determinism, i.e., there are no actions. We then obtain *Markov chains* where the next state and the incurred transition cost are chosen in a purely probabilistic fashion. Referring to the project example above, the activities may be completely planned out, but their effects (i.e. cost and next state) may still be probabilistic, yielding problems of the kind:

(IV) *Will the budget be kept with probability* ≥ 0.85*?*

Closely related to the aforementioned decision problems is the following optimisation problem, referred to as the *quantile query* in [2, 3, 23]. A quantile query asked by a funding body, for instance, could be the following:

(V) *Given a probability threshold* τ*, compute the smallest budget that suffices with probability at least* τ*.*

Non-stochastic problems like (I) are well understood. The purpose of this paper is to investigate the complexity of MDP problems such as (II) and (III), of Markov-chain problems such as (IV), and of quantile queries like (V). More formally, the models we consider are Markov chains and MDPs with non-negative integer costs, and the main focus of this paper is on the *cost problem* for those models: Given a budget constraint φ represented as a Boolean combination of linear inequalities and a probability threshold τ, we study the complexity of determining whether the probability of paths reaching a designated target state with cost consistent with φ is at least τ.

In order to highlight and separate our problems more clearly from those in the literature, let us briefly discuss two approaches that do not, at least not in an obvious way, resolve the core challenges. First, one approach to answer the MDP problems could be to compute a strategy that minimises the *expected* total cost, which is a classical problem in the MDP literature, solvable in polynomial time using linear programming methods [17]. However, minimising the expectation may not be optimal: if you don't want to be late, it may be better to walk than to wait for the bus, even if the bus saves you time in average. The second approach with shortcomings is to phrase problems (II), (III) and (IV) as MDP or Markov-chain *reachability* problems, which are also known to be solvable in polynomial time. This, however, ignores the fact that numbers representing cost are commonly represented in their natural succinct *binary* encoding. Augmenting each state with possible accumulated costs leads to a blow-up of the state space which is exponential in the representation of the input, giving an EXP upper bound as in [3].

Our contribution. The goal of this paper is to comprehensively investigate under which circumstances and to what extent the complexity of the cost problem and of quantile queries may be below the EXP upper bound. We also provide new lower bounds, much stronger than the best known NP lower bound derivable from [14]. We distinguish between acyclic and general control graphs. In short, we show that the cost problem is

– PP-complete for acyclic Markov chains, and hard for the POSSLP problem and in PSPACE in the general case; and
– PSPACE-complete for acyclic MDPs, and EXP-complete for general MDPs.

Related Work. The motivation for this paper comes from the work on quantile queries in [2, 3, 23] mentioned above and on model checking so-called durational probabilistic systems [14] with a probabilistic timed extension of CTL. While the focus of [23] is mainly on "qualitative" problems where the probability threshold is either 0 or 1, an iterative linear-programming-based approach for solving quantile queries has been suggested in [2, 3]. The authors report satisfying experimental results, the worst-case complexity however remains exponential time. Settling the complexity of quantile queries has been identified as one of the current challenges in the conclusion of [3].

Recently, there has been considerable interest in models of stochastic systems that extend weighted graphs or counter systems, see [19] for a very recent survey. Multi-dimensional percentile queries for various payoff functions are studied in [18]. The work by Bruyère et al. [8] has also been motivated by the fact that minimising the expected total cost is not always an adequate solution to natural problems. For instance, they consider the problem of computing a scheduler in an MDP with positive integer weights that ensures that both the expected and the maximum incurred cost remain below a given values. Other recent work also investigated MDPs with a single counter ranging over the non-negative integers, see e.g. [6, 7]. However, in that work updates to the counter can be both positive and negative. For that reason, the analysis focuses on questions about the counter value *zero*, such as designing a strategy that maximises the probability of reaching counter value zero.

2 Preliminaries

We write $\mathbb{N} = \{0, 1, 2, \ldots\}$. For a countable set X we write $dist(X)$ for the set of *probability distributions* over X; i.e., $dist(X)$ consists of those functions $f : X \to [0, 1]$ such that $\sum_{x \in X} f(x) = 1$.

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 an initial state, and $\delta : S \to dist(S)$ 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 $s_0 s_1 \cdots \in \{s_0\} S^\omega$ with $s_i \to s_{i+1}$ for $i \in \mathbb{N}$. We write $Run(s_0 \cdots s_k)$ for the set of runs that start with $s_0 \cdots s_k$. To $\mathcal{M}$ we associate the standard probability space $(Run(s_0), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all basic cylinders $Run(s_0 \cdots s_k)$ with $s_0 \cdots s_k \in \{s_0\} S^*$, and $\mathcal{P} : \mathcal{F} \to [0, 1]$ is the unique probability measure such that $\mathcal{P}(Run(s_0 \cdots s_k)) = \prod_{i=1}^{k} \delta(s_{i-1})(s_i)$.

Markov Decision Processes. A *Markov decision process (MDP)* is a tuple $\mathcal{D} = (S, s_0, A, En, \delta)$, where S is a countable set of states, $s_0 \in S$ is the initial state, A is a finite set of actions, $En : S \to 2^A \setminus \emptyset$ is an action enabledness function that assigns to each state s the set $En(s)$ of actions enabled in s, and $\delta : S \times A \to dist(S)$ is a probabilistic transition function that maps a state s and an action $a \in En(s)$ enabled

in s to a probability distribution over the successor states. A (deterministic, memoryless) *scheduler* for $\mathcal{D}$ is a function $\sigma : S \to A$ with $\sigma(s) \in En(s)$ for all $s \in S$. A scheduler σ induces a Markov chain $\mathcal{M}_\sigma = (S, s_0, \delta_\sigma)$ with $\delta_\sigma(s) = \delta(s, \sigma(s))$ for all $s \in S$. We write $\mathcal{P}_\sigma$ for the corresponding probability measure of $\mathcal{M}_\sigma$.

Cost Processes. A *cost process* is a tuple $\mathcal{C} = (Q, q_0, t, A, En, \Delta)$, where Q is a finite set of control states, $q_0 \in Q$ is the initial control state, t is the target control state, A is a finite set of actions, $En : Q \to 2^A \setminus \emptyset$ is an action enabledness function that assigns to each control state q the set $En(q)$ of actions enabled in q, and $\Delta : Q \times A \to dist(Q \times \mathbb{N})$ is a probabilistic transition function. Here, for $q, q' \in Q$, $a \in En(q)$ and $k \in \mathbb{N}$, the value $\Delta(q, a)(q', k) \in [0, 1]$ is the probability that, if action a is taken in control state q, the cost process transitions to control state q' and cost k is incurred. For the 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 rational), and for each $q \in Q$ and $a \in En(q)$, the distribution $\Delta(q, a)$ is described by the list of triples (q', k, p) with $\Delta(q, a)(q', k) = p > 0$ (so we assume this list to be finite). Consider the directed graph $G = (Q, E)$ with

$$E := \{(q, q') \in (Q \setminus \{t\}) \times Q : \exists a \in En(q)\ \exists k \in \mathbb{N}.\ \Delta(q, a)(q', k) > 0\}\ .$$

We call $\mathcal{C}$ *acyclic* if G is acyclic (which can be determined in linear time).

A cost process $\mathcal{C}$ induces an MDP $\mathcal{D}_\mathcal{C} = (Q \times \mathbb{N}, (q_0, 0), A, En', \delta)$ with $En'(q, c) = En(q)$ for all $q \in Q$ and $c \in \mathbb{N}$, and $\delta((q, c), a)(q', c') = \Delta(q, a)(q', c' - c)$ for all $q, q' \in Q$ and $c, c' \in \mathbb{N}$ and $a \in A$. For a state $(q, c) \in Q \times \mathbb{N}$ in $\mathcal{D}_\mathcal{C}$ we view q as the current control state and c as the current cost, i.e., the cost accumulated thus far. We refer to $\mathcal{C}$ as a *cost chain* if $|En(q)| = 1$ holds for all $q \in Q$. In this case one can view $\mathcal{D}_\mathcal{C}$ as the Markov chain induced by the unique scheduler of $\mathcal{D}_\mathcal{C}$. For cost chains, actions are not relevant, so we describe cost chains just by the tuple $\mathcal{C} = (Q, q_0, t, \Delta)$.

Recall that we restrict schedulers to be deterministic and memoryless, as such schedulers will be sufficient for the objectives in this paper. Note, however, that our definition allows schedulers to depend on the current cost, i.e., we may have schedulers σ with $\sigma(q, c) \neq \sigma(q, c')$.

The accumulated cost K. In this paper we will be interested in the cost accumulated during a run before reaching the target state t. For this cost to be a well-defined random variable, we make two assumptions on the system: (i) We assume that $En(t) = \{a\}$ holds for some $a \in A$ and $\Delta(t, a)(t, 0) = 1$. Hence, runs that visit t will not leave t and accumulate only a finite cost. (ii) We assume that for all schedulers the target state t is almost surely reached, i.e., for all schedulers the probability of eventually visiting a state (t, c) with $c \in \mathbb{N}$ is equal to one. The latter condition can be verified by graph algorithms in time quadratic in the input size, e.g., by computing the *maximal end components* of the MDP obtained from $\mathcal{C}$ by ignoring the cost, see e.g. [4, Alg. 47].

Given a cost process $\mathcal{C}$ we define a random variable $K_\mathcal{C} : Run((q_0, 0)) \to \mathbb{N}$ such that $K_\mathcal{C}((q_0, 0)\ (q_1, c_1)\ \cdots) = c$ if there exists $i \in \mathbb{N}$ with $(q_i, c_i) = (t, c)$. We often drop the subscript from $K_\mathcal{C}$ if the cost process $\mathcal{C}$ is clear from the context. We view $K(w)$ as the accumulated cost of a run w.

From the above-mentioned assumptions on t, it follows that for any scheduler the random variable K is almost surely defined. Dropping assumption (i) would allow the

same run to visit states (t, c_1) and (t, c_2) for two different $c_1, c_2 \in \mathbb{N}$. There would still be reasonable ways to define a cost K, but no apparently best way. If assumption (ii) were dropped, we would have to deal with runs that do not visit the target state t. In that case one could study the random variable K as above *conditioned* under the event that t is visited. For Markov chains, [5, Sec. 3] describes a transformation that preserves the distribution of the conditional cost K, but t is almost surely reached in the transformed Markov chain. In this sense, our assumption (ii) is without loss of generality for cost chains. For general cost processes the transformations of [5] do not work. In fact, a scheduler that "optimises" K conditioned under reaching t might try to avoid reaching t once the accumulated cost has grown unfavourably. Hence, dropping assumption (ii) in favour of conditional costs would give our problems an aspect of multi-objective optimisation, which is not the focus of this paper.

The cost problem. Let x be a fixed variable. An *atomic cost formula* is an inequality of the form $x \leq B$ where $B \in \mathbb{N}$ is encoded in binary. A *cost formula* is an arbitrary Boolean combination of atomic cost formulas. A number $n \in \mathbb{N}$ *satisfies* a cost formula φ, in symbols $n \models \varphi$, if φ is true when x is replaced by n.

This paper mainly deals with the following decision problem: given a cost process $\mathcal{C}$, a cost formula φ, and a probability threshold $\tau \in [0, 1]$, the *cost problem* asks whether there exists a scheduler σ with $\mathcal{P}_\sigma(K_\mathcal{C} \models \varphi) \geq \tau$. The case of an atomic cost formula φ is an important special case. Clearly, for cost chains $\mathcal{C}$ the cost problem simply asks whether $\mathcal{P}(K_\mathcal{C} \models \varphi) \geq \tau$ holds. One can assume $\tau = 1/2$ without loss of generality, thanks to a simple construction, see Prop. 11 in App. A. Moreover, with an oracle for the cost problem at hand, one can use binary search over τ to approximate $\mathcal{P}_\sigma(K \models \varphi)$: i oracle queries suffice to approximate $\mathcal{P}_\sigma(K \models \varphi)$ within an absolute error of 2^{-i}.

By our definition, the MDP $\mathcal{D}_\mathcal{C}$ is in general infinite as there is no upper bound on the accumulated cost. However, when solving the cost problem, there is no need to keep track of costs above B, where B is the largest number appearing in φ. So one can solve the cost problem in so-called *pseudo-polynomial time* (i.e., polynomial in B, not in the size of the encoding of B) by computing an explicit representation of a restriction, say $\widehat{\mathcal{D}}_\mathcal{C}$, of $\mathcal{D}_\mathcal{C}$ to costs up to B, and then applying classical linear-programming techniques [17] to compute the optimal scheduler for the finite MDP $\widehat{\mathcal{D}}_\mathcal{C}$. Since we consider reachability objectives, the optimal scheduler is deterministic and memoryless. This shows that our restriction to deterministic memoryless schedulers is without loss of generality. In terms of our succinct representation we have:

Proposition 1. *The cost problem is in* EXP.

Heuristic improvements to this approach were suggested in [23, 2]. The subject of this paper is to investigate to what extent the EXP complexity is optimal.

3 Quantile Queries

In this section we consider the following function problem, referred to as *quantile query* in [23, 2, 3]. Given a cost chain $\mathcal{C}$ and a probability threshold τ, a quantile query asks for the smallest budget B such that $\mathcal{P}_\sigma(K_\mathcal{C} \leq B) \geq \tau$. We show that polynomially many oracle queries to the cost problem for atomic cost formulas "$x \leq B$" suffice to answer a

quantile query. This can be done using binary search over the budget B. The following proposition, proved in App. B, provides a suitable general upper bound on this binary search, by exhibiting a concrete sufficient budget, computable in polynomial time:

Proposition 2. *Suppose $0 \leq \tau < 1$. Let p_{min} be the smallest non-zero probability and k_{max} be the largest cost in the description of the cost process. Then $\mathcal{P}_\sigma(K \leq B) \geq \tau$ holds for all schedulers σ, where*

$$B := k_{max} \cdot \left\lceil |Q| \cdot \left(-\ln(1-\tau)/p_{min}^{|Q|} \ + \ 1 \right) \right\rceil \ .$$

The case $\tau = 1$ is covered by [23, Thm. 6], where it is shown that one can compute in polynomial time the smallest B with $\mathcal{P}_\sigma(K \leq B) = 1$ for all schedulers σ, if such B exists. We conclude that quantile queries are polynomial-time inter-reducible with the cost problem for atomic cost formulas.

4 Cost Chains

In this section we consider the cost problems for acyclic and general cost chains. Even in the general case we obtain PSPACE membership, avoiding the EXP upper bound from Prop. 1.

Acyclic Cost Chains. The complexity class PP [10] can be defined as the class of languages L that have a probabilistic polynomial-time bounded Turing machine M_L such that for all words x one has $x \in L$ if and only if M_L accepts x with probability at least $1/2$. The class PP includes NP [10], and Toda's theorem states that P^{PP} contains the polynomial-time hierarchy [21]. We show that the cost problem for acyclic cost chains is PP-complete.

Theorem 3. *The cost problem for acyclic cost chains is in PP. It is PP-hard under polynomial-time Turing reductions, even for atomic cost formulas.*

Proof (sketch). To show membership in PP, we construct a probabilistic Turing machine that simulates the acyclic cost chain, and keeps track of the currently accumulated cost on the tape. For the lower bound, it follows from [14, Prop. 4] that an instance of the KTH LARGEST SUBSET problem can be reduced to a cost problem for acyclic cost chains with atomic cost formulas. We show in [11, Thm. 3] that this problem is PP-hard under polynomial-time Turing reductions. □

PP-hardness strengthens the NP-hardness result from [14] substantially: by Toda's theorem it follows that any problem in the polynomial-time hierarchy can be solved by a deterministic polynomial-time bounded Turing machine that has oracle access to the cost problem for acyclic cost chains.

General Cost Chains. For the PP upper bound in Thm. 3, the absence of cycles in the control graph seems essential. Indeed, we can use cycles to show hardness for the POSSLP problem, suggesting that the acyclic and the general case have different complexity. POSSLP is a fundamental problem for numerical computation [1]. Given an arithmetic circuit with operators $+$, $-$, $*$, inputs 0 and 1, and a designated output gate,

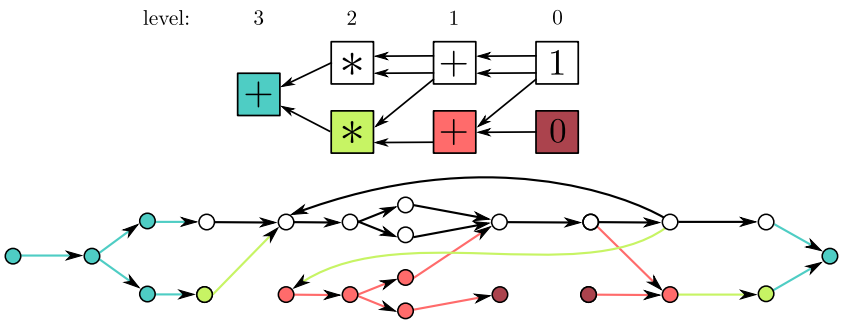

Fig. 1. Top: an arithmetic circuit in normal form. Bottom: a DFA (omitting input letters) corresponding to the construction of Prop. 5. Identical colours indicate a correspondence between gates and states.

the PosSLP problem asks whether the circuit outputs a positive integer. PosSLP is in PSpace; in fact, it lies in the 4th level of the *counting hierarchy (CH)* [1], an analogue to the polynomial-time hierarchy for classes like PP. We have the following theorem:

Theorem 4. *The cost problem for cost chains is in* PSpace *and hard for* PosSLP.

The remainder of this section is devoted to a proof sketch of this theorem. Showing membership in PSpace requires non-trivial results. There is no agreed-upon definition of probabilistic PSpace in the literature, but we can define it in analogy to PP as follows: *Probabilistic* PSpace is the class of languages L that have a probabilistic polynomial-space bounded Turing machine M_L such that for all words x one has $x \in L$ if and only if M_L accepts x with probability at least $1/2$. The cost problem for cost chains is in this class, as can be shown by adapting the argument from the beginning of the proof sketch for Thm. 3, replacing PP with probabilistic PSpace. It was first proved in [20] that probabilistic PSpace equals PSpace, hence the cost problem for cost chains is in PSpace.

For the PosSLP-hardness proof one can assume the following normal form, see the proof of [9, Thm. 5.2]: there are only $+$ and $*$ operators, the corresponding gates alternate, and all gates except those on the bottom level have exactly two incoming edges, cf. the top of Fig. 1. We write $val(g)$ for the value output by gate g. Then PosSLP asks: given an arithmetic circuit (in normal form) including gates g_1, g_2, is $val(g_1) \geq val(g_2)$?

As an intermediate step of independent interest, we show PosSLP-hardness of a problem about deterministic finite automata (DFAs). Let Σ be a finite alphabet and call a function $f : \Sigma \to \mathbb{N}$ a *Parikh function*. The *Parikh image* of a word $w \in \Sigma^*$ is the Parikh function f such that $f(a)$ is the number of occurrences of a in w. We show:

Proposition 5. *Given an arithmetic circuit including gate g, one can compute in logarithmic space a Parikh function f (in binary encoding) and a DFA $\mathcal{A}$ such that $val(g)$ equals the number of accepting computations in $\mathcal{A}$ that are labelled with words that have Parikh image f.*

The construction is illustrated in Fig. 1. It is by induction on the levels of the arithmetic circuit. A gate labelled with "+" is simulated by *branching* into the inductively

constructed gadgets corresponding to the gates this gate connects to. Likewise, a gate labelled with "$*$" is simulated by *sequentially composing* the gadgets corresponding to the gates this gate connects to. It is the latter case that may introduce cycles in the structure of the DFA. Building on this construction, by encoding alphabet letters in natural numbers encoded in binary, we then show:

Proposition 6. *Given an arithmetic circuit including gate g on odd level ℓ, one can compute in logarithmic space a cost process $\mathcal{C}$ and $T \in \mathbb{N}$ with $\mathcal{P}(K_{\mathcal{C}} = T) = val(g)/m$, where $m = \exp_2(2^{(\ell-1)/2+1} - 1) \cdot \exp_d(2^{(\ell-1)/2+1} - 3)$.*

Towards the POSSLP lower bound from Thm. 4, given an arithmetic circuit including gates g_1, g_2, we use Prop. 6 to construct two cost chains $\mathcal{C}_1 = (Q, q_1, t, \Delta)$ and $\mathcal{C}_2 = (Q, q_2, t, \Delta)$ and $T_1, T_2 \in \mathbb{N}$ such that $\mathcal{P}(K_{\mathcal{C}_i} = T_i) = val(g_i)/m$ holds for $i \in \{1, 2\}$ and for $m \in \mathbb{N}$ as in Prop. 6. Then we compute a number $H \geq T_2$ such that $\mathcal{P}(K_{\mathcal{C}_2} > H) < 1/m$. The representation of m from Prop. 6 is of exponential size. However, using Prop. 2, H depends only logarithmically on $m + 1$. We combine $\mathcal{C}_1$ and $\mathcal{C}_2$ to a cost chain $\mathcal{C} = (Q \uplus \{q_0\}, q_0, t, \widetilde{\Delta})$, where $\widetilde{\Delta}$ extends Δ by $\widetilde{\Delta}(q_0)(q_1, H + 1) = 1/2$ and $\widetilde{\Delta}(q_0)(q_2, 0) = 1/2$. By this construction, the new cost chain $\mathcal{C}$ initially either incurs cost $H + 1$ and then emulates $\mathcal{C}_1$, or incurs cost 0 and then emulates $\mathcal{C}_2$. Those possibilities have probability $1/2$ each.

Finally, we compute a suitable cost formula φ such that we have $val(g_1) \geq val(g_2)$ if and only if $\mathcal{P}(K_{\mathcal{C}} \models \varphi) \geq 1/2$, completing the logspace reduction. We remark that the structure of the formula φ, in particular the number of inequalities, is fixed. Only the involved numbers depend on the concrete instance.

5 Cost Processes

Acyclic Cost Processes. We now prove that the cost problem for acyclic cost processes is PSPACE-complete. The challenging part is to show that PSPACE-hardness even holds for *atomic* cost formulas. For our lower bound, we reduce from a generalisation of the classical SUBSETSUM problem: Given a tuple $(k_1, \ldots, k_n, T)$ of natural numbers with n even, the QSUBSETSUM *problem* asks whether the following formula is true:

$$\exists x_1 \in \{0,1\}\, \forall x_2 \in \{0,1\}\, \cdots\, \exists x_{n-1} \in \{0,1\}\, \forall x_n \in \{0,1\}\, :\, \sum_{1 \leq i \leq n} x_i k_i = T$$

Here, the quantifiers $\exists$ and $\forall$ occur in strict alternation. It is shown in [22, Lem. 4] that QSUBSETSUM is PSPACE-complete. One can think of such a formula as a turn-based game, the QSUBSETSUM *game*, played between Player Odd and Player Even. If $i \in \{1, \ldots, n\}$ is odd (even), then turn i is Player Odd's (Player Even's) turn, respectively. In turn i the respective player decides to either *take* k_i by setting $x_i = 1$, or *not* to *take* k_i by setting $x_i = 0$. Player Odd's objective is to make the sum of the taken numbers equal T, and Player Even tries to prevent that. If Player Even is replaced by a random player, then Player Odd has a strategy to win with probability 1 if and only if the given instance is a "yes" instance for QSUBSETSUM. This gives an easy PSPACE-hardness proof for the cost problem with non-atomic cost formulas $\varphi \equiv (x = T)$. In order to

strengthen the lower bound to atomic cost formulas $\varphi \equiv (x \leq B)$ we have to give Player Odd an incentive to take numbers k_i, although she is only interested in not exceeding the budget B. This challenge is addressed in our PSPACE-hardness proof.

The PSPACE-hardness result reflects the fact that the optimal strategy must take the current cost into account, not only the control state, even for atomic cost formulas. This may be somewhat counter-intuitive, as a good strategy should always "prefer small cost". But if there always existed a strategy depending *only* on the control state, one could guess this strategy in NP and invoke the PP-result of Sec. 4 in order to obtain an $\mathrm{NP}^{\mathrm{PP}}$ algorithm, implying $\mathrm{NP}^{\mathrm{PP}} = \mathrm{PSPACE}$ and hence a collapse of the counting hierarchy.

Indeed, for a concrete example, consider the acyclic cost process with $Q = \{q_0, q_1, t\}$, and $En(q_0) = \{a\}$ and $En(q_1) = \{a_1, a_2\}$, and $\Delta(q_0, a)(q_1, +1) = \frac{1}{2}$ and $\Delta(q_0, a)(q_1, +3) = \frac{1}{2}$ and $\Delta(q_1, a_1)(t, +3) = 1$ and $\Delta(q_1, a_2)(t, +6) = \frac{1}{2}$ and $\Delta(q_1, a_2)(t, +1) = \frac{1}{2}$. Consider the atomic cost formula $\varphi \equiv (x \leq 5)$. An optimal scheduler σ plays a_1 in $(q_1, 1)$ and a_2 in $(q_1, 3)$, because additional cost 3, incurred by a_1, is fine in the former but not in the latter configuration. For this scheduler σ we have $\mathcal{P}_\sigma(K \models \varphi) = \frac{3}{4}$.

Theorem 7. *The cost problem for acyclic cost processes is in* PSPACE*. It is* PSPACE*-hard, even for atomic cost formulas.*

Proof (sketch). To prove membership in PSPACE, we consider a procedure OPT that, given $(q, c) \in Q \times \mathbb{N}$ as input, computes the optimal (i.e., maximised over all schedulers) probability $p_{q,c}$ that starting from (q, c) one reaches (t, d) with $d \models \varphi$. The following procedure characterisation of $p_{q,c}$ for $q \neq t$ is crucial for $\textsc{Opt}(q, c)$:

$$p_{q,c} = \max_{a \in En(q)} \sum_{q' \in Q} \sum_{k \in \mathbb{N}} \Delta(q, a)(q', k) \cdot p_{q', c+k}$$

So $\textsc{Opt}(q, c)$ loops over all $a \in En(q)$ and all $(q', k) \in Q \times \mathbb{N}$ with $\Delta(q, a)(q', k) > 0$ and recursively computes $p_{q', c+k}$. Since the cost process is acyclic, the height of the recursion stack is at most $|Q|$. The representation size of the probabilities that occur in that computation is polynomial. To see this, consider the product D of the denominators of the probabilities occurring in the description of Δ. The encoding size of D is polynomial. All probabilities occurring during the computation are integer multiples of $1/D$. Hence computing $\textsc{Opt}(q_0, 0)$ and comparing the result with τ gives a PSPACE procedure.

For the lower bound we reduce the QSUBSETSUM problem, defined above, to the cost problem for an atomic cost formula $x \leq B$. Given an instance $(k_1, \ldots, k_n, T)$ with n is even of the QSUBSETSUM problem, we construct an acyclic cost process $\mathcal{C} = (Q, q_0, t, A, En, \Delta)$ as follows. We take $Q = \{q_0, q_2, \ldots, q_{n-2}, q_n, t\}$. Those control states reflect pairs of subsequent turns that the QSUBSETSUM game can be in. The transition rules Δ will be set up so that probably the control states $q_0, q_2, \ldots, q_n, t$ will be visited in that order, with the (improbable) possibility of shortcuts to t. For even i with $0 \leq i \leq n-2$ we set $En(q_i) = \{a_0, a_1\}$. These actions correspond to Player Odd's possible decisions of not taking, respectively taking k_{i+1}. Player Even's response is modelled by the random choice of not taking, respectively taking k_{i+2} (with probability $1/2$ each). In the cost process, taking a number k_i corresponds to incurring cost k_i. We

also add an additional cost ℓ in each transition.[4] Therefore we define our cost problem to have the atomic formula $x \leq B$ with $B := (n/2) \cdot \ell + T$. For some large number $M \in \mathbb{N}$, formally defined in App. D, we set for all even $i \leq n-2$ and for $j \in \{0, 1\}$:

$$\begin{aligned}
\Delta(q_i, a_j)(q_{i+2}, \ell + j \cdot k_{i+1}) &= (1/2) \cdot (1 - (\ell + j \cdot k_{i+1})/M) \\
\Delta(q_i, a_j)(t, \ell + j \cdot k_{i+1}) &= (1/2) \cdot (\ell + j \cdot k_{i+1})/M \\
\Delta(q_i, a_j)(q_{i+2}, \ell + j \cdot k_{i+1} + k_{i+2}) &= (1/2) \cdot (1 - (\ell + j \cdot k_{i+1} + k_{i+2})/M) \\
\Delta(q_i, a_j)(t, \ell + j \cdot k_{i+1} + k_{i+2}) &= (1/2) \cdot (\ell + j \cdot k_{i+1} + k_{i+2})/M
\end{aligned}$$

So with high probability the MDP transitions from q_i to q_{i+2}, and cost ℓ, $\ell + k_{i+1}$, $\ell + k_{i+2}$, $\ell + k_{i+1} + k_{i+2}$ is incurred, depending on the scheduler's (i.e., Player Odd's) actions and on the random (Player Even) outcome. But with a small probability, which is proportional to the incurred cost, the MDP transitions to t, which is a "win" for the scheduler as long as the accumulated cost is within budget B. We make sure that the scheduler loses if q_n is reached:

$$\Delta(q_n, a)(t, B{+}1) = 1 \qquad \text{with } En(q_n) = \{a\}$$

The MDP is designed so that the scheduler probably "loses" (i.e., exceeds the budget B); but whenever cost k is incurred, a winning opportunity with probability k/M arises. Since $1/M$ is small, the overall probability of winning is approximately C/M if total cost $C \leq B$ is incurred. In order to maximise this chance, the scheduler wants to maximise the total cost without exceeding B, so the optimal scheduler will target B as total cost.

The values for ℓ, M and τ need to be chosen carefully, as the overall probability of winning is not exactly the sum of the probabilities of the individual winning opportunities. By the "union bound", this sum is only an upper bound, and one needs to show that the sum approximates the real probability closely enough. □

General Cost Processes. We show the following theorem:

Theorem 8. *The cost problem is* EXP*-complete.*

Proof (sketch). The EXP upper bound was stated in Prop. 1. Regarding hardness, we build upon *countdown games*, "a simple class of turn-based 2-player games with discrete timing" [12]. Deciding the winner in a countdown game is EXP-complete [12]. Albeit non-stochastic, countdown games are very close to our model: two players move along edges of a graph labelled with positive integer weights and thereby add corresponding values to a succinctly encoded counter. Player 1's objective is to steer the value of the counter to a given number $T \in \mathbb{N}$, and Player 2 tries to prevent that. Our reduction from countdown games in App. D requires a small trick, as in our model the final control state t needs to be reached with probability 1 regardless of the scheduler, and furthermore, the scheduler attempts to achieve the cost target T when and only when the control state $t \in Q$ is visited. □

[4] This is for technical reasons. Roughly speaking, this prevents the possibility of reaching the full budget B before an action in control state q_{n-2} is played.

The Cost-Utility Problem. MDPs with *two* non-negative and non-decreasing integer counters, viewed as cost and utility, respectively, were considered in [2, 3]. Specifically, those works consider problems such as computing the minimal cost C such that the probability of gaining at least a given utility U is at least τ. Possibly the most fundamental of those problems is the following: the *cost-utility problem* asks, given an MDP with both cost and utility, and numbers $C, U \in \mathbb{N}$, whether one can, with probability 1, gain utility at least U using cost at most C. Using essentially the proof of Thm. 8 we show:

Corollary 9. *The cost-utility problem is* EXP*-complete.*

The Universal Cost Problem. We defined the cost problem so that it asks whether *there exists* a scheduler σ with $\mathcal{P}_\sigma(K_\mathcal{C} \models \varphi) \geq \tau$. A natural variant is the *universal cost problem*, which asks whether *for all* schedulers σ we have $\mathcal{P}_\sigma(K_\mathcal{C} \models \varphi) \geq \tau$. Here the scheduler is viewed as an adversary which tries to prevent the satisfaction of φ. Clearly, for cost chains the cost problem and the universal cost problem are equivalent. Moreover, Thms. 7 and 8 hold analogously in the universal case.

Theorem 10. *The universal cost problem for acyclic cost processes is in* PSPACE*. It is* PSPACE*-hard, even for atomic cost formulas. The universal cost problem is* EXP*-complete.*

Proof (sketch). The universal cost problem and the complement of the cost problem (and their acyclic versions) are interreducible in logarithmic space by essentially negating the cost-formulas. The only problem is that if φ is an *atomic* cost formula, then $\neg\varphi$ is not an atomic cost formula. However, the PSPACE-hardness proof from Thm. 7 can be adapted, cf. App. E. □

6 Conclusions and Open Problems

In this paper we have studied the complexity of analysing succinctly represented stochastic systems with a single non-negative and only increasing integer counter. We have improved the known complexity bounds significantly. Among other results, we have shown that the cost problem for Markov chains is in PSPACE and both hard for PP and the POSSLP problem. It would be fascinating and potentially challenging to prove either PSPACE-hardness or membership in the counting hierarchy: the problem does not seem to lend itself to a PSPACE-hardness proof, but the authors are not aware of natural problems, except BITSLP [1], that are in the counting hierarchy and known to be hard for both PP and POSSLP.

Regarding acyclic and general MDPs, we have proved PSPACE-completeness and EXP-completeness, respectively. Our results leave open the possibility that the cost problem for atomic cost formulas is not EXP-hard and even in PSPACE. The technique described in the proof sketch of Thm. 7 cannot be applied to general cost processes, because there we have to deal with paths of exponential length, which, informally speaking, have double-exponentially small probabilities. Proving hardness in an analogous way would thus require probability thresholds τ of exponential representation size.

Acknowledgements. The authors would like to thank Andreas Göbel for valuable hints, Christel Baier and Sascha Klüppelholz for thoughtful feedback on an earlier version of this paper, and the anonymous referees for their helpful comments.

References

1. 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.
2. C. Baier, M. Daum, C. Dubslaff, J. Klein, and S. Klüppelholz. Energy-utility quantiles. In *Proc. NFM*, volume 8430 of *LNCS*, pages 285–299. Springer, 2014.
3. C. Baier, C. Dubslaff, and S. Klüppelholz. Trade-off analysis meets probabilistic model checking. In *Proc. CSL-LICS*, pages 1:1–1:10. ACM, 2014.
4. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
5. C. Baier, J. Klein, S. Klüppelholz, and S. Märcker. Computing conditional probabilities in Markovian models efficiently. In *Proc. TACAS*, volume 8413 of *LNCS*, pages 515–530, 2014.
6. T. Brázdil, V. Brožek, K. Etessami, and A. Kučera. Approximating the termination value of one-counter MDPs and stochastic games. *Inform. Comput.*, 222(0):121 – 138, 2013.
7. T. Brázdil, V. Brožek, K. Etessami, A. Kučera, and D. Wojtczak. One-counter Markov decision processes. In *Proc. SODA*, pages 863–874. SIAM, 2010.
8. 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*, volume 25 of *LIPIcs*, pages 199–213, 2014.
9. K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1):1:1–1:66, 2009.
10. J. Gill. Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, 6(4):675–695, 1977.
11. C. Haase and S. Kiefer. The complexity of the Kth largest subset problem and related problems. Technical Report at http://arxiv.org/abs/1501.06729, 2015.
12. M. Jurdziński, J. Sproston, and F. Laroussinie. Model checking probabilistic timed automata with one or two clocks. *Log. Meth. Comput. Sci.*, 4(3):12, 2008.
13. R. E. Ladner. Polynomial space counting problems. *SIAM J. Comput.*, 18(6):1087–1097, 1989.
14. F. Laroussinie and J. Sproston. Model checking durational probabilistic systems. In *Proc. FoSSaCS*, volume 3441 of *LNCS*, pages 140–154. Springer, 2005.
15. M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *J. ACM*, 47(4):681–720, 2000.
16. C. H. Papadimitriou. Games against nature. In *Proc. FOCS*, pages 446–450, 1983.
17. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 2008.
18. M. Randour, J.-F. Raskin, and O. Sankur. Percentile queries in multi-dimensional Markov decision processes. In *Proc. CAV*, LNCS, 2015.
19. M. Randour, J.-F. Raskin, and O. Sankur. Variations on the stochastic shortest path problem. In *Proc. VMCAI*, volume 8931 of *LNCS*, pages 1–18, 2015.
20. J. Simon. On the difference between one and many. In *Proc. ICALP*, volume 52 of *LNCS*, pages 480–491. Springer, 1977.
21. S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.
22. S. Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1–3):211–229, 2006.
23. M. Ummels and C. Baier. Computing quantiles in Markov reward models. In *Proc. FoSSaCS*, volume 7794 of *LNCS*, pages 353–368. Springer, 2013.

A Proofs of Section 2

Proposition 11. *Let $\mathcal{C}$ be a cost process, φ a cost formula with $n_0 \not\models \varphi$ and $n_1 \models \varphi$ for some $n_0, n_1 \in \mathbb{N}$, and $\tau \in [0,1]$. One can construct in logarithmic space a cost process $\mathcal{C}'$ such that the following holds: There is a scheduler σ for $\mathcal{C}$ with $\mathcal{P}_\sigma(K_\mathcal{C} \models \varphi) \geq \tau$ if and only if there is a scheduler σ' for $\mathcal{C}'$ with $\mathcal{P}_{\sigma'}(K_{\mathcal{C}'} \models \varphi) \geq 1/2$. Moreover, $\mathcal{C}'$ is a cost chain if $\mathcal{C}$ is.*

Proof. Let $\tau < 1/2$. Define $p := (1/2 - \tau)/(1-\tau)$. To construct $\mathcal{C}'$ from $\mathcal{C}$, add a new initial state s_{00} with exactly one enabled action, say a, and set $\Delta(s_{00}, a)(t, n_1) = p$ and $\Delta(s_{00}, a)(s_0, 0) = 1 - p$. In a straightforward sense any scheduler for $\mathcal{C}$ can be viewed as a scheduler for $\mathcal{C}'$ and vice versa. Thus for any scheduler σ we have $\mathcal{P}'_\sigma(K \models \varphi) = p + (1-p) \cdot \mathcal{P}_\sigma(K \models \varphi)$. The statement of the proposition now follows from a simple calculation.

Now let $\tau > 1/2$. Define $p := 1/(2\tau)$. In a similar way as before, add a new initial state s_{00} with exactly one enabled action a, and set $\Delta(s_{00}, a)(t, n_0) = 1 - p$ and $\Delta(s_{00}, a)(s_0, 0) = p$. Thus we have $\mathcal{P}'_\sigma(K \models \varphi) = p \cdot \mathcal{P}_\sigma(K \models \varphi)$, and the statement of the proposition follows. □

B Proofs of Section 3

In this section we prove Prop. 2 from the main text:

Proposition 2. *Suppose $0 \leq \tau < 1$. Let p_{min} be the smallest non-zero probability and k_{max} be the largest cost in the description of the cost process. Then $\mathcal{P}_\sigma(K \leq B) \geq \tau$ holds for all schedulers σ, where*

$$B := k_{max} \cdot \left\lceil |Q| \cdot \left(-\ln(1-\tau)/p_{min}^{|Q|} \; + \; 1 \right) \right\rceil \; .$$

Proof. Define $n := |Q|$. If $p_{min} = 1$, then by our assumption on the almost-sure reachability of t, the state t will be reached within n steps, and the statement of the proposition follows easily. So we can assume $p_{min} < 1$ for the rest of the proof.

Let $j \in \mathbb{N}$ be the smallest integer with

$$j \geq n \cdot \left(\frac{-\ln(1-\tau)}{p_{min}^n} + 1 \right) .$$

It follows:

$$\begin{aligned} \left\lfloor \frac{j}{n} \right\rfloor &\geq \frac{-\ln(1-\tau)}{p_{min}^n} \\ &\geq \frac{\ln(1-\tau)}{\ln(1-p_{min}^n)} \qquad \text{(as } x \leq -\ln(1-x) \text{ for } x < 1) \end{aligned} \tag{1}$$

For $i \in \mathbb{N}$ and $q \in Q$ and a scheduler σ, define $p_i(q, \sigma)$ as the probability that, if starting in q and using the scheduler σ, more than i steps are required to reach the target state t. Define $p_i := \max\{p_i(q,\sigma) : q \in Q,\ \sigma \text{ a scheduler}\}$. By our assumption on the

almost-sure reachability of t, regardless of the scheduler, there is always a path to t of length at most n. This path has probability at least p^n_{min}, so $p_n \leq 1 - p^n_{min}$. If a path of length $\ell \cdot n$ does not reach t, then none of its ℓ consecutive blocks of length n reaches t, so we have $p_{\ell \cdot n} \leq p^\ell_n$. Hence we have:

$$\begin{aligned}
p_j &\leq p_{\lfloor j/n \rfloor \cdot n} && \text{(as } p_i \geq p_{i+1} \text{ for all } i \in \mathbb{N}\text{)} \\
&\leq p_n^{\lfloor j/n \rfloor} && \text{(as argued above)} \\
&\leq (1 - p^n_{min})^{\lfloor j/n \rfloor} && \text{(as argued above)} \\
&= \exp\left(\ln(1 - p^n_{min}) \cdot \lfloor j/n \rfloor\right) \\
&\leq 1 - \tau && \text{(by (1))} \qquad (2)
\end{aligned}$$

Denote by T the random variable that assigns to a run the "time" (i.e., the number of steps) to reach t from s_0. Then we have for all schedulers σ:

$$\begin{aligned}
\mathcal{P}_\sigma(K \leq B) &= \mathcal{P}_\sigma(K \leq j \cdot k_{max}) && \text{(by the definition of } B\text{)} \\
&\geq P_\sigma(T \leq j) && \text{(each step costs at most } k_{max}\text{)} \\
&= 1 - P_\sigma(T > j) \\
&\geq 1 - p_j && \text{(by the definition of } T \text{ and } p_i\text{)} \\
&\geq \tau && \text{(by (2))} ,
\end{aligned}$$

as claimed. □

C Proofs of Section 4

C.1 Proof of Thm. 3

In this section we prove Thm. 3 from the main text:

Theorem 3. *The cost problem for acyclic cost chains is in PP. It is PP-hard under polynomial-time Turing reductions, even for atomic cost formulas.*

Proof. First we prove membership in PP. Recall from the main text that the class PP can be defined as the class of languages L that have a probabilistic polynomial-time bounded Turing machine M_L such that for all words x one has $x \in L$ if and only if M_L accepts x with probability at least $1/2$, see [10] and note that PP is closed under complement [10]. By Prop. 11 it suffices to consider an instance of the cost problem with $\tau = 1/2$. The problem can be decided by a probabilistic Turing machine that simulates the cost chain as follows: The Turing machine keeps track of the control state and the cost, and branches according to the probabilities specified in the cost chain. It accepts if and only if the accumulated cost satisfies φ. Note that the acyclicity of the cost chain guarantees the required polynomial time bound. This proof assumes that the probabilistic Turing machine has access to coins that are biased according to the probabilities in the cost chain. As we show in [11, Lem. 1], this can indeed be assumed for probabilistic polynomial-time bounded Turing machines.

As stated in the main text, the lower bound follows from combining the result from [14, Prop. 4] with [11, Thm. 3]. □

C.2 Proof of Thm. 4

In this section we prove Thm. 4 from the main text:

Theorem 4. *The cost problem for cost chains is in* PSPACE *and hard for* POSSLP.

First we give details on the upper bound. Then we provide proofs of the statements from the main text that pertain to the POSSLP lower bound.

Proof of the Upper Bound in Thm. 4. We show that the cost problem for cost chains is in PSPACE. As outlined in the main text we use the fact that PSPACE equals probabilistic PSPACE. The cost problem for cost chains is in this class. This can be shown in the same way as we showed in Thm. 3 that the cost problem for acyclic cost chains is in PP. More concretely, given an instance of the cost problem for cost chains, we construct in logarithmic space a probabilistic PSPACE Turing machine that simulates the cost chain and accepts if and only if the accumulated cost K satisfies the given cost formula.

The fact that (this definition of) probabilistic PSPACE equals PSPACE was first proved in [20]. A simpler proof can be obtained using a result by Ladner [13] that states that #PSPACE equals FPSPACE, see [13] for definitions. This was noted implicitly, e.g., in [15, Thm. 5.2]. We remark that the class PPSPACE defined in [16] also equals PSPACE, but its definition (which is in terms of stochastic games) is different.

Proof of Prop. 5. Here, we give a formal definition and proof of the construction outlined in the main text which allows for computing the value of an arithmetic circuit as the number of paths in a DFA with a certain Parikh image. First, we formally define the notations informally used in the main text.

We first introduce arithmetic circuits and at the same time take advantage of a normal form that avoids gates labelled with "$-$". This normal form was established in the proof of [9, Thm. 5.2]. An *arithmetic circuit* is a directed acyclic graph $G = (V, E)$ whose leaves are labelled with constants "0" and "1", and whose vertices are labelled with operators "$+$" and "$*$". Subsequently, we refer to the elements of V as *gates*. With every gate we associate a level starting at 0 with leaves. For levels greater than zero, gates on odd levels are labelled with "$+$" and on even levels with "$*$". Moreover, all gates on a level greater than zero have exactly two incoming edges from the preceding level. The upper part of Fig. 1 illustrates an arithmetic circuit in this normal form. We can associate with every gate $v \in V$ a non-negative integer $val(v)$ in the obvious way. In this form, the POSSLP problem asks, given an arithmetic circuit $G = (V, E)$ and two gates $v_1, v_2 \in V$, whether $val(v_1) \geq val(v_2)$ holds.

Regarding the relevant definitions of Parikh images, let $\mathcal{A} = (Q, \Sigma, \Delta)$ be a DFA such that Q is a finite set of *control states*, $\Sigma = \{a_1, \ldots, a_k\}$ is a *finite alphabet*, and $\Delta \subseteq Q \times \Sigma \times Q$ is the set of *transitions*. A *path* π in $\mathcal{A}$ is a sequence of transitions $\pi = \delta_1 \cdots \delta_n \in \Delta^*$ such that $\delta_i = (q_i, a_i, q_i')$ and $\delta_{i+1} = (q_{i+1}, a_{i+1}, q_{i+1}')$ implies $q_i' = q_{i+1}$ for all $1 \leq i < n$. Let $q, q' \in Q$, we denote by $\Pi(\mathcal{A}, q, q')$ the set of all paths starting in q and ending in q'. In this paper, a *Parikh function* is a function $f : \Sigma \to \mathbb{N}$. The *Parikh image* of a path π, denoted $parikh(\pi)$, is the unique Parikh function counting for every $a \in \Sigma$ the number of times a occurs on a transition in π.

The following statement of Prop. 5 makes the one given in the main text more precise.

Proposition 5. *Let $G = (V, E)$ be an arithmetic circuit and $v \in V$. There exists a log-space computable DFA $\mathcal{A} = (Q, \Sigma, \Delta)$ with distinguished control states $q, q' \in Q$ and a Parikh function $f : \Sigma \to \mathbb{N}$ such that*

$$val(v) = |\{\pi \in \Pi(\mathcal{A}, q, q') : parikh(\pi) = f\}|.$$

Proof. We construct $\mathcal{A}$ by induction on the number of levels of V. For every level i, we define an alphabet Σ_i and a Parikh function $f_i : \Sigma_i \to \mathbb{N}$. As an invariant, $\Sigma_i \subseteq \Sigma_{i+1}$ holds for all levels i. Subsequently, denote by $V(i)$ all gates on level i. For every $v \in V(i)$, we define a DFA $\mathcal{A}_v$ such that each $\mathcal{A}_v$ has two distinguished control locations $in(\mathcal{A}_v)$ and $out(\mathcal{A}_v)$. The construction is such that

$$val(v) = |\{\pi \in \Pi(\mathcal{A}_v, in(\mathcal{A}_v), out(\mathcal{A}_v)) : parikh(\pi) = f_i\}|. \tag{3}$$

For technical convenience, we allow transitions to be labelled with subsets $S \subseteq \Sigma$ which simply translates into an arbitrary chain of transitions such that each $a \in S$ occurs exactly once along this chain. We now proceed with the details of the construction starting with gates on level 0.

With no loss of generality we may assume that there are two gates v and w on level 0 labelled with 0 and 1, respectively. Let $\Sigma_0 = \{a\}$ for some letter a. The DFA $\mathcal{A}_v$ and $\mathcal{A}_w$ over Σ_0 is defined as follows: $\mathcal{A}_w$ has a single transition connecting $in(\mathcal{A}_w)$ with $out(\mathcal{A}_w)$ labelled with a, whereas $\mathcal{A}_v$ does not have this transition. Setting $f_0(a) = 1$, it is easily checked that (3) holds for those DFA.

For level $i+1$, we define $\Sigma_{i+1} = \Sigma_i \uplus \{a_v, b_v, c_v : v \in V(i+1)\}$. Let $v \in V(i+1)$ be a gate on level $i+1$ such that v has incoming edges from u and w. Let $\mathcal{A}_u = (Q_u, \Sigma_i, \Delta_u)$ and $\mathcal{A}_w = (Q_w, \Sigma_i, \Delta_w)$ be the DFA representing u and w. Let Q_v be a set of fresh control states. We define $\mathcal{A}_v = (Q_v \cup Q_u \cup Q_w, \Sigma_{i+1}, \Delta_v \cup \Delta_u \cup \Delta_w)$. The particularities of the construction depend on the type of v.

If $i+1$ is odd, i.e., the gates on this level are labelled with "+", then apart from the control states $in(\mathcal{A}_v)$ and $out(\mathcal{A}_v)$, the set Q_v contains three additional control states q, q_1, q_2. Further we set $\Delta_v = \{\delta_1, \ldots, \delta_7\}$ such that

- $\delta_1 = (in(\mathcal{A}_v), S_v, q)$, where $S_v = \{a_w, b_w, c_w : w \in V(i+1), v \neq w\}$;
- $\delta_2 = (q, a_v, q_1)$ and $\delta_3 = (q, b_v, q_2)$;
- $\delta_4 = (q_1, b_v, in(\mathcal{A}_u))$ and $\delta_5 = (q_2, a_v, in(\mathcal{A}_w))$; and
- $\delta_6 = (out(\mathcal{A}_u), c_v, out(\mathcal{A}))$ and $\delta_7 = (out(\mathcal{A}_w), c_v, out(\mathcal{A}))$.

Informally speaking, we simply branch at q into $\mathcal{A}_u$ and $\mathcal{A}_w$, and this in turn enforces that the number of paths in $\Pi(\mathcal{A}_v, in(\mathcal{A}_v), out(\mathcal{A}_v))$ on which a_v occurs once equals the sum of $val(u)$ and $val(w)$. The reason behind using *both* a_v and b_v is that it ensures that the case $u = w$ is handled correctly. Setting $f_{i+1}(a) = 1$ if $a \in \Sigma_{i+1} \setminus \Sigma_i$, and

$f_{i+1}(a) = f_i(a)$ otherwise, we consequently have that (3) holds since

$$\begin{aligned}
&|\{\pi \in \Pi(\mathcal{A}_v, in(\mathcal{A}_v), out(\mathcal{A}_v)) : parikh(\pi) = f_{i+1}\}| \\
&= |\{\pi \in \Pi(\mathcal{A}_u, in(\mathcal{A}_u), out(\mathcal{A}_u)) : parikh(\pi) = f_i\}| + \\
&\quad + |\{\pi \in \Pi(\mathcal{A}_w, in(\mathcal{A}_w), out(\mathcal{A}_w)) : parikh(\pi) = f_i\}| \\
&= val(u) + val(w) \\
&= val(v).
\end{aligned}$$

The case of $i + 1$ being even can be handled analogously, but instead of using branching we use sequential composition in order to simulate the computation of a gate labelled with "$*$". Apart from the control states $in(\mathcal{A}_v)$ and $out(\mathcal{A}_v)$, the set Q_v contains an additional control state q. Further we set $\Delta_v = \{\delta_1, \ldots, \delta_4\}$ such that

- $\delta_1 = (in(\mathcal{A}_v), S_v, q)$, where $S_v = \{a_w, b_w, c_w : w \in V(i+1), v \neq w\}$;
- $\delta_2 = (q, a_v, in(\mathcal{A}_u))$;
- $\delta_3 = (out(\mathcal{A}_u), b_v, in(\mathcal{A}_w))$; and
- $\delta_4 = (out(\mathcal{A}_w), c_v, out(\mathcal{A}_v))$.

A difference to the case where $i + 1$ is odd is that via the definition of f_{i+1} we have to allow for paths that can traverse *both* $\mathcal{A}_u$ and $\mathcal{A}_w$. Consequently, we define $f_{i+1}(a) = 1$ if $a \in \Sigma_{i+1} \setminus \Sigma_i$, and $f_{i+1}(a) = 2f_i(a)$ otherwise. Similarly as above, (3) holds since

$$\begin{aligned}
&|\{\pi \in \Pi(\mathcal{A}_v, in(\mathcal{A}_v), out(\mathcal{A}_v)) : parikh(\pi) = f_{i+1}\}| \\
&= |\{\pi \in \Pi(\mathcal{A}_u, in(\mathcal{A}_u), out(\mathcal{A}_u)) : parikh(\pi) = f_i\}| \cdot \\
&\quad \cdot |\{\pi \in \Pi(\mathcal{A}_w, in(\mathcal{A}_w), out(\mathcal{A}_w)) : parikh(\pi) = f_i\}| \\
&= val(u) \cdot val(w) \\
&= val(v).
\end{aligned}$$

Due to the inductive nature of the construction, the cautious reader may on the first sight cast doubt that the computation of $\mathcal{A}_v$ and f can be performed in logarithmic space. However, a closer look reveals that the graph underlying $\mathcal{A}_v$ has a simple structure and its list of edges can be constructed without prior knowledge of the DFA on lower levels. Likewise, even though f contains numbers which are exponential in the number of levels of G, the structure of f is simple and only contains numbers which are powers of two, and hence f is computable in logarithmic space as well. □

Proof of Prop. 6. The following statement of Prop. 6 makes the one given in the main text more precise.

Proposition 6. *Let $G = (V, E)$ be an arithmetic circuit. Let $v \in V$ be a gate on level ℓ with odd ℓ. There exist a log-space computable cost process $\mathcal{C}$ and $T \in \mathbb{N}$ with $\mathcal{P}(K_{\mathcal{C}} = T) = val(v)/m$, where $m = \exp_2(2^{(\ell-1)/2+1} - 1) \cdot \exp_d(2^{(\ell-1)/2+1} - 3)$.*

For a clearer proof structure we define an intermediate formalism between DFA and cost chains. A *typed cost chain* $\mathcal{T} = (Q, q_0, t, \Gamma, \Delta)$ is similar to a cost chain, but with costs (i.e., natural numbers) replaced with functions $\Gamma \to \mathbb{N}$. The intuition is that instead

of a single cost, a typed cost chain keeps track of several types of cost, and each type is identified with a symbol from Γ. More precisely, Q is a finite set of control states, $q_0 \in Q$ is the initial control state, t is the target control state, Γ is a finite alphabet, and $\Delta : Q \to dist(Q \times \mathbb{N}^\Gamma)$ is a probabilistic transition function.

A typed cost chain $\mathcal{T}$ induces a Markov chain in the same way as a cost chain does, but the state space is $Q \times \mathbb{N}^\Gamma$ rather than $Q \times \mathbb{N}$. Formally, $\mathcal{T}$ induces the Markov chain $\mathcal{D}_\mathcal{T} = (Q \times \mathbb{N}^\Gamma, (q_0, \mathbf{0}), \delta)$, where by $\mathbf{0}$ we mean the function $c : \Gamma \to \mathbb{N}$ with $c(a) = 0$ for all $a \in \Gamma$, and $\delta(q, c)(q', c') = \Delta(q)(q', c' - c)$ holds for all $q, q' \in Q$ and $c, c' \in \mathbb{N}^\Gamma$, where by $c' - c$ we mean $c'' : \Gamma \to \mathbb{N}$ with $c''(a) = c'(a) - c(a)$ for all $a \in \Gamma$. As before, we assume that the target control state t is almost surely reached. We write $K_\mathcal{T}$ for the (multi-dimensional) random variable that assigns a run in $\mathcal{D}_\mathcal{T}$ the typed cost $c : \Gamma \to \mathbb{N}$ that is accumulated upon reaching t.

Lemma 12. *Let $G = (V, E)$ be an arithmetic circuit. Let $v \in V$ be a gate on level ℓ with odd ℓ. Let $d = |V| + 1$. There exist a log-space computable typed cost chain $\mathcal{T} = (Q, q_0, t, \Gamma, \Delta)$ and $c : \Gamma \to \mathbb{N}$ such that $\mathcal{P}(K_\mathcal{T} = c) = val(v)/m$, where*

$$m = \exp_2(2^{(\ell-1)/2+1} - 1) \cdot \exp_d(2^{(\ell-1)/2+1} - 3)\,.$$

Proof. With no loss of generality we may assume that the maximum level of V is ℓ and that v is the only gate on level ℓ. The idea is to translate the DFA obtained in Prop. 5 into a suitable typed cost chain. Subsequently, we refer to the terminology used in the proof of Prop. 5.

Let $\mathcal{A} = (Q, \Sigma, \Delta)$ be the DFA, $q_0 = in(\mathcal{A}_v)$, $t = out(\mathcal{A}_v)$, and $f : \Sigma \to \mathbb{N}$ be the Parikh function obtained from Prop. 5. We define $\Gamma = \Sigma \uplus \{e_j : 1 \leq j \leq d\}$ and alter $\mathcal{A}$ as follows:

- for the gate $w \in V$ on level 0 labelled with 0, we add an edge from $in(\mathcal{A}_w)$ to t labelled with e_1; and
- for every $w \in V$ such that $w \neq v$, we add k edges labelled with $e_1, \ldots, e_k$ from $out(\mathcal{A}_w)$ to t, where k is the difference between d and the number of outgoing edges from $out(\mathcal{A}_w)$.

The DFA $\mathcal{A}' = (Q, \Gamma, \Delta')$ obtained from this construction has the property that t can be reached from any control state, and that the number of outgoing edges from any $out(\mathcal{A}_w)$ for $w \neq v$ is uniform. Finally, we define $c : \Gamma \to \mathbb{N}$ such that c coincides with f for all $a \in \Sigma$ and $c(e_j) = 0$ for all $1 \leq j \leq k$. The intuition behind the e_j is that they indicate errors, and once an edge with an e_j is traversed it is impossible to reach t with Parikh image c. Thus, in particular property (3) is preserved in $\mathcal{A}'$.

We now transform $\mathcal{A}'$ into a typed cost chain $\mathcal{T}$. Subsequently, for $a \in \Gamma$ let $c_a : \Gamma \to \{0, 1\}$ be the function such that $c_a(b) = 1$ if $b = a$ and $c_a(b) = 0$ otherwise. For our transformation, we perform the following steps:

- every alphabet letter $a \in \Gamma$ labelling a transition of $\mathcal{A}'$ is replaced by c_a;
- the probability distribution over edges is chosen uniformly; and
- a self-loop labelled with $\mathbf{0}$ and probability 1 is added at t.

We observe that the transition probabilities of $\mathcal{T}$ are either $1/d$, $1/2$ or 1. Since t can be reached from any control state, it is eventually reached with probability 1.

For every level i and every $w \in V(i)$, let p_w denote the probability that, starting from $in(\mathcal{A}_w)$, the control state $out(\mathcal{A}_w)$ is reached *and* typed cost c_i is accumulated. Here, c_i refers to the Parikh function f_i constructed in the proof of Prop. 5, where we assert that $c_i(a) = 0$ for all $a \in \Gamma$ on which the "original" f_i is undefined. Since $t = out(\mathcal{A}_v)$ is almost surely reached from $q_0 = in(\mathcal{A}_v)$, we have $p_v = \mathcal{P}(K_{\mathcal{T}} = c_\ell)$. So in order to prove the lemma, it suffices to prove for all $i \in \mathbb{N}$:

$$p_w = \frac{val(w)}{m(i)} \qquad \text{for all } w \in V(i),$$

where

$$m(i) = \begin{cases} \exp_2(2^{i/2+1} - 2) \cdot \exp_d(2^{i/2+1} - 4) & \text{if } i \text{ is even} \\ \exp_2(2^{(i-1)/2+1} - 1) \cdot \exp_d(2^{(i-1)/2+1} - 3) & \text{if } i \text{ is odd}\,. \end{cases}$$

We proceed by induction on the level i. Let $i = 0$. If w is labelled with 1 then there is exactly one outgoing transition from $in(\mathcal{A}_w)$, and this transition goes to $out(\mathcal{A}_w)$ and incurs cost c_0. So we have $p_w = 1$ as required. If w is labelled with 0, then the only outgoing transition from $in(\mathcal{A}_w)$ incurs cost c with $c(e_1) = 1$, hence $p_w = 0$.

For the induction step, let $i \geq 0$. Let $w \in V(i+1)$ and let $u, u' \in V(i)$ be the gates connected to w. If $i + 1$ is odd then w is labelled with "+", and by the construction of $\mathcal{A}_w$ and the transformation above we have

$$\begin{aligned} p_w &= \frac{1}{2} \cdot \frac{1}{d} \cdot (p_u + p_{u'}) \\ &= \frac{1}{2} \cdot \frac{1}{d} \cdot \frac{val(u) + val(u')}{\exp_2(2^{i/2+1} - 2) \cdot \exp_d(2^{i/2+1} - 4)} && \text{by the ind. hypoth.} \\ &= \frac{val(u) + val(u')}{\exp_2(2^{i/2+1} - 1) \cdot \exp_d(2^{i/2+1} - 3)} \\ &= \frac{val(w)}{m(i+1)}. \end{aligned}$$

The factor $1/2$ is the probability of branching into $in(\mathcal{A}_u)$ or $in(\mathcal{A}_{u'})$, and $1/d$ is the probability that when leaving $out(\mathcal{A}_u)$ respectively $out(\mathcal{A}_{u'})$, the transition to $out(\mathcal{A}_w)$ is taken.

Otherwise, if $i+1$ is even, we have

$$\begin{aligned} p_w &= \frac{1}{d^2} \cdot p_u \cdot p_{u'} \\ &= \frac{1}{d^2} \cdot \frac{val(u) \cdot val(u')}{(\exp_2(2^{(i-1)/2+1}-1) \cdot \exp_d(2^{(i-1)/2+1}-3))^2} \quad \text{by the ind. hypoth.} \\ &= \frac{1}{d^2} \cdot \frac{val(u) \cdot val(u')}{\exp_2(2^{(i-1)/2+2}-2) \cdot \exp_d(2^{(i-1)/2+2}-6)} \\ &= \frac{val(u) \cdot val(u')}{\exp_2(2^{(i+1)/2+1}-2) \cdot \exp_d(2^{(i+1)/2+1}-4)} \\ &= \frac{val(w)}{m(i+1)}. \end{aligned}$$

Here, $1/d^2$ is the probability that when leaving $out(\mathcal{A}_u)$ the transition to $in(\mathcal{A}_{u'})$ is taken, and that when leaving $out(\mathcal{A}_{u'})$ the transition to $out(\mathcal{A}_w)$ is taken. □

In order to complete the proof of Prop. 6, we now show how a typed cost chain can be transformed into a cost chain. The idea underlying the construction in the next lemma is that we can encode alphabet symbols into the digits of natural numbers represented in a suitable base.

Lemma 13. *Let Γ be a finite alphabet, and let $c, c_1, \ldots, c_n : \Gamma \to \mathbb{N}$ be functions represented as tuples with numbers encoded in binary. There exists a log-space computable homomorphism $h : \mathbb{N}^\Gamma \to \mathbb{N}$ such that for all $\lambda_1, \ldots, \lambda_n \in \mathbb{N}$ we have*

$$\sum_{i=1}^{n} \lambda_i c_i = c \iff \sum_{i=1}^{n} \lambda_i h(c_i) = h(c)\,.$$

Proof. Let $\Gamma = \{a_0, \ldots, a_{k-1}\}$, $m = \sum_{a\in\Gamma} c(a)$, and $b = m+1$. We define $h : \mathbb{N}^\Gamma \to \mathbb{N}$ as

$$h(d) = d(a_0) \cdot b^0 + d(a_1) \cdot b^1 + \cdots + d(a_{k-1}) \cdot b^{k-1} + \left(\sum\nolimits_{a\in\Gamma} d(a)\right) \cdot b^k.$$

The homomorphism h encodes any $d : \Gamma \to \mathbb{N}$ into the k least significant digits of a natural number represented in base b, and the $k+1$-th digit serves as a check digit.

Suppose $\sum_{i=1}^{n} \lambda_i c_i = c$. Then

$$\begin{aligned} \sum_{i=1}^{n} \lambda_i h(c_i) &= \sum\nolimits_{j=0}^{k-1} \left(\sum\nolimits_{i=1}^{n} \lambda_i c_i(a_j)\right) \cdot b^j + \left(\sum\nolimits_{a\in\Gamma} \sum\nolimits_{i=1}^{n} \lambda_i c_i(a)\right) \cdot b^k \\ &= \sum\nolimits_{j=0}^{k-1} c(a_j) \cdot b^j + \sum\nolimits_{a\in\Gamma} c(a) \cdot b^k \\ &= h(c). \end{aligned}$$

Conversely, assume that $\sum_{i=1}^{n} \lambda_i h(c_i) = h(c)$. By definition of h, the check digit $k+1$ ensures that

$$\sum_{a\in\Gamma} \sum_{i=1}^{n} \lambda_i c_i(a) = \sum_{a\in\Gamma} c(a) = m < b.$$

Thus, in particular for a fixed $a_j \in \Sigma$ we have

$$\sum_{i=1}^{n} \lambda_i c_i(a_j) < b.$$

But now, since $\sum_{i=1}^{n} \lambda_i h(c_i) = h(c)$ we have

$$\sum_{i=1}^{n} \lambda_i c_i(a_j) = c(a_j),$$

and consequently $\sum_{i=1}^{n} \lambda_i c_i = c$. □

By replacing every typed cost function c in $\mathcal{T}$ with $h(c)$, an easy application of Lem. 13 now yields the following corollary.

Corollary 14. *Let $\mathcal{T} = (Q, q_0, t, \Gamma, \Delta)$ be a typed cost chain and $c : \Gamma \to \mathbb{N}$. There exist a log-space computable cost chain $\mathcal{C} = (Q, q_0, t, \delta)$ and $n \in \mathbb{N}$ such that*

$$\mathcal{P}(K_{\mathcal{T}} = c) = \mathcal{P}(K_{\mathcal{C}} = n).$$

Together with Lem. 12, this completes the proof of Prop. 6.

Proof of the Lower Bound in Thm. 4. Let $G = (V, E)$ be an arithmetic circuit with $v_1, v_2 \in V$. Without loss of generality we assume that v_1, v_2 are on level $\ell \in \mathbb{N}$ with odd ℓ. In the following, we construct in logarithmic space a cost chain $\mathcal{C}$ and a cost formula φ such that

$$val(v_1) \geq val(v_2) \iff \mathcal{P}(K_{\mathcal{C}} \models \varphi) \geq 1/2\,. \tag{4}$$

Using Prop. 6 we first construct two cost chains $\mathcal{C}_1 = (Q, q_1, t, \Delta)$ and $\mathcal{C}_2 = (Q, q_2, t, \Delta)$ and $T_1, T_2 \in \mathbb{N}$ such that $\mathcal{P}(K_{\mathcal{C}_i} = T_i) = val(v_i)/m$ holds for $i \in \{1, 2\}$ and for $m \in \mathbb{N}$ as given by Prop. 6. We compute a number $H \in \mathbb{N}$ with $H \geq T_2$ such that $\mathcal{P}(K_{\mathcal{C}_2} > H) < 1/m$. By Prop. 2, it suffices to take

$$H \geq \max\left\{T_2,\ k_{max} \cdot \left\lceil |Q| \cdot \left(\ln(m+1)/p_{min}^{|Q|} + 1\right)\right\rceil\right\}\ ,$$

where k_{max} and p_{min} refer to $\mathcal{C}_2$. Let $\varepsilon := \mathcal{P}(K_{\mathcal{C}_2} > H \ \wedge\ K_{\mathcal{C}_2} \neq H + 1 + T_1)$. We have

$$0 \ \leq\ \varepsilon \ \leq\ \mathcal{P}(K_{\mathcal{C}_2} > H) \ <\ 1/m\,.$$

Now we combine $\mathcal{C}_1$ and $\mathcal{C}_2$ to a cost chain $\mathcal{C} = (Q \uplus \{q_0\}, q_0, t, \widetilde{\Delta})$, where $\widetilde{\Delta}$ extends Δ by

$$\widetilde{\Delta}(q_0)(q_1, H+1) = 1/2 \quad \text{and} \quad \widetilde{\Delta}(q_0)(q_2, 0) = 1/2\,.$$

By this construction, the new cost chain $\mathcal{C}$ initially either incurs cost $H + 1$ and then emulates $\mathcal{C}_1$, or incurs cost 0 and then emulates $\mathcal{C}_2$. Those possibilities have probability $1/2$ each. We define the cost formula

$$\varphi := (x \leq T_2 - 1) \ \vee\ (T_2 + 1 \leq x \leq H) \ \vee\ (x = H + 1 + T_1)\,.$$

The construction of $\mathcal{C}$ and the definition of ε gives that $\mathcal{P}(K_{\mathcal{C}} \models \varphi)$ is equal to

$$\begin{aligned}&\frac{1}{2} \cdot \mathcal{P}(K_{\mathcal{C}_1} = T_1) + \frac{1}{2} \cdot \big(\mathcal{P}(K_{\mathcal{C}_2} \leq H \ \vee \ K_{\mathcal{C}_2} = H + 1 + T_1) - \mathcal{P}(K_{\mathcal{C}_2} = T_2)\big) \\ =&\frac{1}{2} \cdot val(v_1)/m + \frac{1}{2} \cdot \big(1 - \varepsilon - val(v_2)/m\big)\end{aligned}$$

It follows that we have $\mathcal{P}(K_{\mathcal{C}} \models \varphi) \geq 1/2$ if and only if $val(v_1)/m \geq val(v_2)/m + \varepsilon$. Since $0 \leq \varepsilon < 1/m$ and $val(v_1), val(v_2)$ are integer numbers, we have shown the equivalence (4). This completes the proof of the POSSLP lower bound. □

Let us make two remarks on the construction just given: First, the representation of m from Prop. 6 is of exponential size. However, the computation of H only requires an upper bound on the *logarithm* of $m + 1$. Therefore, the reduction can be performed in logarithmic space. Second, the structure of the cost formula φ, in particular the number of inequalities, is fixed. Only the constants T_1, T_2, H depend on the instance.

D Proofs of Section 5

Theorem 7. *The cost problem for acyclic cost processes is in* PSPACE*. It is* PSPACE*-hard, even for atomic cost formulas.*

Proof. In the main body of the paper we proved the upper bound and gave a sketch of the PSPACE-hardness construction. Following up on this, we now provide the details of that reduction.

Let $k_{max} := \max\{k_1, k_2, \ldots, k_n\}$. We choose $\ell := 1 + nk_{max}$. Before an action in control state q_{n-2} is played, at most the following cost is incurred:

$$\frac{n-2}{2} \cdot (\ell + 2k_{max}) \ = \ \frac{n}{2} \cdot \ell + nk_{max} - \ell - 2k_{max} \ < \ \frac{n}{2} \cdot \ell \ \leq \ \frac{n}{2} \cdot \ell + T \ = \ B \ ,$$

so one cannot reach the full budget B before an action in control state q_{n-2} is played. We choose

$$M := 2^{n/2} n^2 \ell^2 \quad \text{and} \quad \tau := \left(B - \frac{1}{2} \cdot \frac{1}{2^{n/2}}\right) \big/ M \ . \tag{5}$$

For the sake of the argument we slightly change the standard *old* MDP to a *new* MDP, but without affecting the scheduler's winning chances. The control state t is removed. Any old transition δ from q_i to t is redirected: with the probability of δ the new MDP transitions to q_{i+2} and incurs the cost of δ; in addition, one *marble* is gained if the accumulated cost including the one of δ is at most the budget B. The idea is that a win in the old MDP (i.e., a transition to t having kept within budget) should correspond exactly to gaining *at least one* marble in the new MDP. The new MDP will be easier to analyse.

We make the definition of the new MDP more precise: When in q_i, the new MDP transitions to q_{i+2} with probability 1. The cost incurred and marbles gained during that transition depend on the action taken and on probabilistic decisions as follows. Suppose action a_j (with $j \in \{0, 1\}$) is taken in q_i, and cost C_i has been accumulated up to q_i. Then:

1. $j' \in \{0,1\}$ is chosen with probability $1/2$ each.
2. Cost $\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}$ is incurred.
3. If $C_i + \ell + j \cdot k_{i+1} + j' \cdot k_{i+2} \leq B$, then, in addition, one marble is gained with probability $\frac{\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}}{M}$, and no marble is gained with probability $1 - \frac{\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}}{M}$.

In the new MDP, the scheduler's objective is, during the path from q_0 to q_n, to gain *at least one marble*. Since an optimal scheduler in the new MDP does not need to take into account whether or when marbles have been gained, we assume that schedulers in the new MDP do not take marbles into account. The new MDP is constructed so that the winning chances are the same in the old and the new MDP; in fact, any scheduler in the old MDP translates into a scheduler with the same winning chance in the new MDP, and vice versa.

Fix a scheduler σ in the new MDP. A vector $\mathbf{x} = (x_2, x_4, \ldots, x_n) \in \{0,1\}^{n/2}$ determines the cost incurred during a run, in the following way: when σ takes action a_j (for $j \in \{0,1\}$) in state q_i, then cost $c_i(\sigma, \mathbf{x}) := \ell + j \cdot k_{i+1} + x_{i+2} \cdot k_{i+2}$ is added upon transitioning to q_{i+2}. Conversely, a run determines the vector $\mathbf{x}$. Let $\hat{p}_i^\sigma(\mathbf{x})$ denote the conditional probability (conditioned under $\mathbf{x}$) that a marble is gained upon transitioning from q_i to q_{i+2}. We have:

$$\hat{p}_i^\sigma(\mathbf{x}) = \begin{cases} c_i(\sigma, \mathbf{x})/M & \text{if } c_0(\sigma, \mathbf{x}) + c_2(\sigma, \mathbf{x}) + \cdots + c_i(\sigma, \mathbf{x}) \leq B \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

It follows that we have:

$$\hat{p}_i^\sigma(\mathbf{x}) \;\leq\; \frac{2\ell}{M} \tag{7}$$

$$\sum_{\text{even } i=0}^{n-2} \hat{p}_i^\sigma(\mathbf{x}) \;\leq\; B/M \tag{8}$$

Denote by p_i^σ (for $i = 0, 2, \ldots, n-2$) the (total) probability that a marble is gained upon transitioning from q_i to q_{i+2}. By the law of total probability we have

$$p_i^\sigma \;=\; \sum_{\mathbf{x} \in \{0,1\}^{n/2}} \frac{1}{2^{n/2}} \cdot \hat{p}_i^\sigma(\mathbf{x}) \tag{9}$$

$$\text{and hence, by (7)}, \qquad p_i^\sigma \leq \frac{2\ell}{M} \tag{10}$$

We show that Player Odd has a winning strategy in the QSUBSETSUM game if and only if the probability of winning in the new MDP is at least τ.

– Assume that Player Odd has a winning strategy in the QSUBSETSUM game. Let σ be the scheduler in the new MDP that emulates Player Odd's winning strategy from the QSUBSETSUM game. Using σ the accumulated cost upon reaching q_n is exactly B, with probability 1. So for all $\mathbf{x} \in \{0,1\}^{n/2}$ we have:

$$c_0(\sigma, \mathbf{x}) + c_2(\sigma, \mathbf{x}) + \cdots + c_{n-2}(\sigma, \mathbf{x}) \;=\; B \tag{11}$$

Thus:

$$\begin{aligned}
\sum_{\text{even } i=0}^{n-2} p_i^\sigma &= \sum_{\mathbf{x}\in\{0,1\}^{n/2}} \frac{1}{2^{n/2}} \sum_{\text{even } i=0}^{n-2} \hat{p}_i^\sigma(\mathbf{x}) && \text{by (9)} \\
&= \sum_{\mathbf{x}\in\{0,1\}^{n/2}} \frac{1}{2^{n/2}} \sum_{\text{even } i=0}^{n-2} c_i(\sigma,\mathbf{x})/M && \text{by (6) and (11)} \\
&= B/M && \text{by (11)}
\end{aligned} \tag{12}$$

Further we have:

$$\begin{aligned}
\sum_{\substack{\text{even } i,j \\ i<j\leq n-2}} p_i^\sigma p_j^\sigma &\overset{\text{by (10)}}{\leq} \binom{n/2}{2} \left(\frac{2\ell}{M}\right)^2 = \frac{\frac{n}{2}\cdot\left(\frac{n}{2}-1\right)\cdot 4\ell^2}{2M^2} \\
&\leq \frac{n^2\ell^2}{2M^2} \overset{\text{by (5)}}{=} \left(\frac{1}{2}\cdot\frac{1}{2^{n/2}}\right)/M
\end{aligned} \tag{13}$$

Recall that the probability of winning equals the probability of gaining at least one marble. The latter probability is, by the inclusion-exclusion principle, bounded below as follows:

$$\sum_{\text{even } i=0}^{n-2} p_i^\sigma \;-\; \sum_{\substack{\text{even } i,j \\ i<j\leq n-2}} p_i^\sigma p_j^\sigma \overset{\text{by (12) and (13)}}{\geq} \left(B - \frac{1}{2}\cdot\frac{1}{2^{n/2}}\right)/M \overset{\text{by (5)}}{=} \tau$$

We conclude that the probability of winning is at least τ.

– Assume that Player Odd does not have a winning strategy in the QSUBSETSUM game. Consider any scheduler σ for the new MDP. Since the corresponding strategy in the QSUBSETSUM game is not winning, there exists $\mathbf{y} \in \{0,1\}^{n/2}$ with $c_0(\sigma,\mathbf{y}) + c_2(\sigma,\mathbf{y}) + \cdots + c_{n-2}(\sigma,\mathbf{y}) \neq B$. By (6) it follows:

$$\sum_{\text{even } i=0}^{n-2} \hat{p}_i^\sigma(\mathbf{y}) \;\leq\; (B-1)/M \tag{14}$$

By the union bound the probability of gaining at least one marble is bounded above as follows:

$$\begin{aligned}
\sum_{\text{even } i=0}^{n-2} p_i^\sigma &= \sum_{\mathbf{x}\in\{0,1\}^{n/2}} \frac{1}{2^{n/2}} \sum_{\text{even } i=0}^{n-2} \hat{p}_i^\sigma(\mathbf{x}) && \text{by (9)} \\
&\leq \left(1 - \frac{1}{2^{n/2}}\right)\cdot\frac{B}{M} + \frac{1}{2^{n/2}}\cdot\frac{B-1}{M} && \text{by (8) and (14)} \\
&= \left(B - \frac{1}{2^{n/2}}\right)/M \\
&< \tau && \text{by (5)}
\end{aligned}$$

We conclude that the probability of winning is less than τ.

This completes the log-space reduction. □

Theorem 8. *The cost problem is* EXP*-complete.*

Proof. We reduce from the problem of determining the winner in a countdown game [12]. A *countdown game* is a tuple $(S, \circ\!\!\rightarrow, s_0, T)$ where S is a finite set of states, $\circ\!\!\rightarrow \subseteq S \times \mathbb{N} \setminus \{0\} \times S$ is a transition relation, $s_0 \in S$ is the initial state, and T is the final value. We write $s \overset{k}{\circ\!\!\rightarrow} r$ if $(s, k, r) \in \circ\!\!\rightarrow$. A configuration of the game is an element $(s, c) \in S \times \mathbb{N}$. The game starts in configuration $(s_0, 0)$ and proceeds in moves: if the current configuration is $(s, c) \in S \times \mathbb{N}$, first Player 1 chooses a number k with $0 < k \leq T - c$ and $s \overset{k}{\circ\!\!\rightarrow} r$ for at least one $r \in S$; then Player 2 chooses a state $r \in S$ with $s \overset{k}{\circ\!\!\rightarrow} r$. The resulting new configuration is $(r, c + k)$. Player 1 wins if she hits a configuration from $S \times \{T\}$, and she loses if she cannot move (and has not yet won). (We have slightly paraphrased the game from [12] for technical convenience, rendering the term *countdown* game somewhat inept.)

The problem of determining the winner in a countdown game was shown EXP-complete in [12]. Let $(S, \circ\!\!\rightarrow, s_0, T)$ be a countdown game. We construct a cost process $\mathcal{C} = (Q, s_0, t, A, En, \Delta)$ so that Player 1 can win the countdown game if and only if there is a scheduler σ with $\mathcal{P}_\sigma(K = T) = 1$. The intuition is that Player 1 corresponds to the scheduler and Player 2 corresponds to randomness. We take

$$Q := S \cup \{q_i : i \in \mathbb{N},\ 2^i \leq T\} \cup \{t\}\,.$$

Intuitively, the states in S are used in a *first phase*, which directly reflects the countdown game. The states q_i are used in a *second phase*, which is acyclic and ends in the final control state t.

For all $s \in S$ we take

$$En(s) := \{a_{stop}\} \cup \{k \in \mathbb{N} \setminus \{0\} : \exists\, r \in S.\, s \overset{k}{\circ\!\!\rightarrow} r\}\,.$$

Whenever $s \overset{k}{\circ\!\!\rightarrow} r$, we set $\Delta(s, k)(r, k) > 0$. (We do not specify the exact values of positive probabilities, as they do not matter. For concreteness one could take a uniform distribution.) Those transitions directly reflect the countdown game. Whenever $s \overset{k}{\circ\!\!\rightarrow} r$, we also set $\Delta(s, k)(q_0, k) > 0$. Those transitions allow "randomness" to enter the second phase, which starts in q_0. Further, for all $s \in S$ we set $\Delta(s, a_{stop})(t, 0) = 1$. Those transitions allow the scheduler to jump directly to the final control state t, skipping the second phase.

Now we describe the transitions in the second phase. Let $i_{max} \in \mathbb{N}$ be the largest integer with $2^{i_{max}} \leq T$. For all $i \in \{0, 1, \ldots, i_{max}\}$ we take $En(q_i) = \{a_0, a_1\}$ and

$$\begin{aligned} \Delta(q_i, a_0)(q_{i+1}, 0) &= 1 \quad \text{and} \\ \Delta(q_i, a_1)(q_{i+1}, 2^i) &= 1\ , \end{aligned}$$

where $q_{i_{max}+1}$ is identified with t. The second phase allows the scheduler to incur an arbitrary cost between 0 and T (and possibly more). That phase is acyclic and leads to t.

Observe that t is reached with probability 1. We show that Player 1 can win the countdown game if and only if the scheduler in the cost process can achieve $K = T$ with probability 1.

Assume Player 1 can win the countdown game. Then the scheduler can emulate Player 1's winning strategy. If randomness enters the second phase while the cost c accumulated so far is at most T, then the scheduler incurs additional cost $T - c$ in the second phase and wins. If and when accumulated cost exactly T is reached in the first phase, the scheduler plays a_{stop}, so it jumps to t and wins. Since the scheduler emulates Player 1's winning strategy, it will not get in a state in which the accumulated cost is larger than T.

Conversely, assume Player 2 has a winning strategy in the countdown game. If the scheduler jumps to t while accumulated cost T has not yet been reached, the scheduler loses. If the scheduler does not do that, randomness emulates with non-zero probability Player 2's winning strategy. This leads to a state $(s, c) \in S \times \mathbb{N}$ with $c > T$, from which the scheduler loses with probability 1. This completes the log-space reduction. □

Corollary 9. *The cost-utility problem is* EXP*-complete.*

Proof. Membership in EXP follows from Prop. 1.

For hardness, observe that the proof of Thm. 8 reveals that the following problem is EXP-hard. The *qualitative cost problem* asks, given a cost process and $T \in \mathbb{N}$, whether there exists a scheduler σ with $\mathcal{P}_\sigma(K = T) = 1$. Reduce the qualitative cost problem to the cost-utility problem where both the cost and the utility in the new MDP are increased as the cost in the cost process. Then we have $\mathcal{P}_\sigma(K = T) = 1$ in the cost process if and only if in the new MDP the cost is at most T and the utility is at least T with probability 1. □

E Proofs of Section 5

Theorem 10. *The universal cost problem for acyclic cost processes is in* PSPACE*. It is* PSPACE*-hard, even for atomic cost formulas. The universal cost problem is* EXP*-complete.*

Proof. Considering the proof sketch in the main text, it remains to show that the universal cost problem for acyclic cost processes and atomic cost formulas is PSPACE-hard. By a straightforward logspace reduction as in the beginning of the proof sketch in the main text, it suffices to prove PSPACE-hardness of the following problem: given an acyclic cost process $\mathcal{C}$ and a number $B \in \mathbb{N}$ and a probability τ, does there exist a scheduler σ with $\mathcal{P}_\sigma(K_\mathcal{C} < B) < \tau$?

For that we adapt the reduction from Theorem 7. The differences to that reduction arise from the fact that the scheduler now tries to maximise the probability of achieving cost *at least* B. Given an instance $(k_1, \ldots, k_n, T)$, where n is even, of the QSUBSETSUM problem, we take, as before,

$$B := \frac{n}{2} \cdot \ell + T \,,$$

for an $\ell \in \mathbb{N}$ defined later. Further, we construct an acyclic cost process $\mathcal{C} = (Q, q_0, t, A, En, \Delta)$ similarly as before. In particular, we take again $Q = \{q_0, q_2, \ldots, q_{n-2}, q_n, t\}$. For a large number $M \in \mathbb{N}$, defined later, we set for all even $i \leq n-2$ and for $j \in \{0, 1\}$:

$$\Delta(q_i, a_j)(q_{i+2}, \ell + j \cdot k_{i+1}) = \frac{1}{2} \cdot \left(1 - \frac{\ell + j \cdot k_{i+1}}{M}\right)$$
$$\Delta(q_i, a_j)(q_{i+2}, \ell + j \cdot k_{i+1} + k_{i+2}) = \frac{1}{2} \cdot \left(1 - \frac{\ell + j \cdot k_{i+1} + k_{i+2}}{M}\right)$$
$$\Delta(q_i, a_j)(t, 0) = \frac{1}{2} \cdot \frac{\ell + j \cdot k_{i+1}}{M} + \frac{1}{2} \cdot \frac{\ell + j \cdot k_{i+1} + k_{i+2}}{M}$$

So with a high probability the MDP transitions from q_i to q_{i+2}, and cost ℓ, $\ell + k_{i+1}$, $\ell + k_{i+2}$, $\ell + k_{i+1} + k_{i+2}$ is incurred, depending on the scheduler's (i.e., Player Odd's) actions and on the random (Player Even) outcome. But with a small probability, which is proportional to the cost that would be otherwise incurred, the MDP takes a zero-cost transition to t, which is a "loss" for the scheduler, because, as in the old reduction, ℓ is chosen big enough so that the total cost is strictly smaller than B before an action in control state q_{n-2} has been played. There is a single zero-cost transition from q_n to t:

$$\Delta(q_n, a)(t, 0) = 1 \qquad \text{with } En(q_n) = \{a\}$$

The MDP is designed so that the scheduler probably "wins" (i.e., reaches cost at least B), if Player Odd can always reach cost at least T; but whenever cost k is incurred, there is a small probability k/M of losing. Since $1/M$ is small, the overall probability of losing is approximately C/M if total cost $C \geq B$ is incurred. In order to minimise this probability, the scheduler wants to minimise the total cost while still incurring cost at least B, so the optimal scheduler will target B as total cost.

Similarly to the old reduction, the values for ℓ, M and τ need to be chosen carefully, as the overall probability of losing is not exactly the sum of the individual losing probabilities. Rather, this sum is – by the "union bound" – only an upper bound. One needs to show that the sum approximates the real probability closely enough.

Now we give the details. Let $k_{max} := \max\{k_1, k_2, \ldots, k_n\}$. We choose $\ell := 1 + nk_{max}$ as in the old reduction, so one cannot reach the full budget B before an action in control state q_{n-2} is played. Without loss of generality we can assume that $T \leq nk_{max}$, as otherwise the instance of the QSUBSETSUM problem would be trivial. Hence we have:

$$B + 1 \;=\; \frac{n}{2}\ell + T + 1 \;\leq\; \frac{n}{2}\ell + \ell \;\leq\; n\ell \;\leq\; n^2\ell^2 \tag{15}$$

We choose

$$M := 2^{n/2} n^2 \ell^2 \quad \text{and} \quad \tau := \left(B + \frac{1}{2} \cdot \frac{1}{2^{n/2}}\right) / M \,. \tag{16}$$

For the sake of the argument we slightly change the standard *old* MDP to a *new* MDP, but without affecting the scheduler's winning chances. The new MDP will be easier to analyse. The control state t is removed. When in q_i, the new MDP transitions to q_{i+2}

with probability 1. The cost incurred and marbles gained during that transition depend on the action taken and on probabilistic decisions as follows. Suppose action a_j (with $j \in \{0,1\}$) is taken in q_i, and cost C_i has been accumulated up to q_i. Then:

1. $j' \in \{0,1\}$ is chosen with probability $1/2$ each.
2. Cost $\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}$ is incurred.
3. One marble is gained with probability $\frac{\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}}{M}$, and no marble is gained with probability $1 - \frac{\ell + j \cdot k_{i+1} + j' \cdot k_{i+2}}{M}$.

In the new MDP, the scheduler's objective is, during the path from q_0 to q_n, *to gain no marble and to accumulate cost at least* B. Since an optimal scheduler in the new MDP does not need to take into account whether or when marbles have been gained, we assume that schedulers in the new MDP do not take marbles into account. The new MDP is constructed so that the winning chances are the same in the old and the new MDP; in fact, any scheduler in the old MDP translates into a scheduler with the same winning chance in the new MDP, and vice versa.

Fix a scheduler σ in the new MDP. A vector $\mathbf{x} = (x_2, x_4, \ldots, x_n) \in \{0,1\}^{n/2}$ determines the cost incurred during a run, in the following way: when σ takes action a_j (for $j \in \{0,1\}$) in state q_i, then cost $c_i(\sigma, \mathbf{x}) := \ell + j \cdot k_{i+1} + x_{i+2} \cdot k_{i+2}$ is added upon transitioning to q_{i+2}. Conversely, a run determines the vector $\mathbf{x}$. Let $\hat{p}_i^\sigma(\mathbf{x})$ denote the conditional probability (conditioned under $\mathbf{x}$) that a marble is gained upon transitioning from q_i to q_{i+2}. We have:

$$\hat{p}_i^\sigma(\mathbf{x}) \;=\; c_i(\sigma, \mathbf{x})/M \tag{17}$$

$$\leq\; 2\ell/M \tag{18}$$

Denote by p_i^σ (for $i = 0, 2, \ldots, n-2$) the (total) probability that a marble is gained upon transitioning from q_i to q_{i+2}. By the law of total probability we have

$$p_i^\sigma = \sum_{\mathbf{x} \in \{0,1\}^{n/2}} \frac{1}{2^{n/2}} \cdot \hat{p}_i^\sigma(\mathbf{x}) \;.$$

It follows:

$$p_i^\sigma = \sum_{\mathbf{x} \in \{0,1\}^{n/2}} \frac{1}{2^{n/2}} \cdot c_i(\sigma, \mathbf{x})/M \qquad \text{by (17)} \tag{19}$$

$$p_i^\sigma \leq \frac{2\ell}{M} \qquad \text{by (18)} \tag{20}$$

We show that Player Odd has a winning strategy in the QSUBSETSUM game if and only if the probability of losing in the new MDP is less than τ.

– Assume that Player Odd has a winning strategy in the QSUBSETSUM game. Let σ be the scheduler in the new MDP that emulates Player Odd's winning strategy from the QSUBSETSUM game. Using σ the accumulated cost upon reaching q_n is exactly B, with probability 1. So for all $\mathbf{x} \in \{0,1\}^{n/2}$ we have:

$$c_0(\sigma, \mathbf{x}) + c_2(\sigma, \mathbf{x}) + \cdots + c_{n-2}(\sigma, \mathbf{x}) \;=\; B \tag{21}$$

Since the scheduler accumulates, with probability 1, cost exactly B, the probability of losing equals the probability of gaining at least one marble. By the union bound this probability is bounded above as follows:

$$\begin{aligned}\sum_{\text{even } i=0}^{n-2} p_i^\sigma &= \sum_{\mathbf{x}\in\{0,1\}^{n/2}} \frac{1}{2^{n/2}} \sum_{\text{even } i=0}^{n-2} c_i(\sigma,\mathbf{x})/M && \text{by (19)}\\ &= B/M && \text{by (21)}\\ &< \tau && \text{by (16)}\end{aligned}$$

We conclude that the probability of losing is less than τ.

– Assume that Player Odd does not have a winning strategy in the QSUBSETSUM game. Consider any scheduler σ for the new MDP. Suppose that there exists $\mathbf{y} \in \{0,1\}^{n/2}$ with $c_0(\sigma,\mathbf{y})+c_2(\sigma,\mathbf{y})+\cdots+c_{n-2}(\sigma,\mathbf{y}) < B$. Recall that the scheduler loses if it accumulates cost less than B. So the probability of losing is at least

$$\frac{1}{2^{n/2}} = \frac{n^2\ell^2}{2^{n/2}n^2\ell^2} \overset{\text{by (15)}}{\geq} \frac{B+1}{M} \overset{\text{by (16)}}{\geq} \tau\,.$$

So we can assume for the rest of the proof that for all $\mathbf{x} \in \{0,1\}^{n/2}$ we have

$$c_0(\sigma,\mathbf{x}) + c_2(\sigma,\mathbf{x}) + \cdots + c_{n-2}(\sigma,\mathbf{x}) \;\geq\; B\,. \tag{22}$$

Since the strategy corresponding to σ in the QSUBSETSUM game is not winning, there exists $\mathbf{y} \in \{0,1\}^{n/2}$ with

$$c_0(\sigma,\mathbf{y}) + c_2(\sigma,\mathbf{y}) + \cdots + c_{n-2}(\sigma,\mathbf{y}) \;\geq\; B+1\,. \tag{23}$$

We have:

$$\begin{aligned}&\sum_{\text{even } i=0}^{n-2} p_i^\sigma\\ &= \sum_{\mathbf{x}\in\{0,1\}^{n/2}} \frac{1}{2^{n/2}} \sum_{\text{even } i=0}^{n-2} c_i(\sigma,\mathbf{x})/M && \text{by (19)}\\ &\geq \left(1-\frac{1}{2^{n/2}}\right)\cdot\frac{B}{M} + \frac{1}{2^{n/2}}\cdot\frac{B+1}{M} && \text{by (22) and (23)}\\ &= \left(B+\frac{1}{2^{n/2}}\right)/M\end{aligned} \tag{24}$$

Further we have:

$$\begin{aligned}\sum_{\substack{\text{even } i,j\\ i<j\leq n-2}} p_i^\sigma p_j^\sigma &\overset{\text{by (20)}}{\leq} \binom{n/2}{2}\left(\frac{2\ell}{M}\right)^2 = \frac{\frac{n}{2}\cdot\left(\frac{n}{2}-1\right)\cdot 4\ell^2}{2M^2}\\ &\leq \frac{n^2\ell^2}{2M^2} \overset{\text{by (16)}}{=} \left(\frac{1}{2}\cdot\frac{1}{2^{n/2}}\right)/M\end{aligned} \tag{25}$$

Recall that the scheduler loses if it gains at least one marble. So the probability of losing is, by the inclusion-exclusion principle, bounded below as follows:

$$\sum_{\text{even } i=0}^{n-2} p_i^\sigma \;-\; \sum_{\substack{\text{even } i,j \\ i<j\leq n-2}} p_i^\sigma p_j^\sigma \overset{\text{by (24) and (25)}}{\geq} \left(B + \frac{1}{2}\cdot\frac{1}{2^{n/2}}\right)/M \overset{\text{by (16)}}{=} \tau$$

We conclude that the probability of losing is at least τ.

This completes the log-space reduction. □