Stochastization of Weighted Automata*

Guy Avni Orna Kupferman

School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel.

Abstract

Nondeterministic weighted finite automata (WFAs) map input words to real numbers. Each transition of a WFA is labeled by both a letter from some alphabet and a weight. The weight of a run is the sum of the weights on the transitions it traverses, and the weight of a word is the minimal weight of a run on it. In *probabilistic weighted automata* (PWFAs), the transitions are further labeled by probabilities, and the weight of a word is the expected weight of a run on it. We define and study *stochastization* of WFAs: given a WFA $\mathcal{A}$, stochastization turns it into a PWFA $\mathcal{A}'$ by labeling its transitions by probabilities. The weight of a word in $\mathcal{A}'$ can only increase with respect to its weight in $\mathcal{A}$, and we seek stochastizations in which $\mathcal{A}'$ α-approximates $\mathcal{A}$ for the minimal possible factor $\alpha \geq 1$. That is, the weight of every word in $\mathcal{A}'$ is at most α times its weight in $\mathcal{A}$. We show that stochastization is useful in reasoning about the competitive ratio of randomized online algorithms and in approximated determinization of WFAs. We study the problem of deciding, given a WFA $\mathcal{A}$ and a factor $\alpha \geq 1$, whether there is a stochastization of $\mathcal{A}$ that achieves an α-approximation. We show that the problem is in general undecidable, yet can be solved in PSPACE for a useful class of WFAs.

1 Introduction

A recent development in formal methods for reasoning about reactive systems is an extension of the Boolean setting to a multi-valued one. The multi-valued component may originate from the system, for example when propositions are weighted or when transitions involve costs and rewards [16], and may also originate from rich specification formalisms applied to Boolean systems, for example when asking quantitative questions about the system [7] or when specifying its quality [1]. The interest in multi-valued reasoning has led to growing interest in *nondeterministic weighted finite*

*The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 278410, and from The Israel Science Foundation (grant no 1229/10).

automata (WFAs), which map an input word to a value from a semi-ring over a large domain [12, 23].

Many applications of WFAs use the tropical semi-ring $\langle \mathbb{R}^+ \cup \{\infty\}, \min, +, \infty, 0\rangle$. There, each transition has a *weight*, the weight of a run is the sum of the weights of the transitions taken along the run, and the weight of a word is the minimal weight of a run on it. Beyond the applications of WFAs over the tropical semi-ring in quantitative reasoning about systems, they are used also in text, speech, and image processing, where the costs of the WFA are used in order to account for the variability of the data and to rank alternative hypotheses [11, 24].

A different kind of applications of WFAs uses the semi-ring $\langle \mathbb{R}^+ \cup \{\infty\}, +, \times, 0, 1\rangle$. There, the weight of a run is the product of the weights of the transitions taken along it, and the weight of a word is the sum of the weights of the runs on it. In particular, when the weights on the transitions are in $[0, 1]$ and form a probabilistic transition function (that is, for every state q and letter σ, the sum of the weights of the σ-transitions from q is 1), we obtain a *probabilistic finite automaton* (PFA, for short). In fact, the probabilistic setting goes back to the 60's [25].

The theoretical properties of WFAs are less clean and more challenging than these of their Boolean counterparts. For example, not all WFAs can be determinized [23], and the problem of deciding whether a given WFA has an equivalent deterministic WFA is open. As another example, the containment problem is undecidable for WFAs [21]. The multi-valued setting also leads to new questions about automata and their languages, like approximated determinization [3] or discounting models [13].

By combining the tropical and the probability semi-rings, we obtain a *probabilistic weighted finite automaton* (PWFA, for short). There, each transition has two weights, which we refer to as the *cost* and the *probability*. The weight that the PWFA assigns to a word is then the expected cost of the runs on it. That is, as in the tropical semi-ring, the cost of each run is the sum the costs of the transitions along the run, and as in probabilistic automata, the contribution of each run to the weight of a word depends on both its cost and probability. While PFAs have been extensively studied (e.g., [6]), we are only aware of [20] in which PWFAs were considered.

We introduce and study *stochastization* of WFAs. Given a WFA $\mathcal{A}$, stochastization turns it into a PWFA $\mathcal{A}'$ by labeling its transitions with probabilities. Recall that in a WFA, the weight of a word is the minimal weight of a run on it. Stochastization of a WFA $\mathcal{A}$ results in a PWFA $\mathcal{A}'$ with the same set of runs, and the weight of a word is the expected cost of these runs. Accordingly, the weight of a word in $\mathcal{A}'$ can only increase with respect to its weight in $\mathcal{A}$. Hence, we seek stochastizations in which $\mathcal{A}'$ *α-approximates* $\mathcal{A}$ for the minimal possible factor $\alpha \geq 1$. That is, the weight of every word in $\mathcal{A}'$ is at most α times its weight in $\mathcal{A}$. We note that stochastization has been studied in the Boolean setting in [14], where a PFA is constructed from an NFA. [1] Before describing our contribution, we motivate stochastization further.

[1] Beyond considering the Boolean setting, the work in [14] concerns the ability to instantiate probabilities so that at least one word is accepted with probability arbitrarily close to 1. Thus, the type of questions and motivations are very different from these we study here in the weighted setting.

In [2], the authors describe a framework for using WFAs over the tropical semi-ring in order to reason about *online algorithms*. An online algorithm can be viewed as a reactive system: at each round, the environment issues a request, and the algorithm should process it. The sequence of requests is not known in advance, and the goal of the algorithm is to minimize the overall cost of processing the sequence. Online algorithms for many problems have been extensively studied [8]. The most interesting question about an online algorithm refers to its *competitive ratio*: the worst-case (with respect to all input sequences) ratio between the cost of the algorithm and the cost of an optimal solution – one that may be given by an *offline* algorithm, which knows the input sequence in advance. An online algorithm that achieves a competitive ratio α is said to be α-*competitive*.

Consider an optimization problem P with requests in Σ. The set of online algorithms for P that use memory S, for some finite set S, induces a WFA $\mathcal{A}_P$, with alphabet Σ and state space S, such that the transitions of $\mathcal{A}_P$ correspond to actions of the algorithms and the cost of each transition is the cost of the corresponding action. It is shown in [2] that many optimization problems have algorithms that use finite memory. Each run of $\mathcal{A}_P$ on a sequence $w \in \Sigma^*$ of requests corresponds to a way of serving the requests in w by an algorithm with memory S. Thus, the weight of w in $\mathcal{A}_P$ is the cost of an optimal offline algorithm on w that uses memory S. On the other hand, an online algorithm has to process each request as soon as it arrives and corresponds to a deterministic automaton embodied in $\mathcal{A}_P$. Accordingly, there exists an α-competitive online algorithm for P, for $\alpha \geq 1$, iff $\mathcal{A}_P$ embodies a deterministic automaton $\mathcal{A}'_P$ that α-approximates $\mathcal{A}_P$. The framework in [2] enables formal reasoning about the competitive ratio of online algorithms. The framework has been broaden to online algorithms with an extended memory or a bounded lookahead, and to a competitive analysis that takes into an account assumptions about the environment [3]. An additional useful broadening of the framework would be to consider *randomized* online algorithms, namely ones that may toss coins in order to choose their actions. Indeed, it is well known that many online algorithms that use randomized strategies achieve a better competitive ratio [8]. Technically, this means that rather than pruning the WFA $\mathcal{A}_P$ to a deterministic one, we consider its stochastization.

Recall that not all WFAs have equivalent or even α-approximating deterministic WFAs. Stochastization is thus useful in finding an approximate solution to problems that are intractable in the nondeterministic setting and are tractable in the probabilistic one. We describe two such applications. One is *reasoning about quantitative properties of probabilistic systems*. In the Boolean setting, while one cannot model check probabilistic systems, typically given by a Markov chain or a Markov decision process, with respect to a specification given by means of a nondeterministic automaton, it is possible to take the product of a probabilistic system with a deterministic or a probabilistic automaton, making model checking easy for them [26]. In the weighted setting, a quantitative specification may be given by a weighted automaton. Here too the product can be defined only with a deterministic or a probabilistic automaton. By stochastizating a WFA specification, we obtain a PWFA (a.k.a. a *rewarded Markov chain* in this context [17]) and can perform approximated model checking. A second application is *approximated determinization*. Existing algorithms for α-determinization [23, 3] handle families of WFAs in which different cycles that

can be traversed by runs on the same word cannot have weights that differ in more than an α multiplicative factor (a.k.a. "the α-twins property"). Indeed, cycles as above induce problematic cycles for subset-construction-type determinization constructions. As we show, stochastization can average such cycles, leading to an approximated-determinization construction that successfully α-determinizes WFAs that do not satisfy the α-twin property and thus could not be α-determinized using existing constructions.Let us note that another candidate application is *weighted language equivalence*, which is undecidable for WFAs but decidable for PWFA [20]. Unfortunately, however, weighted equivalence becomes pointless once approximation enters the picture.

Given a WFA $\mathcal{A}$ and a factor $\alpha \geq 1$, the *approximated stochastization problem* (AS problem, for short) is to decide whether there is a stochastization of $\mathcal{A}$ that α-approximates it. We study the AS problem and show that it is in general undecidable. Special tractable cases include two types of restrictions. First, restrictions on α: we show that when $\alpha = 1$, the problem coincides with determinization by pruning of WFAs, which can be solved in polynomial time. Then, restrictions on the structure of the WFA: we define the class of constant-ambiguous WFAs, namely WFAs whose degree of nondeterminism is a constant, and show that the AS problem for them is in PSPACE. On the other hand, the AS problem is NP-hard already for 7-ambiguous WFAs, namely WFAs that have at most 7 runs on each word. Even more restricted are tree-like WFAs, for which the problem can be solved in polynomial time, and so is the problem of finding a minimal approximation factor α. We show that these restricted classes are still expressive enough to model interesting optimization problems.

2 Preliminaries

A *nondeterministic finite weighted automaton* on finite words (WFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$, where Σ is an alphabet, Q is a finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a total transition relation (i.e., for every $q \in Q$ and $\sigma \in \Sigma$, there is at least one state $q' \in Q$ with $\langle q, \sigma, q' \rangle \in \Delta$), $q_0 \in Q$ is an initial state, and $\tau : \Delta \to \mathbb{R}^+$ is a weight function that maps each transition to a non-negative real value, which is the cost of traversing this transition. If for every $q \in Q$ and $\sigma \in \Sigma$ there is exactly one $q' \in Q$ such that $\langle q, \sigma, q' \rangle \in \Delta$, then $\mathcal{A}$ is a *deterministic WFA* (DWFA, for short). We assume that all states are reachable from the initial state. Consider a transition $t = \langle q, \sigma, q' \rangle \in \Delta$. We use *source*$(t)$, *label*$(t)$, and *target*$(t)$, to refer to q, σ, and q', respectively. It is sometimes convenient to use a transition function rather than a transition relation. Thus, we use $\delta_{\mathcal{A}} : Q \times \Sigma \to 2^Q$, where for $q \in Q$ and $\sigma \in \Sigma$, we define $\delta_{\mathcal{A}}(q, \sigma) = \{p \in Q : \langle q, \sigma, p \rangle \in \Delta\}$. When $\mathcal{A}$ is clear from the context we do not state it implicitly.

A *run* of $\mathcal{A}$ on a word $w = w_1 \dots w_n \in \Sigma^*$ is a sequence of transitions $r = r_1, \dots, r_n$ such that *source*$(r_1) \in Q_0$, for $1 \leq i < n$ we have *target*(r_i) = *source*(r_{i+1}), and for $1 \leq i \leq n$ we have *label*$(r_i) = w_i$. For a word $w \in \Sigma^*$, we denote by *runs*$(\mathcal{A}, w)$ the set of all runs of $\mathcal{A}$ on w. Note that since Δ is total, there is a run of $\mathcal{A}$ on every word in Σ^*, thus $|\textit{runs}(\mathcal{A}, w)| \geq 1$, for all

$w \in \Sigma^*$.[2] The value of the run, denoted $val(r)$, is the sum of costs of transitions it traverses. That is, $val(r) = \sum_{1 \leq i \leq n} \tau(r_i)$. We denote by *first*$(r)$ and *last*(r) the states in which r starts and ends, respectively, thus $start(r) =$ *source*(r_1) and *last*$(r) =$ *target*(r_n). Since $\mathcal{A}$ is nondeterministic, there can be more than one run on each word. We define the value that $\mathcal{A}$ assigns to the word w, denoted $val(\mathcal{A}, w)$, as the value of the minimal-valued run of $\mathcal{A}$ on w. That is, for every $w \in \Sigma^*$, we define $val(\mathcal{A}, w) = \min\{val(r) : r \in \textit{runs}(\mathcal{A}, w)\}$.

A *probabilistic finite weighted automaton* on finite words (PWFA, for short) is $\mathcal{P} = \langle \Sigma, Q, D, q_0, \tau \rangle$, where Σ, Q, q_0, and τ are as in WFAs, and $D : Q \times \Sigma \times Q \to [0, 1]$ is a *probabilistic transition function*. That is, it assigns for each two states $q, p \in Q$ and letter $\sigma \in \Sigma$ the probability of moving from q to p with letter σ. Accordingly, we have $\sum_{p \in Q} D(q, \sigma, p) = 1$, for every $q \in Q$ and $\sigma \in \Sigma$. We sometimes refer to a transition relation $\Delta_D \subseteq Q \times \Sigma \times Q$ induced by D. For two states $q, p \in Q$ and letter $\sigma \in \Sigma$, we have $\Delta_D(q, \sigma, p)$ iff $D(q, \sigma, p) > 0$. Then, $\tau : \Delta_D \to \mathbb{R}^+$ assigns positive weights to transitions with a positive probability. As in WFAs, we assume that all states are accessible from the initial state by path with a positive probability. Note that if for every $q \in Q$ and $\sigma \in \Sigma$, there is a state $p \in Q$ with $D(q, \sigma, p) = 1$, then $\mathcal{P}$ is a DWFA.

A run $r = r_1, \ldots, r_n$ of $\mathcal{P}$ on $w = w_1 \ldots w_n \in \Sigma^*$ is a sequence of transitions defined as in WFAs. The probability of r, denoted $\Pr[r]$, is $\prod_{1 \leq i \leq n} D(r_i)$. Similarly to WFAs, for $w \in \Sigma^*$, we denote by *runs*$(\mathcal{P}, w)$ the set of all runs of $\mathcal{P}$ on w with positive probability. We define $val(\mathcal{P}, w)$ to be the expected value of a run of $\mathcal{P}$ on w, thus $val(\mathcal{P}, w) = \sum_{r \in \textit{runs}(\mathcal{P}, w)} \Pr[r] \cdot val(r)$.

We say that a WFA $\mathcal{A}$ is k-ambiguous, for $k \in \mathbb{N}$, if k is the minimal number such that for every word $w \in \Sigma^*$, we have $|runs(\mathcal{A}, w)| \leq k$. We say that a WFA $\mathcal{A}$ is *constant-ambiguous* (a CA-WFA, for short) if $\mathcal{A}$ is k-ambiguous from some $k \in \mathbb{N}$. The definitions for PWFAs are similar, thus CA-PWFAs have a bound on the number of possible runs with positive probability.

A *stochastization* of a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$ is a construction of a PWFA that is obtained from $\mathcal{A}$ by assigning probabilities to its nondeterministic choices. Formally, it is a PWFA $\mathcal{A}^D = \langle \Sigma, Q, D, q_0, \tau \rangle$ obtained from $\mathcal{A}$ such that D is consistent with Δ. Thus, $\Delta_D = \Delta$. Note that since the transition function of $\mathcal{A}$ is total, there is always a stochastization of $\mathcal{A}$. Note also that if $\delta(q, \sigma)$ is a singleton $\{p\}$, then $D(q, \sigma, p) = 1$.

Recall that in a nondeterministic WFA, the value of a word is the minimal value of a run on it. Stochastization of a WFA $\mathcal{A}$ results in a PWFA $\mathcal{A}^D$ with the same set of runs, and the value of a word is some average of the values of these runs. Accordingly, the value of a word in $\mathcal{A}^D$ can only increase with respect to its value in $\mathcal{A}$. We would like to find a stochastization with which $\mathcal{A}^D$ *approximates* $\mathcal{A}$

Consider two weighted automata $\mathcal{A}$ and $\mathcal{B}$, and a factor $\alpha \in \mathbb{R}$ such that $\alpha \geq 1$. We say that $\mathcal{B}$ α-approximates $\mathcal{A}$ if, for every word $w \in \Sigma^*$, we have $\frac{1}{\alpha} \cdot val(\mathcal{A}, w) \leq val(\mathcal{B}, w) \leq \alpha \cdot val(\mathcal{A}, w)$.

[2] A different way to define WFAs would be to designate a set of accepting states. Then, the language of a WFA is the set of words that have an accepting run, and it assigns values to words in its language. Since it is possible to model acceptance by weights, our definition simplifies the setting and all states can be thought of as accepting.

We denote the latter also by $\frac{1}{\alpha}\mathcal{A} \leq \mathcal{B} \leq \alpha\mathcal{A}$. When $\alpha = 1$, we say that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*. Note that $\mathcal{A}$ and $\mathcal{B}$ are not necessarily the same type of automata.

A decision problem and an optimization problem naturally arise from this definition:

- *Approximation stochastization* (AS, for short): Given a WFA $\mathcal{A}$ and a factor $\alpha \geq 1$, decide whether there is a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$, in which case we say that $\mathcal{A}^D$ is an α-stochastization of $\mathcal{A}$.
- *Optimal approximation stochastization* (OAS, for short): Given a WFA $\mathcal{A}$, find the minimal $\alpha \geq 1$ such that there an α-stochastization of $\mathcal{A}$.

Recall that for every distribution function D, we have that $\mathcal{A} \leq \mathcal{A}^D$. Thus, in both the AS and OAS problems it is sufficient to require $\mathcal{A}^D \leq \alpha \cdot \mathcal{A}$.

Remark 2.1 [Tightening the approximation by a square-root factor] For a WFA $\mathcal{A}$ and $\beta \in [0, 1]$, let $\mathcal{A}_\beta$ be $\mathcal{A}$ with costs multiplied by β. It is easy to see that for all WFAs $\mathcal{A}$ and $\mathcal{B}$, we have $\frac{1}{\sqrt{\alpha}} \cdot \mathcal{A} \leq \mathcal{B}_{1/\sqrt{\alpha}} \leq \sqrt{\alpha} \cdot \mathcal{A}$ iff $\mathcal{A} \leq \mathcal{B} \leq \alpha \cdot \mathcal{A}$. In particular, taking $\mathcal{B}$ to be $\mathcal{A}^D$ for some distribution function D for $\mathcal{A}$, we have that $\mathcal{A}^D$ α-approximates $\mathcal{A}$ iff $\mathcal{A}^D_{1/\sqrt{\alpha}}$ $\sqrt{\alpha}$-approximates $\mathcal{A}$. It follows that when altering of weights is possible, we can tighten the approximation by a square-root factor. ∎

3 Motivation

In Section 1, we discussed the application of stochastization in reasoning about quantitative properties of probabilistic systems, reasoning about randomized online algorithms, and approximated determinization. Below we elaborate on the last two.

3.1 A WFA-Based Approach to Reasoning about Online Algorithms

In this section we describe [2]'s WFA-based approach to reasoning about online algorithms and extend it to account for randomized ones. An online algorithm with requests in Σ and actions in A corresponds to a function $g : \Sigma^+ \to A$ that maps sequences of requests (the history of the interaction so far) to an action to be taken. In general, the algorithm induces an infinite state space, as it may be in different states after processing different input sequences in Σ^*. For a finite set S of configurations, we say that g *uses memory* S, if there is a regular mapping of Σ^* into S such that g behaves in the same manner on identical continuations of words that are mapped to the same configuration.

We model the set of online algorithms that use memory S and solve an optimization problem P with requests in Σ and actions in A, by a WFA $\mathcal{A}_P = \langle \Sigma, S, \Delta, s_0, \tau \rangle$, such that Δ and τ describe

transitions between configurations and their costs, and s_0 is an initial configuration. Formally, $\Delta(s, \sigma, s')$ if the set $A' \subseteq A$ of actions that process the request σ from configuration s by updating the configuration to s' is non-empty, in which case $\tau(\langle s, \sigma, s' \rangle)$ is the minimal cost of an action in A'.

An offline algorithm knows the sequence of requests in advance and thus can resolve nondeterminism to obtain a minimal cost. Accordingly, the cost that an offline algorithm with state space S assigns to a sequence of requests $w \in \Sigma^*$ is exactly $val(\mathcal{A}_P, w)$. On the other hand, an online algorithm is a DWFA $\mathcal{A}'_P$ obtained from $\mathcal{A}_P$ by pruning nondeterministic choices. The competitive ratio of the online algorithm, namely the ratio between its performance and that of the offline algorithm, on the sequence of requests that maximizes this ratio, is then the factor α such that $\mathcal{A}'_P$ α-approximates $\mathcal{A}_P$. A randomized online algorithm for P that uses state space S can be viewed as a function from S to a probability distribution on A, which induces a probabilistic transition function on top of $\mathcal{A}_P$. Consequently, we have the following:

Theorem 3.1 *Consider an online problem P and a set S of configurations. Let $\mathcal{A}_P$ be a WFA with state space S that models online algorithms for P that use memory S. For all $\alpha \geq 1$, there is a randomized online algorithm for P using memory S that achieves competitive ratio α iff $\mathcal{A}_P$ has an α-stochastization.*

Example 3.2 The Ski-rental problem. Assume that renting skis costs \$1 per day and buying skis has a one-time cost of $\$M$. The online ski-rental problem copes with the fact it is not known in advance how many skiing days are left. Given an input request "skiing continues today", the online algorithm should decide whether to buy or rent skis. Typically, it is also assumed that renting skis is only allowed for at most $m \geq M$ consecutive days.

The WFA $\mathcal{A}$ induced by the ski-rental problem with parameters M and m is depicted in Fig. 1. Formally, $\mathcal{A} = \langle \{a\}, \{1, \ldots, m, q_{own}\}, \Delta, 1, \tau \rangle$, where Δ and τ are described below. A state $1 \leq i < m$ has two outgoing transitions: $\langle i, a, i+1 \rangle$ with weight 1, corresponds to renting skis at day i, and $\langle i, a, q_{own} \rangle$ with weight M, corresponding to buying skis at day i. Finally, there are transitions $\langle m, a, q_{own} \rangle$ with weight M and $\langle q_{own}, a, q_{own} \rangle$ with weight 0. The optimal deterministic online algorithm is due to [19]; rent skis for $M-1$ consecutive days, and buy skis on the M-th day, assuming skiing continues. It corresponds to the DWFA obtained by pruning all transitions but $\langle i, a, i+1 \rangle$, for $1 \leq i < M$, and $\langle M, a, q_{own} \rangle$. This DWFA achieves an optimal approximation factor of $2 - \frac{1}{M}$.

We describe a simple probabilistic algorithm that corresponds to a stochastization of $\mathcal{A}$ that achieves a better bound of $2 - \frac{1.5}{M}$. Intuitively, before skiing starts, toss a coin. If it turns out "heads", buy skis on the $(M-1)$-th day, and if it turns out "tails", buy on the M-th day. The corresponding distribution function D is depicted in red in Fig. 1 It is not hard to see that the worst case of this stochastization is attained by the word a^M for which we have $val(\mathcal{A}, a^M) = M$ and $val(\mathcal{A}^D, a^M) = \frac{1}{2} \cdot (M - 2 + M) + \frac{1}{2} \cdot (M - 1 + M) = 2M - 1.5$, thus $val(\mathcal{A}^D, a^M) \leq$

$(2-\frac{1.5}{M})\cdot val(\mathcal{A},a^M)$. Finding the optimal distribution function takes care [18, 9] and can achieve an approximation of $1+\frac{1}{(1+1/M)^M-1}\approx e/(e-1)\approx 1.582$ for $M\gg 1$. ∎

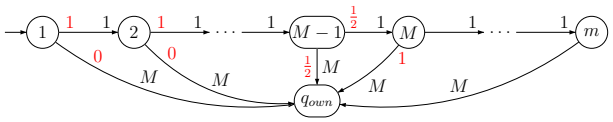

Figure 1: The WFA that is induced by the ski-rental problem with parameters M and m.

Next, we show that it is sometimes useful to apply the stochastization after extending the state space of $\mathcal{A}_P$. We illustrate this phenomena on the *Paging problem*. Before presenting the problem, we formalize the notion of memory.

Consider an online problem P and assume the corresponding WFA is $\mathcal{A}=\langle\Sigma,Q,\Delta,q_0,\tau\rangle$. A *memory set* is a pair $\mathcal{M}=\langle M,m_0\rangle$, where M is a finite set of memory states and $m_0\in M$ is an initial memory state. We augment $\mathcal{A}$ with $\mathcal{M}$ to construct a WFA $\mathcal{A}\times\mathcal{M}=\langle\Sigma,Q\times M,\Delta',\langle q_0,m_0\rangle,\tau'\rangle$, where $t'=\langle\langle q,m\rangle,\sigma,\langle q',m'\rangle\rangle\in\Delta'$ iff $t=\langle q,\sigma,q'\rangle\in\Delta$, in which case $\tau(t)=\tau'(t')$. Note that for every word $w\in\Sigma^*$, we have $val\big((\mathcal{A}\times\mathcal{M}),w\big)=val(\mathcal{A},w)$. Using memory is potentially helpful as every stochastization of $\mathcal{A}$ has a matching stochastization of $\mathcal{A}\times\mathcal{M}$ that achieves the same approximation factor, but not the other way around, and similarly for DBPs.

Example 3.3 The paging problem In the paging problem we have a two-level memory hierarchy: A slow memory that contains n different pages, and a cache that contains at most k different pages. Typically, $k\ll n$. Pages that are in the cache can be accessed at zero cost. If a request is made to access a page that is not in the cache, a *page fault* occurs and the page should be brought into the cache, at a cost of 1. If the cache is full, some other page should first be evicted from the cache. The paging problem is that of deciding which pages to keep in the cache in order to minimize the number of page faults.

Let $[n]=\{1,\ldots,n\}$. A paging problem with parameters n and k induces the WFA $\mathcal{A}=\langle[n],Q,\Delta,\emptyset,\tau\rangle$, where $Q=\{C\subseteq[n]:|C|\leq k\}$ is the set of all possible cache configurations and there is a transition $t=\langle C,c,C'\rangle\in\Delta$ iff (1) $c\in C$ in which case page c is in the cache, thus $C'=C$ and $\tau(t)=0$, (2) $c\notin C$ and $|C|<k$ in which case there is a page fault and the cache is not full, thus $C'=C\cup\{c\}$ and $\tau(t)=1$, and (3) $c\notin C$ and $|C|=k$ in which case there is a page fault and we evict some page $c'\in C$ from the cache and replace it with c, thus $C'=(C\setminus\{c'\})\cup\{c\}$ and $\tau(t)=1$.

An offline algorithm knows the sequence of requests and advance and thus evicts pages according to their need in the future. Is is shown in [8] that every online deterministic paging algorithm achieves a competitive ratio of at least k. As a result, for every memory set $\mathcal{M}$, every DBP of $\mathcal{A}\times\mathcal{M}$ achieves an approximation of at least k. There are quite a few deterministic algorithms that

are optimal [8], two of which are "first in first out" (FIFO) and "least recently used" (LRU). Both algorithms use memory and we are not aware of optimal deterministic algorithms that do not.

In the probabilistic setting, the algorithm RANDOM that uses no memory achieves a competitive ratio of k; when a page fault occurs, choose a page uniformly at random and evict it. The stochastization that corresponds to RANDOM is given by the distribution function D_1 that is defined as follows. Consider $C \in Q$ such that $|C| = k$ and $c \notin C$. There are k c-labeled outgoing transitions from C corresponding to the k candidate pages to be evicted from C. For $C' \in Q$ such that $\langle C, c, C' \rangle \in \Delta$, we define $D_1(C, c, C') = \frac{1}{k}$. By the above, $\mathcal{A}^{D_1}$ k-approximates $\mathcal{A}$.

The optimal competitive ratio for a probabilistic algorithm is the harmonic number $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} \approx \log(k)$. We describe the algorithm MARK, which achieves a slightly worst competitive ratio of $2H_k$. Initially, all pages are unmarked. When a page is requested, it becomes marked. When a page fault occurs and the cache is full, select uniformly at random a page that is not marked and evict it. If all pages in the cache are marked and a page fault occurs, unmark all the pages, evict one of them as in the above, and insert the requested page marked.

In order to implement the algorithm, memory is needed. Assume there is an order on the pages, e.g., the standard order on numbers. We use a memory $\mathcal{M}$ with states $2^{[k]}$. Consider a state $\langle C, m \rangle$ of $\mathcal{A} \times \mathcal{M}$, with $C = \{c_1, \ldots, c_k\}$ such that $c_1 < c_2 < \ldots < c_k$. The memory state m represents the unmarked pages in C, thus for $1 \leq i \leq k$, the page c_i is unmarked iff $i \in m$. We describe the stochastization that corresponds to MARK by means of the distribution function D_2 that is defined as follows. Consider a state $\langle C, m \rangle$ where $C = \{c_1, \ldots, c_k\}$, and a request $c \notin C$. Thus, if c is read at configuration $\langle C, m \rangle$ a page fault occurs, and since $|C| = k$, we need to evict one of the pages in C. Consider $j \in m$, thus c_j is an unmarked page in the cache and it is a candidate for eviction when c is requested. Let C' be the state in which we replace c with c_j, thus $C' = (C \setminus \{c_j\}) \cup \{c\}$. Since c is inserted marked to the cache its index in C' does not appear the new memory state m', thus $m' = \{i \in m : c_i < c\} \cup \{i + 1 : i \in m \text{ and } c < c_i\}$. There are $|m|$ unmarked pages in C, so the probability of evicting c_j is $\frac{1}{|m|}$, thus we define $D_2(\langle C, m \rangle, c, \langle C', m' \rangle) = \frac{1}{|m|}$. Every outgoing transition from $\langle C, m \rangle$ that is not of this type gets probability 0. By the above, $(\mathcal{A} \times \mathcal{M})^{D_2}$ $2H_k$-approximates $\mathcal{A}$. ∎

3.2 Approximated Determinization

Not all WFAs can be determinized. Since some applications require deterministic automata, one way to cope with WFAs that cannot be determinized is to α-*determinize* them, namely construct a DWFA that α-approximates them, for $\alpha \geq 1$. Our second application is an extension of the class of WFAs that can be approximately determinized.

In [23], Mohri describes a determinization construction for a subclass of WFAs – these that have the *twins property*. In [4], the authors define the α-*twins property*, for $\alpha \geq 1$, and describe an α-determinization construction for WFAs that satisfy it. We briefly define the properties below. Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$ and two states $q_1, q_2 \in Q$. We say that q_1 and q_2 are

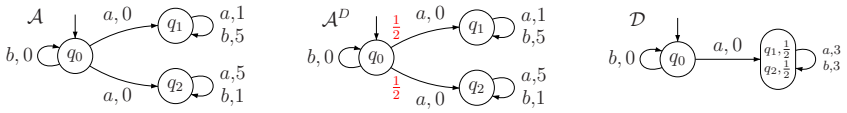

Figure 2: An illustration of our algorithm for approximated determinization of WFAs.

pairwise reachable if there is a word $u \in \Sigma^*$ such that there are runs of $\mathcal{A}$ on u that end in q_1 and q_2. Also, we say that q_1 and q_2 have the t-twins property if they are either not pairwise reachable, or, for every word $v \in \Sigma^*$, if π_1 and π_2 are v-labeled cycle starting from q_1 and q_2, respectively, then $val(\pi_1) \leq t \cdot val(\pi_2)$. We say that $\mathcal{A}$ has the α-twins property iff every two states in $\mathcal{A}$ have the α-twins property. The α-twins property coincides with Mohri's twins property when $\alpha = 1$.

The α-twins property can be thought of as a *lengthwise* requirement; there might be many runs on a word, but there is a bound on how different the runs are. Our algorithm applies to CA-WFAs. Recall that such WFAs have a dual, *widthwise*, property: the number of runs on a word is bounded by some constant. The algorithm proceeds as follows. Given a CA-WFA $\mathcal{A}$, we first find an α-stochastization $\mathcal{A}^D$ of it. Since stochastization maintains constant ambiguity, we obtain a CA-PWFA. As we show in Theorem 5.4, CA-PWFA can be determinized, thus we find an α-determinization of $\mathcal{A}$.

Example 3.4 Consider the WFA $\mathcal{A}$ that is depicted in Fig. 2. Note that $\mathcal{A}$ is 2-ambiguous. The optimal stochastization of $\mathcal{A}$ is given by the distribution function D that assigns $D(q_0, a, q_1) = D(q_0, a, q_2) = \frac{1}{2}$. The resulting PWFA $\mathcal{A}^D$ is also depicted in the figure. Then, we construct the DWFA $\mathcal{D}$ by applying the determinization construction of Theorem 5.4. Clearly, the DWFA $\mathcal{D}$ 3-approximates $\mathcal{A}$.

We note that $\mathcal{A}$ has the 5-twins property, and this is the minimal t. That is, for every $t < 5$, $\mathcal{A}$ does not have the t-twins property. The DWFA $\mathcal{D}'$ that is constructed from $\mathcal{A}$ using the approximated determinization construction of [4] has the same structure as $\mathcal{D}$ only that the self loops that have weight 3 in $\mathcal{D}$, have weight 5 in $\mathcal{D}'$. Thus, $\mathcal{D}'$ 5-approximates $\mathcal{A}$. ■

4 Stochastization of General WFAs

In this section we study the AS and OAS problems for general WFA. We start with some good news, showing that the exact stochastization problem can be solved efficiently. Essentially, it follows from the fact that exact stochastization amounts to determinization by pruning, which can be solved in polynomial time [2].

Theorem 4.1 *The exact stochastization problem can be solved in polynomial time.*

Proof: Recall that we say that a DWFA $\mathcal{D}$ is a DBP of $\mathcal{A}$ if $\mathcal{D}$ is obtained by removing transitions from $\mathcal{A}$ until a DWFA is formed. It is shown in [2] that deciding, given a WFA $\mathcal{A}$, whether there is an equivalent DBP of $\mathcal{A}$, can be solved in polynomial time. We claim that there is an equivalent DBP of $\mathcal{A}$ iff there is a distribution function D such that $L(\mathcal{A}^D) = L(\mathcal{A})$. Since a DBP of $\mathcal{A}$ is a stochastization of $\mathcal{A}$, the first direction is easy. For the second direction, given a distribution function D such that $L(\mathcal{A}^D) = L(\mathcal{A})$ we construct a DWFA $\mathcal{D}$ by a DPB of $\mathcal{A}$ by arbitrarily choosing, for every $q \in Q$ and $\sigma \in \Sigma$, a transition $e = \langle q, \sigma, q' \rangle \in \Delta$ with $D(e) > 0$ and removing all other outgoing σ-labeled transitions from q. We claim that $L(\mathcal{D}) = L(\mathcal{A})$. Indeed, otherwise there is a word $w \in \Sigma^*$ such that $val(\mathcal{D}, w) < val(\mathcal{A}, w)$ and the run of $\mathcal{D}$ on w gets a positive probability in $\mathcal{A}^D$. Thus, $val(\mathcal{A}^D, w) < val(\mathcal{A}, w)$, and we are done. ∎

We proceed to the bad news.

Theorem 4.2 *The AS problem is undecidable.*

Proof: In Section 1 we mentioned PFA, which add probabilities to finite automata. Formally, a PFA is $\mathcal{P} = \langle \Sigma, Q, P, q_0, F \rangle$, where $F \subseteq Q$ is a set of accepting states and the other components are as in PWFAs. Given a word $w \in \Sigma^*$, each run of $\mathcal{P}$ on w has a probability. The value $\mathcal{P}$ assigns to w is the probability of the accepting runs. We say that $\mathcal{P}$ is *simple* if the image of P is $\{0, 1, \frac{1}{2}\}$. For $\lambda \in [0, 1]$, the λ-emptiness problem for PFAs gets as input a PFA $\mathcal{P}$, and the goal is to decide whether there is a word $w \in \Sigma^*$ such that $val(\mathcal{P}, w) > \lambda$. It is well known that the emptiness problem for PFAs is undecidable for $\lambda \in (0, 1)$ [6, 22]. Furthermore, it is shown in [15] that the emptiness problem is undecidable for simple PFAs and $\lambda = \frac{1}{2}$. In the full version we construct, given a simple PFA $\mathcal{P}$, a WFA $\mathcal{A}$ such that $\mathcal{P}$ is $\frac{1}{2}$-empty iff there is an $\frac{2+\sqrt{7}}{3}$-stochastization of $\mathcal{A}$.

Consider a simple PFA $\mathcal{P} = \langle \Sigma, Q, P, q_0, F \rangle$. Let $\alpha \geq \frac{2+\sqrt{7}}{3} \approx 1.55$. We construct a WFA $\mathcal{A} = \langle \Sigma', S, \Delta, s_0, \tau \rangle$ such that $\mathcal{P}$ is $\frac{1}{2}$-empty iff there is an α-stochastization of $\mathcal{A}$. The alphabet of $\mathcal{A}$ is $\Sigma' = \Sigma \cup Q \cup \{\$, \#\}$ and its states are $S = S_L \cup S_R \cup \{s_0, s_{sink}\}$, where $S_R = Q$. We refer to S_L as the *left* component and to S_R as the *right* component.

Intuitively, consider a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. Consider a word $w \in \Sigma^*$. We define $\mathcal{A}$ so that every run r of $\mathcal{P}$ on w has a corresponding run r' of $\mathcal{A}^D$ on the word $\$w\$$, where (1) $\Pr[r] = \Pr[r']$, (2) if r is accepting, then $val(r') = \gamma$ and $val(r') = \gamma + 1$ otherwise, for $\gamma = 1.5\alpha - \frac{1}{2}$. Combining the two, we have $val(\mathcal{A}^D, w) = (1+\gamma) \cdot \Pr[accept(\mathcal{P}, w)] + \gamma \cdot \Pr[reject(\mathcal{P}, w)]$. Since $\Pr[reject(\mathcal{P}, w)] + \Pr[accept(\mathcal{P}, w)] = 1$, we have $val(\mathcal{A}^D, w) = \gamma + \Pr[accept(\mathcal{P}, w)]$. Finally, we define $\mathcal{A}$ so that $val(\mathcal{A}, \$w\$) = 1.5$. We show that $\mathcal{P}$ is $\frac{1}{2}$-empty. Since $\mathcal{A}^D$ α-approximates $\mathcal{A}$, we have $val(\mathcal{A}^D, \$w\$) = \gamma + \Pr[accept(\mathcal{P}, w)] \leq 1.5 \cdot \alpha = val(\mathcal{A}, w) \cdot \alpha$. By our choice of γ, we have $\Pr[accept(\mathcal{P}, w)] \leq \frac{1}{2}$, and we are done.

We describe the intuition of the reduction (see an illustration of $\mathcal{A}$ in Fig. 3). Consider a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. Consider a word $w \in \Sigma^*$. We define $\mathcal{A}$ so that every run r of $\mathcal{P}$ on w has a corresponding run r' of $\mathcal{A}^D$ on the word $\$w\$$, where

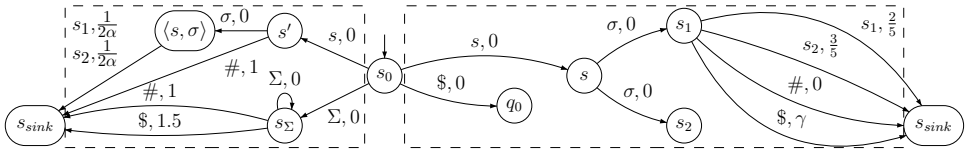

Figure 3: An illustration of the construction of a WFA $\mathcal{A}$ from a PFA $\mathcal{P}$. The probabilistic transition function P of $\mathcal{P}$ has $P(s, \sigma, s_1) = P(s, \sigma, s_2) = \frac{1}{2}$. Also, $s_2 \notin F$, thus $\tau(s_2, \$, s_{sink}) = \gamma$.

(1) $\Pr[r] = \Pr[r']$, (2) if r is accepting, then $val(r') = \gamma$ and $val(r') = \gamma + 1$ otherwise, for $\gamma = 1.5\alpha - \frac{1}{2}$. Finally, we define $\mathcal{A}$ so that $val(\mathcal{A}, \$w\$) = 1.5$. Before showing how we define $\mathcal{A}$ to have these properties, we show that they imply that $\mathcal{P}$ is $\frac{1}{2}$-empty. Combining (1) and (2), we have $val(\mathcal{A}^D, w) = (1+\gamma) \cdot \Pr[accept(\mathcal{P}, w)] + \gamma \cdot \Pr[reject(\mathcal{P}, w)]$. Since $\Pr[reject(\mathcal{P}, w)] + \Pr[accept(\mathcal{P}, w)] = 1$, we have $val(\mathcal{A}^D, w) = \gamma + \Pr[accept(\mathcal{P}, w)]$. Since $\mathcal{A}^D$ α-approximates $\mathcal{A}$, we have $val(\mathcal{A}^D, \$w\$) = \gamma + \Pr[accept(\mathcal{P}, w)] \leq 1.5 \cdot \alpha = val(\mathcal{A}, w) \cdot \alpha$. By our choice of γ, we have $\Pr[accept(\mathcal{P}, w)] \leq \frac{1}{2}$, and we are done.

Next, we describe the construction of $\mathcal{A}$. The right component is a WFA with the same structure as $\mathcal{P}$ in which all transitions have weight 0. That is, $S_R = Q$ and, for $s, s' \in S_L$ and $\sigma \in \Sigma$, there is a transition $t = \langle s, \sigma, s' \rangle \in \Delta$ iff $P(s, \sigma, s') > 0$, in which case $\tau(t) = 0$. Recall that unlike PFAs, in WFAs all states are accepting. We simulate acceptance by $\mathcal{P}$ as follows. We use the letter $\$ \in \Sigma' \setminus \Sigma$ to mark the end of a word over Σ. For every state $s \in S_R$, there is a transition from s to s_{sink} labeled $\$$. The weight of the transition is $\gamma + 1$ if s is accepting in $\mathcal{P}$, and is γ otherwise, for $\gamma = 1.5\alpha - \frac{1}{2}$. For technical reasons we use $\$$ to mark the start of a word over Σ, thus $\langle s_0, \$, q_0 \rangle \in \Delta$, where recall that s_0 and q_0 are the initial states of $\mathcal{A}$ and $\mathcal{P}$, respectively. So, there is a one-to-one correspondence between runs of $\mathcal{P}$ on a word $w \in \Sigma^*$ and runs of $\mathcal{A}$ on the word $\$w\$$ that proceed to the right component.

The *left* component of $\mathcal{A}$ has the following properties. (1) For a word $w \in \Sigma^*$, the cheapest run of $\mathcal{A}$ on the word $\$w\$$ proceeds to the left component and has value 1.5. Consider a distribution D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. Then, (2) D assigns probability 0 to every run that proceeds to the left component, and (3) D coincides with P on the transitions in the right component of $\mathcal{A}$, thus for every $t = \langle s, \sigma, s' \rangle \in \Delta$ such that $s, s' \in Q$ and $\sigma \in \Sigma$, we have $D(t) = P(t)$.

Formally, the states of $\mathcal{A}$ are $S = S_L \cup S_R \cup \{s_0, s_{sink}\}$, where $S_R = Q$ and $S_L = \{q' : q \in Q\} \cup \{\langle q, \sigma \rangle : q \in Q \text{ and } \sigma \in \Sigma\} \cup \{s_\Sigma\}$. We describe the transitions of $\mathcal{A}$ as well as their weights. We start with the right component. As in the above, for every $s, s' \in S_R$ and $\sigma \in \Sigma$, we have $\langle s, \sigma, s' \rangle \in \Delta$ iff $P(s, \sigma, s') > 0$. The weights of these transitions is 0. For $s \in S_R$ and

$\sigma \in \Sigma' \setminus \Sigma$, we have $\delta_{\mathcal{A}}(s, \sigma) = s_{sink}$. The weights of these transitions are

$$\tau(s, \sigma, s_{sink}) = \begin{cases} \gamma & \text{if } \sigma = \$ \text{ and } s \in F, \\ \gamma + 1 & \text{if } \sigma = \$ \text{ and } s \notin F, \\ \frac{2}{5} & \text{if } \sigma \in Q \text{ and } \sigma = s, \\ \frac{3}{5} & \text{if } \sigma \in Q \text{ and } \sigma \neq s, \\ 0 & \text{if } \sigma = \#. \end{cases}$$

All outgoing transitions from s_0 have weight 0. Recall that $\Sigma' = Q \cup \Sigma \cup \{\#, \$\}$. We specify them below.

$$\delta_{\mathcal{A}}(s_0, \sigma) = \begin{cases} \{q_0, s_\Sigma\} & \text{if } \sigma = \$, \\ \{q', q\} & \text{if } \sigma \in Q, \\ s_{sink} & \text{if } \sigma \in \Sigma \cup \{\#\}. \end{cases}$$

Finally, we describe in the left component of $\mathcal{A}$. Recall that we require WFAs to be full. We do not specify all the transitions in $\mathcal{A}$ implicitly. The ones we do not specify lead to s_{sink} and have a high value so that the cheapest run of $\mathcal{A}$ on the word that uses such a transition proceeds through the right component of $\mathcal{A}$. Consider a state $s \in S_L$. We define $\delta_{\mathcal{A}}(s, \#) = s_{sink}$ with weight 1. Recall that for $w \in \Sigma^*$, the minimal run of $\mathcal{A}$ on $\$w\$$ proceeds to the left component and has value 1.5. Thus, for $\sigma \in \Sigma$, we define $\delta_{\mathcal{A}}(s_\Sigma, \sigma) = s_\Sigma$ with weight 0. Also, $\delta_{\mathcal{A}}(s_\Sigma, \$) = s_{sink}$ with weight 1.5. For $q, q_1, q_2 \in Q$ and $\sigma \in \Sigma$ such that $P(q, \sigma, q_1) = P(q, \sigma, q_2) = \frac{1}{2}$, there are runs of $\mathcal{A}$ on the words $q\sigma q_1$ and $q\sigma q_2$ that turn left and have value $\frac{1}{2\alpha}$. Thus, For $q \in Q$ and $\sigma \in \Sigma$, we define $\delta_{\mathcal{A}}(q', \sigma) = \langle q, \sigma \rangle$ with weight 0 and, for $q' \in Q$ such that $P(q, \sigma, q') = \frac{1}{2}$, we have $\delta_{\mathcal{A}}(\langle q, \sigma \rangle, q') = s_{sink}$ with weight $\frac{1}{2\alpha}$. Note that the transitions in the left component are deterministic, thus there is exactly one legal distribution for these transitions. Finally, for every $\sigma \in \Sigma'$ we have $\delta_{\mathcal{A}}(s_{sink}, \sigma) = s_{sink}$ with value 0.

We claim that if there is a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$, then $\mathcal{P}$ is empty. The proof follows from the following two claims:

Claim 4.3 *Every outgoing transition from s_0 that leads to the left component has probability 0 under D.*

Claim 4.4 *The probability that D assigns to transitions in the right component coincides with P, thus for $t = \langle q, \sigma, p \rangle \in Q \times \Sigma \times Q$, we claim that $D(t) = P(t)$, where recall that $S_R = Q$.*

We show that $\mathcal{P}$ is $\frac{1}{2}$-empty, thus for every $w \in \Sigma^*$ we claim that $val(\mathcal{P}, w) \leq \frac{1}{2}$. Consider a word $w \in \Sigma^*$. Note that the runs of $\mathcal{A}$ on the word $\$w\$$ all proceed to the right component except for one run that proceeds to the left component through s_Σ and has value 1.5. Consider a run $r = t_1, r', t_2$ of $\mathcal{A}$ on $w\$$ that proceeds to the right component, where t_1 and t_2 are transitions

labeled \$. Note that r' is a run of $\mathcal{P}$ on w as $\delta_{\mathcal{A}}(s_0, \$) = q_0$. If r' is accepting then the value of r is $\gamma + 1$ and it is γ otherwise. Recall that $\gamma = 1.5\alpha - \frac{1}{2}$. Since $\alpha \geq 1.5$, we have $\gamma > 1.5$, thus $val(\mathcal{A}, \$w\$) = 1.5$ is attained by the run that proceeds to the left component, which by Claim 4.3 has probability 0 under D. Moreover, by Claim 4.4 we have that $\Pr[r']$ in $\mathcal{P}$ equals $Pr(r)$ in $\mathcal{A}^D$. Thus, $val(\mathcal{A}^D, \$w\$) = (\gamma + 1) \cdot val(\mathcal{P}, w) + \gamma \cdot (1 - val(\mathcal{P}, w)) = \gamma + val(\mathcal{P}, w)$. Combining with $val(\mathcal{A}^D, \$w\$) \leq \alpha \cdot val(\mathcal{A}, \$w\$) = \alpha \cdot 1.5$, we have $val(\mathcal{P}, w) \leq \frac{1}{2}$, and we are done.

We continue and prove the two claims. First, consider a state $s \in S_L$ and $\sigma \in \Sigma'$ such that $\langle s_0, \sigma, s\rangle \in \Delta$. We claim that $D(s_0, \sigma, s) = 0$. Recall that $\tau(s, \#, q_{sink}) = 1$ as $s \in S_L$. Note that there is $s' \in S_R \cap \delta_{\mathcal{A}}(s_0, \sigma)$ with $\tau(s_0, \sigma, s') = 0$. Since $s' \in S_R$, we have $\tau(s', \#, s_{sink}) = 0$. Thus, $val(\mathcal{A}, \sigma\#) = 0$. Assume towards contradiction that $D(s_0, \sigma, s) > 0$. Then, $val(\mathcal{A}^D, \sigma\#) \geq D(s_0, \sigma, s) > 0$, thus $\mathcal{A}^D$ does not α-approximate $\mathcal{A}$, and we reach a contradiction.

Next, consider $t = \langle q, \sigma, q'\rangle \in Q \times \Sigma \times Q$. We claim that $D(t) = P(t)$. Recall that $\mathcal{P}$ is simple, so $P(t) \in \{0, \frac{1}{2}, 1\}$. We distinguish between three cases. The cases in which $P(t) = 0$ and $P(t) = 1$ are trivial as D is consistent with Δ and must assign probability 0 and 1 to t, respectively. In the last case, there is a state $q'' \in Q$ such that $\delta_{\mathcal{P}}(q, \sigma) = \{q', q''\}$. Consider the words $w_1 = q\sigma q'$ and $w_2 = q\sigma q''$. Recall that we defined $\mathcal{A}$ so that there are three runs on each of these words. For $i = 1, 2$, there is one run on w_i that proceeds to the left component, traverses the states $s_0, q', \langle q, \sigma\rangle, s_{sink}$, and has value $\frac{1}{2\alpha}$. There are two runs that proceed to the right component. The first traverses the states s_0, q, q', s_{sink} and the second traverses the states s_0, q, q'', s_{sink}. The value of the first run for w_1 is $\frac{2}{5}$ and for w_2 it is $\frac{3}{5}$, and the values in the second run are opposite. Since $\alpha \geq 1.5$, the run that proceeds to the left component is the cheapest run on each of the words and $val(\mathcal{A}, w_1) = val(\mathcal{A}, w_2) = \frac{1}{2\alpha}$. By Claim 4.3, D assigns probability 0 to this run. We claim that $D(q, \sigma, q') = D(q, \sigma, q'') = \frac{1}{2}$. Otherwise, wlog, $D(q, \sigma, q') = \frac{1}{2} + \xi$, for $\xi > 0$. Then, we have $val(\mathcal{A}^D, w_2) = (\frac{1}{2} + \xi) \cdot \frac{3}{5} + (\frac{1}{2} - \xi) \cdot \frac{2}{5} = \frac{1}{2} + \frac{\xi}{5} > \alpha \cdot \frac{1}{2\alpha} = \alpha \cdot val(\mathcal{A}, w_1)$.

For the second direction, assume $\mathcal{P}$ is empty. Consider the distribution D that coincides with P. That is, D assigns probability 0 to transitions that lead left from s_0 and probability 1 to transitions that lead right. Also, for every $t \in Q \times \Sigma \times Q$, we have $D(t) = P(t)$. Note that all other transitions in $\mathcal{A}$ are deterministic and must be assigned probability 1 by D. We claim that $\mathcal{A}^D$ α-approximates $\mathcal{A}$.

Consider $w \in \Sigma'^*$. We go over the different cases and show that $val(\mathcal{A}^D, w) \leq \alpha \cdot val(\mathcal{A}, w)$. First, assume $w = q\sigma q'$ for $q, q' \in Q$ and $\sigma \in \Sigma$. We distinguish between two cases. In the first case, $P(q, \sigma, q') = \frac{1}{2}$. There are three runs of $\mathcal{A}$ on the word w. One run proceeds to the left component, has value $\frac{1}{2\alpha}$, and probability 0 under D. There are two runs that proceed to the right component. The first run traverses the states s_0, q, q', s_{sink}, and the second, assuming $\delta_{\mathcal{A}}(q, a) = \{q', q''\}$, traverses the states s_0, q, q'', s_{sink}. The first has value $\frac{2}{5}$ and the second has value $\frac{3}{5}$. As in the above, $\alpha \cdot val(\mathcal{A}, w) = \alpha \cdot \frac{1}{2\alpha} = val(\mathcal{A}^D, w)$. In the second case, $P(q, \sigma, q')$ is either 0 or 1. Then, the runs that proceed to the left component have value greater than $\frac{2}{5}$. So, the cheapest run on w is a run that proceeds to the right component and has value $\frac{2}{5}$. Moreover,

the runs that proceed to the right component have value at most $\frac{3}{5}$. Since $\alpha \geq 1.5$, we have $\alpha \cdot val(\mathcal{A}, w) = \alpha \cdot \frac{2}{5} \geq \frac{3}{5} \geq val(\mathcal{A}^D, w)$. The proof is similar for words in $Q \cdot \Sigma^* \cdot Q$.

Assume $w \in \$ \cdot \Sigma^* \cdot \$$. Similar to the above, we have $val(\mathcal{A}, w) = 1.5$ and $val(\mathcal{A}^D, w) = \gamma + val(\mathcal{A}, w) = 1.5\alpha - \frac{1}{2} + val(\mathcal{P}, w)$. Since $\mathcal{P}$ is empty, we have $val(\mathcal{P}, w) \leq \frac{1}{2}$, thus $\alpha \cdot val(\mathcal{A}, w) \geq val(\mathcal{A}^D, w)$.

Assume $w \in Q \cdot \Sigma^* \cdot \$$. Then, the cheapest run of $\mathcal{A}$ on w proceeds to the right component and has value at least γ. The maximal value of a run of $\mathcal{A}$ on w that proceeds to the right component is $\gamma + 1$. Thus, $\alpha \cdot val(\mathcal{A}, w) \geq \alpha \cdot \gamma$ and $val(\mathcal{A}^D, w) \leq \gamma + 1$. Recall that $\gamma = 1.5\alpha - \frac{1}{2}$. Since $\alpha \geq \frac{2+\sqrt{7}}{3}$, we have $1.5\alpha^2 - 2\alpha - \frac{1}{2} \geq 0$, thus $\alpha \cdot val(\mathcal{A}, w) \geq val(\mathcal{A}^D, w)$.

Assume $w \in Q \cdot \Sigma^* \cdot (\# + \epsilon)$ or $w = \$$. Then, all runs of $\mathcal{A}$ on w that proceed to the right component have value 0. These are the only runs that get a probability that is higher than 0 under D, so $val(\mathcal{A}, w) = val(\mathcal{A}^D, w) = 0$. Finally, since the value of the self loop of s_{sink} is 0, every other word $w' \in \Sigma'^*$ has a prefix w that is considered above, and has $val(\mathcal{A}, w) = val(\mathcal{A}, w')$ and $val(\mathcal{A}^D, w) = val(\mathcal{A}^D, w')$, and we are done. ∎

5 Stochastization of Constant Ambiguous WFAs

Recall that a CA-WFA has a bound on the number of runs on each word. We show that the AS problem becomes decidable for CA-WFAs. For the upper bound, we show that when the ambiguity is fixed, the problem is in PSPACE. Also, when we further restrict the input to be a *tree-like* WFAs, the OAS problem can be solved in polynomial time. We note that while constant ambiguity is a serious restriction, many optimization problems, including the ski-rental we describe here, induce WFAs that are constant ambiguous. Also, many theoretical challenges for WFAs like the fact that they cannot be determinized, apply already to CA-WFA. We start with a lower bound, which we prove in the full version by a reduction from 3SAT.

Theorem 5.1 *The AS problem is NP-hard for 7-ambiguous WFAs.*

Proof: Consider a 3CNF formula $\theta = C_1 \wedge \ldots \wedge C_m$ over the variables $X = \{x_1, \ldots, x_n\}$. Let $C = \{C_1, \ldots, C_m\}$. Consider $\alpha > 1$. We construct a 7-ambiguous WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$ such that $\mathcal{A}$ has an α-approximation iff θ is satisfiable.

We describe the components of $\mathcal{A}$. The alphabet is $\Sigma = C \cup \{\#, \$\}$. As in Theorem 4.2, the states of $\mathcal{A}$ consist of *left* and *right* components $Q = Q_L \cup Q_R \cup \{q_0, q_{sink}\}$, where $Q_R = X \cup \{x_{pos}, x_{neg} : x \in X\}$ and $Q_L = \{C'_i, C''_i : C_i \in C\}$. We describe the transitions and their weights. First, the outgoing transitions from q_0. For every clause $C_i \in C$, there are three outgoing transitions that lead right: $\langle q_0, C_i, x \rangle \in \Delta$ iff C_i has a literal in $\{x, \neg x\}$, and one transition that leads left $\langle q_0, C_i, C'_i \rangle \in \Delta$. The weights of all these transitions is 0. In the right component, for $x \in X$, there are two \$-labeled transitions $\langle x, \$, x_{pos} \rangle, \langle x, \$, x_{neg} \rangle \in \Delta$. For $C_i \in C$, there

is a transition $\langle x_{pos}, C_i, q_{sink}\rangle \in \Delta$. The weight of the transition is 1 if x is a literal of C_i and α otherwise. Similarly, the weight of the transition $\langle x_{neg}, C_i, q_{sink}\rangle$ is 1 if $\neg x$ is a literal of C_i and α otherwise. Finally, for $x \in X$, the weight of the transition $\langle x, \#, q_{sink}\rangle \in \Delta$ is 0. In the left component, there is are transitions $\langle C_i', \$, C_i''\rangle \in \Delta$ with weight 0 and $\langle C_i'', C_i, q_{sink}\rangle$ with weight $\frac{1}{\alpha}$. The weight of the transition $\langle C_i', \#, q_{sink}\rangle$ is 1. We do not specify the other transitions implicitly. They all lead to q_{sink}, where the ones in the right component have a low weight whereas the ones in the left component have a high weight. It is not hard to see that $\mathcal{A}$ is 7-ambiguous and its size is polynomial in n and m.

Assume there is a satisfying assignment $f : X \to \{0, 1\}$. We show that there is a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. In fact D is a determinization by pruning of $\mathcal{A}$. For every $C_i \in C$, there is a literal $\ell \in \{x, \neg x\}$ that is satisfied by f. We define $D(q_0, C_i, x) = 1$. For $x \in X$, if $f(x) = 1$, we define $D(x, \$, x_{pos}) = 1$, and if $f(x) = 0$, we define $D(x, \$, x_{neg}) = 1$. This completes the definition of D as the other transitions are deterministic. We claim that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. Clearly, all runs that proceed to the left component from q_0 have probability 0 under D. We show that for $w \in C \cdot \$ \cdot C$, we have $val(\mathcal{A}^D, w) \leq \alpha \cdot val(\mathcal{A}^D, w)$. We distinguish between two cases. In the first case $w = C_i\$C_i$, for some $C_i \in C$. Then, $val(\mathcal{A}, w) = \frac{1}{\alpha}$ and it is attained in a run that proceeds to the left component and thus gets probability 0. Recall that D is a DBP of $\mathcal{A}$. Thus, there is a single run of $\mathcal{A}^D$ on w with positive probability. The value of the run is 1, thus $val(\mathcal{A}^D, w) = 1 \leq \alpha \cdot \frac{1}{\alpha} = \alpha \cdot val(\mathcal{A}, w)$, and we are done. In the second case, $w = C_i\$C_j$, for $C_i \neq C_j \in C$. The minimal valued run of $\mathcal{A}$ on w proceeds to the right component and has value 1. On the other hand, the run of $\mathcal{A}^D$ on w has a value of at most α, thus we have $val(\mathcal{A}^D, w) \leq \alpha \leq \alpha \cdot 1 = val(\mathcal{A}, w)$, and we are done. It is not hard to see that every word $w \in \Sigma^*$ has a word in $C\$C$ as its prefix, in which case the values assigned by the two automata coincide with the above, or w does not have a word in $C\$C$ as its prefix in which $val(\mathcal{A}^D, w) = val(\mathcal{A}, w)$.

For the other direction, assume D is a distribution function such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. We define an assignment $f : X \to \{0, 1\}$ as follows. For $x \in X$, if $D(x, \$, x_{pos}) = 1$, then $f(x) = 1$, and $f(x) = 0$ otherwise. We claim that f is satisfying. Consider a clause C_i. Consider the word $w = C_i\$C_i$. Note that there are 7 runs of $\mathcal{A}$ on w; one that proceeds to the left component and six that proceed to the right component. Further note that the run that proceeds to the left component must have probability 0 as otherwise there is a run of $\mathcal{A}^D$ on the word $C_i\#$ with probability greater than 0. Since the value of such a run is 1 and $val(\mathcal{A}, C_i\#) = 0$, this would contradict the fact that $\mathcal{A}^D$ is an α-approximation of $\mathcal{A}$. Consider a run r that traverses the states q_0, x, ℓ, q_{sink} of $\mathcal{A}$ on w with positive probability, where $\ell \in \{x_{pos}, x_{neg}\}$. Since $\mathcal{A}^D$ α-approximates $\mathcal{A}$, and $val(\mathcal{A}, w) = \frac{1}{\alpha}$, the value of r must be 1. Thus, the literal ℓ appears in C_i. Moreover, the run that traverses the states $q_0, x, (\neg\ell), q_{sink}$ that has value $\alpha > 1$ has probability 0 in $\mathcal{A}^D$. Thus, $D(x, \$, \ell) = 1$, thus $f(\ell) = 1$ and C_i is satisfied, and we are done. ∎

Consider a k-ambiguous WFA $\mathcal{A}$, a factor α, and a distribution function D. When k is fixed, it is possible to decide in polynomial time whether $\mathcal{A}^D$ α-approximates $\mathcal{A}$. Thus, a tempting approach to show that the AS problem is in NP is to bound the size of the optimal distribution

function. We show below that this approach is doomed to fail, as the optimal distribution function may involve irrational numbers even when the WFA has only rational weights.

Theorem 5.2 *There is a 4-ambiguous WFA with rational weights for which every distribution that attains the optimal approximation factor includes irrational numbers.*

Proof: We start by describing the intuition of the construction. A first attempt to construct the WFA $\mathcal{A}$ would be to set pairs of nondeterministic choices $x_1, x_1', x_2, x_2', x_3$, and x_3' at states q_1, q_2, and q_3, respectively. Then, define $\mathcal{A}$ so there is a word $w_{1,2}$ that has three runs in $\mathcal{A}$. The three runs on $w_{1,2}$ perform the nondeterministic choices x_1', x_1 and x_2', and x_1 and x_2. Then, consider the words $w_{1,2}a$ and $w_{1,2}b$. The first two runs have value 1 on $w_{1,2}a$ and value 2 on $w_{1,2}b$, and the last run has value 1 on $w_{1,2}a$ and value 2 on $w_{1,2}b$. A distribution function D that minimizes the maximum of $val(\mathcal{A}^D, w_{1,2}a)$ and $val(\mathcal{A}^D, w_{1,2}b)$ assigns 1.5 in both cases. Since $val(\mathcal{A}, w_{1,2}a) = val(\mathcal{A}, w_{1,2}b) = 1$, the optimal approximation factor is at least 1.5. Moreover, a distribution that attains this approximation factor must assign probability $1/2$ to the first two runs and assign probability $1/2$ to the last run. Thus, we have a constraint $x_1 \cdot x_2 = 1/2$. We do the same trick twice more. Namely, we define $\mathcal{A}$ so that there are two more words $w_{1,3}$ and $w_{2,3}$ whose runs generate the constraints $x_1 \cdot x_3 = 1/2$ and $x_2 \cdot x_3 = 1/2$, respectively. A distribution D that satisfies all constraints must assign $D(x_1) = D(x_2) = D(x_3) = \frac{1}{\sqrt{2}}$.

However, there are two complications. First, consider the word w_1 whose prefix coincides with a prefix of $w_{1,2}$ and its suffix coincides with a suffix of $w_{2,3}$. There is a run on w_1 that traverses q_1, q_2, and q_3 therefore making the nondeterministic choices x_1, x_2, and x_3. Second, consider the word w_2 whose prefix coincides with $w_{2,3}$ and its suffix coincides with $w_{1,2}$. There are two runs of $\mathcal{A}$ on w_2 and each performs one nondeterministic choice; x_2 and x_2'. There is an extension of w_1 or w_2 with a or b such that the resulting word has a value of 1 in $\mathcal{A}$ and a value of more than 1.5 in $\mathcal{A}^D$, for any distribution D. The solution to the first problem is easier. One of the runs on w_1 performs only the nondeterministic choice x_1'. We assign a low value to this run, so that the average value of every word that has w_1 as a prefix is low. In order to solve the second problem, we add a fourth nondeterministic choice between x_4 and x_4' such that the path from the initial state to q_2 traverses x_4. Then, there is a third run on w_2 that chooses x_4', and we set its value to be low. Finally, we require an optimal distribution function to assign a positive value to x_4.

The full construction of the WFA $\mathcal{A}$ is depicted in Fig. 4. Its alphabet is $\{a, b\}$. Weights that are not stated are 0, and missing transitions lead with value 0 to the sink. Finally, the small states mark the sink and are there for ease of drawing.

First, we claim that the optimal approximation factor α^* is at least 1.5. Indeed, consider the words $w_1 = a^5a$ and $w_2 = a^5b$. For $i = 1, 2$, let r_1^i, r_2^i and r_3^i, be the three runs of $\mathcal{A}$ on w_i, where r_1^i performs the nondeterministic choice x_1', r_2^i performs the nondeterministic choices x_1 and x_2', and r_3^i performs the nondeterministic choices x_1 and x_2. Note that $val(r_1^1) = val(r_2^1) = val(r_3^2) = 1$ and $val(r_3^1) = val(r_1^2) = val(r_2^2) = 2$. Consider a distribution function D. We have $val(\mathcal{A}^D, w_i) = D(x_1') \cdot val(r_1^i) + D(x_1) \cdot D(x_2') \cdot val(r_2^i) + D(x_1) \cdot D(x_2) \cdot val(r_3^i)$. Clearly, a

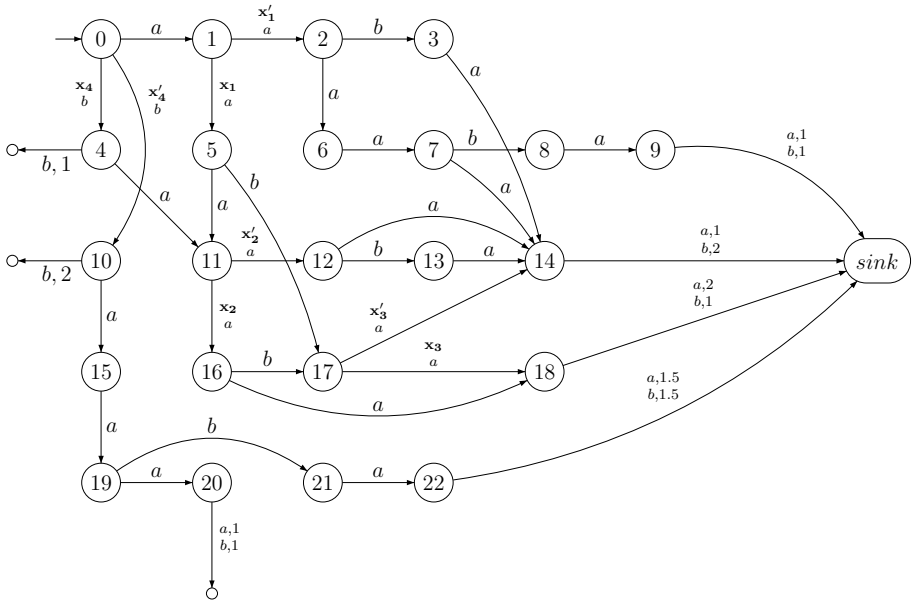

Figure 4: A 4-ambiguous WFA whose optimal stochastization is achieved with a distribution that includes real numbers.

distribution D that minimize the maximum of both expressions assigns $D(x'_1) + D(x_1) \cdot D(x'_2) = D(x_1) \cdot D(x_2) = 1/2$. Thus, $val(\mathcal{A}^D, w_i) = 1.5$ and since $val(\mathcal{A}, w_i) = 1$, we have $\alpha^* \geq 1.5$.

Next, we claim that there is a distribution function D^* that attains $\alpha^* = 1.5$. Let $D^*(x_1) = D^*(x_2) = D^*(x_3) = \frac{1}{\sqrt{2}}$ and $D^*(x_4) = \frac{1}{2}$. We claim that $\mathcal{A}^D$ is an 1.5-approximation of $\mathcal{A}$. We list below the *interesting* words in Σ^*, thus words that get a positive value in $\mathcal{A}$, and for every word $w \in \Sigma^*$ that gets a positive value, there is a word w' that is a prefix of w and both words get the same values in $\mathcal{A}$ and $\mathcal{A}^D$. It is not hard to see that the values $\mathcal{A}$ assigns to all the words in the list is 1, thus we verify that $\mathcal{A}^D$ assigns a value of at most 1.5 to these words.

- $val(\mathcal{A}^D, a^5a) = D(x'_1) \cdot 1 + D(x'_1) \cdot D(x'_2) \cdot 1 + D(x_1) \cdot D(x_2) \cdot 2 = 1.5$.
- $val(\mathcal{A}^D, a^5b) = D(x'_1) \cdot 2 + D(x'_1) \cdot D(x'_2) \cdot 2 + D(x_1) \cdot D(x_2) \cdot 1 = 1.5$.
- $val(\mathcal{A}^D, aabaa) = D(x'_1) \cdot 1 + D(x'_1) \cdot D(x'_3) \cdot 1 + D(x_1) \cdot D(x_3) \cdot 2 = 1.5$.
- $val(\mathcal{A}^D, aabab) = D(x'_1) \cdot 2 + D(x'_1) \cdot D(x'_3) \cdot 2 + D(x_1) \cdot D(x_3) \cdot 1 = 1.5$.
- $val(\mathcal{A}^D, baabaa) = D(x'_4) \cdot 1.5 + D(x_4) \cdot D(x'_2) \cdot 1 + D(x_4) \cdot D(x_2) \cdot D(x'_3) \cdot 1 + D(x_4) \cdot D(x_2) \cdot D(x_3) \cdot 2 = 1.5$.

- $val(\mathcal{A}^D, baabab) = D(x'_4) \cdot 1.5 + D(x_4) \cdot D(x'_2) \cdot 2 + D(x_4) \cdot D(x_2) \cdot D(x'_3) \cdot 2 + D(x_4) \cdot D(x_2) \cdot D(x_3) \cdot 1 = 1.5$.
- $val(\mathcal{A}^D, bb) = D(x_4) \cdot 1 + D(x'_4) \cdot 2 = 1.5$.
- $val(\mathcal{A}^D, baaaa) = D(x'_4) \cdot 1 + D(x_4) \cdot D(x'_2) \cdot 1 + D(x_4) \cdot D(x_2) \cdot 2 \approx 1.354$.
- $val(\mathcal{A}^D, baaab) = D(x'_4) \cdot 1 + D(x_4) \cdot D(x'_2) \cdot 2 + D(x_4) \cdot D(x_2) \cdot 1 \approx 1.146$.
- $val(\mathcal{A}^D, a^4baa) = D(x'_1) \cdot 1 + D(x_1) \cdot D(x'_2) \cdot 1 + D(x_1) \cdot D(x_2) \cdot D(x'_3) \cdot 1 + D(x_1) \cdot D(x_2) \cdot D(x_3) \cdot 2 \approx 1.354$.
- $val(\mathcal{A}^D, a^4bab) = D(x'_1) \cdot 1 + D(x_1) \cdot D(x'_2) \cdot 2 + D(x_1) \cdot D(x_2) \cdot D(x'_3) \cdot 2 + D(x_1) \cdot D(x_2) \cdot D(x_3) \cdot 1 \approx 1.354$.

To conclude the proof, consider a distribution function D such that $\mathcal{A}^D$ is an 1.5-approximation of $\mathcal{A}$. We claim that $D(x_1) = D(x_2) = D(x_3) = \frac{1}{\sqrt{2}}$, thus D consists of real numbers. As in the above, $val(\mathcal{A}^D, a^5a) = val(\mathcal{A}^D, a^5b) = 1.5$, thus $D(x_1) \cdot D(x_2) = 1/2$. Similarly, $val(\mathcal{A}^D, aabaa) = val(\mathcal{A}^D, aabab) = 1.5$, thus $D(x_1) \cdot D(x_3) = 1/2$. Combining the two, we have $D(x_2) = D(x_3)$. Next, note that $D(x_4) \geq 1/2$, as otherwise $val(\mathcal{A}^D, bb) > 1.5$. Combining with $val(\mathcal{A}^D, baabaa) \leq 1.5$ and $val(\mathcal{A}^D, baabab) \leq 1.5$, we get $D(x_2) = D(x_3) = \frac{1}{\sqrt{2}}$. Thus, $D(x_1) = \frac{1}{\sqrt{2}}$, and we are done. ∎

We now turn to show that the AS problem is decidable for CA-WFAs. Our proof is based on the following steps: (1) We describe a determinization construction for CA-PWFAs. The probabilities of transitions in the CA-PWFA affects the weights of the transitions in the obtained DWFA. (2) Consider a CA-WFA $\mathcal{A}$. We attribute each transition of $\mathcal{A}$ by a variable indicating the probability of taking the transition. We define constraints on the variables so that there is a correspondence between assignments and distribution functions. Let $\mathcal{A}'$ be the obtained CA-PWFA. Note that rather than being a standard probabilistic automaton, it is *parameterized*, thus it has the probabilities as variables. Since $\mathcal{A}'$ has the same structure as $\mathcal{A}$, it is indeed constant ambiguous. (3) We apply the determinization construction to $\mathcal{A}'$. The variables now appear in the weights of the obtained parameterized DWFA. (4) Given $\alpha \geq 1$, we add constraints on the variables that guarantee that the assignment results in an α-stochastization of $\mathcal{A}$. For that, we define a parameterized WFA $\mathcal{A}''$ that corresponds to $\alpha\mathcal{A} - \mathcal{A}'$ and the constraints ensure that it assigns a positive cost to all words. (5) We end up with a system of polynomial constraints, where the system is of size polynomial in $\mathcal{A}$. Deciding whether such a system has a solution is known to be in PSPACE.[3]

We now describe the steps in more detail.

Determinization of CA-WFAs Before we describe the determinization construction for CA-WFAs, we show that PWFAs are not in general determinizable. This is hardly surprising as neither WFAs

[3]The latter problem, a.k.a. the *existential theory of the reals* [10] is known to be NP-hard and in PSPACE. Improving its upper bound to NP would improve also our bound here.

nor PFAs are determinizable. We note, however, that the classic examples that show that WFAs are not determinizable use a 2-ambiguous WFA, and as we show below, 2-ambiguous PWFAs are determinizable. Formally, we have the following.

Theorem 5.3 *There is a PWFA with no equivalent DWFA.*

Proof: Consider the PWFA depicted in Fig. 5. Assume towards contradiction that there is a DWFA $\mathcal{D}$ that is equivalent to $\mathcal{A}$. Let γ be the smallest absolute value on one of $\mathcal{B}$'s transitions. Then, $\mathcal{B}$ cannot assign values of higher precision than γ. In particular, for $n \in \mathbb{N}$ such that $2^{-n} < \gamma$, it cannot assign the value 2^{-n} to the word $a^n b$. ∎

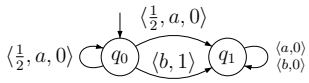

Figure 5: A PWFA with no equivalent DWFA.

Consider a PWFA $\mathcal{P} = \langle \Sigma, Q, D, q_0, \tau \rangle$. For a state $q \in Q$ and $\sigma \in \Sigma$, we say that $\mathcal{P}$ has a *σ-probabilistic choice* at q if there is a transition $\langle q, \sigma, q' \rangle \in \Delta_D$ such that $0 < D(\langle q, \sigma, q' \rangle) < 1$. This is indeed a choice as Δ_D must include a different σ-labeled outgoing transition from q with positive probability. We sometimes refer to such a choice simply as a *probabilistic choice*. We extend the definition to runs as follows. Consider a run $r = r_1, \ldots, r_n$ of $\mathcal{P}$ on some word. We say that r makes a probabilistic-choice at index $1 \leq i \leq n$ if there is a *label*(r_i)-choice at state *source*(r_i). We then say that r *chooses* the transition r_i. Note that when $\Pr[r] > 0$ and the probabilistic choices of r are $i_1, \ldots, i_\ell$, then $\Pr[r] = \prod_{1 \leq j \leq \ell} D(t_{i_j})$.

Given $\mathcal{P}$, we construct a DWFA $\mathcal{D} = \langle \Sigma, S, \Delta, q'_0, \tau' \rangle$ equivalent to $\mathcal{P}$ as follows. The states of $\mathcal{D}$ are $S \subseteq 2^{Q \times [0,1]}$. Thus, a state in S is a set of pairs, each consisting of a state q in $\mathcal{P}$ and the probability of reaching q. Thus, the construction is similar to the subset construction used for determinizing NFWs, except that each state in $\mathcal{P}$ is paired with the probability of visiting it in $\mathcal{D}$. More formally, consider such a state $s = \{\langle q_1, p_1 \rangle, \ldots, \langle q_\ell, p_\ell \rangle\}$ and a run r on a word $w \in \Sigma^*$ that ends in s. Then, for $1 \leq i \leq \ell$, we have $p_i = \Pr[\{r \in runs(\mathcal{P}, w) : r \text{ end in } q_i\}]$. We define the transitions and their weights accordingly: There is a transition $t = \langle s, \sigma, s' \rangle \in \Delta$ iff for every pair $\langle q'_i, p'_i \rangle \in s'$ we have $p'_i = \sum_{\langle q_j, p_j \rangle \in q} D(q_j, \sigma, q'_i) \cdot p_j$ and $p'_i > 0$. For $s \in S$, the states of s are $st(s) = \{q \in Q : \langle q, p \rangle \in s\}$. The weight of t is $\tau'(t) = \sum_{t' = \langle q_i, \sigma, q'_j \rangle \in st(q) \times \{\sigma\} \times st(q')} \tau(t') \cdot p_j \cdot D(t')$.

While it is not hard to see that $\mathcal{D}$ is equivalent to $\mathcal{P}$, there is no a-priori bound on the size of $\mathcal{D}$. In the full version we show that when $\mathcal{P}$ is constant ambiguous, then $\mathcal{D}$ has a finite number of states. Intuitively, we relate the probabilities that appear in the states of $\mathcal{D}$ with probabilistic choices that the runs of $\mathcal{P}$ perform. Then, we argue that a run of $\mathcal{P}$ on some word $w \in \Sigma^*$ performs at most $k-1$ probabilistic choices as every choice "spawns" another run of $\mathcal{P}$ on w and $|runs(\mathcal{P}, w)| \leq k$. Thus, we can bound the number of states in $\mathcal{D}$, implying the following.

Theorem 5.4 *Consider a k-ambiguous PWFA $\mathcal{P}$ with m transitions. There is a DWFA $\mathcal{D}$ that is equivalent to $\mathcal{P}$ with $m^{O(k^2)}$ states.*

Proof: Consider a CA-PWFA $\mathcal{P} = \langle \Sigma, Q, D, q_0, \tau \rangle$. Recall that the probability of a run $r = r_1, \ldots, r_n$ on some word is $\prod_{1 \leq i \leq n} D(r_i)$. Since the probability of every transition that is not a probabilistic choice is 1, we have the following.

Observation 5.5 *The probability of a run r that performs the probabilistic choices $t_1, \ldots, t_\ell$ is $Pr(r) = \prod_{1 \leq i \leq \ell} D(t_i)$.*

Next, we bound the number of probabilistic choices a run of $\mathcal{P}$ makes.

Observation 5.6 *Consider a run r of $\mathcal{P}$ with positive probability on some word $w \in \Sigma^*$. If r performs ℓ probabilistic choices, then there are at least $\ell + 1$ runs of $\mathcal{P}$ on w with positive probability.*

Proof: Let r be a run of $\mathcal{P}$ on $w = w_1 \ldots w_n \in \Sigma^*$. Assume that r performs the probabilistic choices $r_i = \langle q_{i-1}, w_i, q_i \rangle$, for $1 \leq i \leq n$. Since there is a w_i-choice at q_{i-1}, there is a state $q'_i \in Q$ such that $D(q_{i-1}, w_i, q'_i) > 0$. It is not hard to see that there is a run r' with positive probability on the suffix $w[i+1, \ldots, n]$ of w that starts in q'_i. Then, $r_1, r_2, \ldots r_{i-1}, r'$ is a run with positive probability of $\mathcal{P}$ on w. For every probabilistic choice we can find such a run. Including r, we have $\ell + 1$ different runs on w. ∎

Consider the DWFA $\mathcal{D} = \langle \Sigma, S, \Delta, q'_0, \tau' \rangle$ that is constructed from $\mathcal{P}$ as described in the body of the paper. We claim that $|S| = m^{O(k^2)}$. Consider a state $q = \langle \langle q_1, p_1 \rangle, \ldots, \langle q_\ell, p_\ell \rangle \rangle \in S$. Recall that for $1 \leq i \leq \ell$, the pair $\langle q_i, p_i \rangle$ consists of a state $q_i \in Q$ and a probability $p_i \in (0, 1]$. Furthermore, for a run r of $\mathcal{D}$ on $w \in \Sigma^*$ with *last*$(r) = q$, we have $p_i = \Pr[r' \in \textit{runs}(\mathcal{P}, w) : \textit{last}(r') = q_i]$. A corollary of Observation 5.6 is that $\mathcal{P}$ has no cycle with a probabilisic choice. Consider a run $r' \in \textit{runs}(\mathcal{P}, w)$ and let $P(r') \subseteq Q \times \Sigma \times Q$ be the probabilistic choices r' makes. Again, by Observation 5.6, we have $|P(r')| \leq k$. By Observation 5.5, we have $\Pr[r'] = D(P(r'))$.

Similarly to the above, each state in S consists of at most k pairs. In order to bound the number of states in S, we bound the number of possible probabilities that appear in the states in S. First, we bound the number of possible probabilities of the runs of $\mathcal{P}$. Let $P = \{\Pr[r'] : \exists w \in \Sigma^* \text{ with } r' \in \textit{runs}(\mathcal{P}, w)\}$. Recall that $|\Delta_D| = m$. Since every run makes at most k probabilistic choices, we have $|P| \leq \binom{m}{k}$. Since every probability that appears in a state in S is a sum of at most k numbers in P, there are at most $\binom{m}{k}^k$ such probabilities. Thus, there are at most $\binom{|Q|}{k} \cdot \binom{m}{k}^k$ states in S, which is $m^{O(k^2)}$, and we are done. ∎

Emptiness of WFAs Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$. We say that $\mathcal{A}$ is *empty* if for every $w \in \Sigma^*$, we have $val(\mathcal{A}, w) \geq 0$. Assume $|Q| = n$. We show how to decide whether $\mathcal{A}$ is empty. We define functions $f_0, f_1, \ldots, f_n : Q_\mathcal{A} \to \mathbb{R}$ as follows. We define $f_0(q) = 0$, for every $q \in Q_\mathcal{A}$, and, for $0 \leq i \leq n-1$, we define $f_{i+1}(q) = \min\big\{\{f_i(q)\} \cup \{\tau(t) + f_i(p) : t = \langle q, \sigma, p \rangle \in \Delta_\mathcal{A} \text{ and } \sigma \in \Sigma\}\big\}$. It is not hard to prove by induction that for every $q \in Q$ and $0 \leq i \leq n$, the

value $f_i(q)$ is the minimum between 0 and the value of the minimal run of length at most i that starts in q. These equations are known as *Bellman equations*. The following lemma is standard.

Lemma 5.7 *The WFA $\mathcal{A}$ is empty iff $f_{n-1}(q_0) = 0$ and $f_{n-1}(q) = f_n(q)$, for every $q \in Q$.*

An upper bound for the AS problem We are now ready to present the solution to the stochastization problem for CA-WFAs with weights in $\mathbb{Q}^+$. Given an input WFA $\mathcal{A}$ and a factor $\alpha \geq 1$, we construct a *polynomial feasibility problem*. The input to such a problem is a set of n variables X and polynomial constraints $P_i(\overline{x}) \leq 0$, for $1 \leq i \leq m$. The goal is to decide whether there is a point $\overline{p} \in \mathbb{R}^n$ that satisfies all the constraints, thus $P_i(\overline{p}) \leq 0$, for $1 \leq i \leq m$. The problem is known to be in PSPACE [10].

Theorem 5.8 *The AS problem for CA-WFAs is in PSPACE.*

Proof: We describe the intuition and the details can be found in the full version. Consider a k-ambiguous WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$. We associate with $\mathcal{A}$ the following constraints. First, let $X_D = \{x_t : t \in \Delta\}$ be a set of variables, and let $\mathcal{A}^D$ be a parameterized stochastization of $\mathcal{A}$ in which the probability of a transition t is x_t. Next, let $\mathcal{D}^D$ be the parameterized DWFA obtained by applying to $\mathcal{A}^D$ the determinization construction of Theorem 5.4. Since the variables of $\mathcal{A}^D$ are the probabilities of the transitions and in the determinization construction the probabilities move to the transition weights, the variables in $\mathcal{D}^D$ appear only in the weights.

The first type of constraints are ones that ensure that the assignment to the variables in X_D forms a probabilistic transition function. The second type depends on α and ensures that $\mathcal{D}^D \leq \alpha \cdot \mathcal{A}$. This is done by applying constraints that ensures the emptiness of the parameterized WFA $\mathcal{A}'' = (\alpha\mathcal{A} - \mathcal{D}^D)$. This is done by applying the constraints in Lemma 5.7 to the parameterized WFA $\mathcal{A}'' = (\alpha\mathcal{A} - \mathcal{D}^D)$. When $\mathcal{A}''$ has n states, we need a variable $f_i(q)$, for every $i = 0, \ldots, n$ and state q in $\mathcal{A}''$. The number of additional constraints is then polynomial in n and in the number of transitions of $\mathcal{A}''$. Thus, all in all we have polynomially many constraints and the numbers that appear in them are of size polynomial in the size of the weights of $\mathcal{A}$. Thus, the size of the program is polynomial in the size of $\mathcal{A}$, and we are done.

We describe the construction formally. Recall that $X_D = \{x_t : t \in \Delta\}$. For $t \in \Delta$, we have a constraint $0 \leq x_t \leq 1$, and, for $q \in Q$ and $\sigma \in \Sigma$, we have a constraint $\sum_{t=\langle q,\sigma,q' \rangle \in \Delta} x_t = 1$. Accordingly, an assignment $\nu : X_D \to \mathbb{R}$ corresponds to a distribution function D_ν, where $D_\nu(t) = \nu(x_t)$, for $t \in \Delta$.

Recall that $\mathcal{A}^D$ is the parameterized PWFA in which the probability of the transition t is x_t. Further recall that the parameterized DWFA $\mathcal{D}^D = \langle \Sigma, S, \Delta', s_0, \tau' \rangle$ is obtained by applying the determinization construction of Theorem 5.4 to $\mathcal{A}^D$.

Consider a state $s \in S$. Recall that s consists of pairs $\langle q_i, p_i \rangle$ where $q_i \in Q$ is a state of $\mathcal{A}^D$ and p_i is the probability of reaching q_i. That is, for a run r of $\mathcal{D}^D$ on a word $w \in \Sigma^*$ such that *last*$(r) =$

s, we have $p_i = \Pr[r' \in \mathit{runs}(\mathcal{A}^D, w) : \mathit{last}(r') = q_i]$. We claim that p_i is a polynomial of degree at most k. Consider a run $r' \in \mathit{runs}(\mathcal{A}^D, w)$ such that $\mathit{last}(r') = q_i$. By Observation 5.5, assuming r' makes the probabilistic choices $P \subseteq \Delta$, then $\Pr[r'] = \Pr[P] = \prod_{t \in \Delta} x_t$. By Observation 5.6, the run r' performs at most k probabilistic choices, thus $|P| \leq k$ and $\Pr[r']$ is a polynomial of degree at most k. It follows that q_i is a polynomial of degree at most k as it is a summation of polynomials of degree at most k.

We claim that the weights that are assigned by τ' are polynomials of degree at most k with coefficients that are weights in $\mathcal{A}$. Consider a transition $t = \langle s, \sigma, s' \rangle \in \Delta'$, and assume $s = \{\langle q_1, p_1 \rangle, \ldots, \langle q_\ell, p_\ell \rangle\}$. Recall that $\tau'(t) = \sum_{t' = \langle q_i, \sigma, q_i' \rangle \in st(s) \times \{\sigma\} \times st(s')} p_i \cdot D(t') \cdot \tau(t')$. We claim that each element in the summation is a polynomial of degree at most k. For $1 \leq i \leq \ell$, let t_i be a σ-labeled outgoing transition from q_i. Thus, $\mathit{target}(t_i)$ is a state in s'. If t_i is not a nondeterministic choice in $\mathcal{A}$, then $D(t_i) = 1$. By the above, p_i is a polynomial of degree at most k, and since $\tau(t_i) \in \mathbb{Q}$, we have $p_i \cdot \tau(t_i)$ is a polynomial of degree at most k. Next, if t_i is a nondeterministic choice, then p_i is a polynomial of degree strictly less than k as otherwise there is a run that performs more than k probabilistic choices, contradicting Observation 5.6. Thus, $p_i \cdot D(t_i) \cdot \tau(t_i) = p_i \cdot x_{t_i} \cdot \tau(t_i)$ is a polynomial of degree at most k. To conclude the proof, note that the coefficients in both cases are weights in $\mathcal{A}$.

Next, we construct the parameterized WFA $\mathcal{A}'' = \langle \Sigma, Q \times S, \Delta'', \langle q_0, s_0 \rangle, \tau'' \rangle$, where $t = \langle \langle q, s \rangle, \sigma, \langle q', s' \rangle \rangle \in \Delta''$ iff $t_1 = \langle q, \sigma, q' \rangle \in \Delta$ and $t_2 = \langle s, \sigma, s' \rangle \in \Delta'$ in which case $\tau''(t) = \alpha \cdot \tau(t_1) - \tau'(t_2)$. By the above, $\tau'(t_2)$ is a polynomial of degree at most k with coefficients that are weights in $\mathcal{A}$, thus $\tau''(t)$ is a polynomial of degree at most k with coefficients that are either weights in $\mathcal{A}$ of products of weights in $\mathcal{A}$ and α.

We return to the construction of the polynomial program. Let $n = |Q \times S|$. Recall that the set of variables of the program is X. We have already defined $X_D \subseteq X$. We now define the rest of the variables following Lemma 5.7. We have $\{f_i(q) : q \in Q \times S \text{ and } 0 \leq i \leq n\} \subseteq X$. We define constraints as follows. For $q \in Q \times S$, we have a constraint $f_0(q) = 0$. For $0 \leq i < n$, we have a constraint $f_{i+1}(q) \leq f_i(q)$, and for $t \in \Delta''$ having $\mathit{source}(t) = q$ and $\mathit{target}(t) = q'$, we have a constraint $f_{i+1}(q) \leq f_i(q') + \tau''(t)$. Finally, we have constraints $f_n(q) = f_{n-1}(q)$ and $f_{n-1}(\langle q_0, s_0 \rangle) = 0$. By the above, the constraints are polynomials of degree at most k with coefficients that are polynomials of the weights in $\mathcal{A}$ and α. Thus, the size of the system is polynomial in $\mathcal{A}$ and α.

We claim that the system is feasible iff there is a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$. For the first direction, assume the system is feasible, and let $\nu : X \to \mathbb{R}$ be an assignment that satisfies all the constraints. Recall that D_ν is the corresponding distribution function. Let $\mathcal{A}''_\nu$ be the (concrete) WFA that is obtained from $\mathcal{A}''$ by assigning to the variables the concrete values given by ν. Consider a word $w \in \Sigma^*$. It is not hard to see that $val(\mathcal{A}''_\nu, w) = \alpha \cdot val(\mathcal{A}, w) - val(\mathcal{A}^{D_\nu}, w)$. By Lemma 5.7, $\mathcal{A}''_\nu$ is empty, thus $val(\mathcal{A}''_\nu, w) \geq 0$. Combining the two, we have $\alpha \cdot val(\mathcal{A}, w) \geq val(\mathcal{A}^{D_\nu}, w)$, and we are done.

For the second direction, assume that there is a distribution function D such that $\mathcal{A}^D$ α-

approximates $\mathcal{A}$. We define an assignment $\nu : X \to \mathbb{R}$ and show that it satisfies the constraints. First, we define $\nu : X_D \to \mathbb{R}$ to coincide with D, thus $\nu(x_t) = D(t)$, for $t \in \Delta$. Let $\mathcal{A}''_\nu$ be the WFA as constructed in the above. Similarly to the above, since $\mathcal{A}^D$ α-approximates $\mathcal{A}$, we have that $\mathcal{A}''_\nu$ is empty. We complete the definition of ν. For $q \in Q \times S$, we define $\nu(f_0(q)) = 0$. For $1 \leq i \leq n$, we define $\nu(f_i(q)) = \min\left\{\nu(f_{i-1}(q)), \min\{\nu(f_{i-1}(p)) + \tau'(t)(\nu) : t = \langle q, \sigma, p\rangle \in \Delta''\}\right\}$. Clearly, $\nu(f_i(q)) \leq \nu(f_{i-1}(q))$ and $\nu(f_i(q)) \leq \nu(f_{i-1}(p)) + \tau'(t)(\nu)$, for every $t = \langle q, \sigma, p\rangle \in \Delta'$. Note that the other constraints are also satisfied; namely, $\nu(f_{n-1}(q_0)) = 0$ and, for $q \in Q'$, we have $\nu(f_{n-1}(q)) = \nu(f_n(q))$. Indeed, the definition of ν matches the algorithm to decide emptiness of WFAs, which precedes Lemma 5.7, and $\mathcal{A}''_\nu$ is indeed empty, and we are done. ∎

Remark 5.9 A different way to specify emptiness by constraints is to consider all simple paths and cycles in the automaton, as was done in [5]. A polynomial treatment of the exponentially many linear constraints then involves a separation oracle, and the ellipsoid method. The method we use here has polynomially many linear constraints to start with, and its implementation is considerably more practical. ∎

6 Solving the OAS problem for tree-like WFAs

We say that a WFA $\mathcal{A}$ is *tree-like* if the graph induced by its nondeterministic choices is a tree (note that $\mathcal{A}$ is constant ambiguous if the graph is acyclic). Thus, intuitively, for every nondeterministic choice t, there is one way to resolve other nondeterministic choices in order to reach t. Note that the WFA that corresponds to the ski-rental problem as well as the WFA from Example 3.4 are tree-like.

Formally, the graph $G_\mathcal{A}$ of nondeterministic choices of a WFA $\mathcal{A}$ is $\langle V, E\rangle$, where the set V of vertices consists of transitions of $\mathcal{A}$ that participate in a nondeterministic choice as well as a *source* node, thus $V = \{t \in \Delta : |\delta(\mathit{source}(t), \mathit{label}(t))| > 1\} \cup \{s\}$. Consider nondeterministic choices $t_1, t_2 \in \Delta$. There is a directed edge $\langle t_1, t_2\rangle \in E$ iff there is a deterministic path from $\mathit{target}(t_1)$ to $\mathit{source}(t_2)$. Also, there is an edge $\langle s, t_1\rangle \in E$ iff there is a (possibly empty) deterministic path from q_0 to $source(t_1)$. Note that E can include self-loops. A crucial observation is that when $\mathcal{A}$ is constant ambiguous, then $G_\mathcal{A}$ is a directed acyclic graph. We say that $\mathcal{A}$ is *tree-like* when $G_\mathcal{A}$ is a tree. When $G_\mathcal{A}$ is a tree, for every $v \in V \setminus \{s\}$, we say that $u \in V$ is the *parent* of v if u is the unique vertex such that $\langle v, u\rangle \in E$.

Recall that the OAS problem gets as input a WFA $\mathcal{A}$ and the goal is to find the minimal factor $\alpha \geq 1$ such that there is a distribution function D such that $\mathcal{A}^D$ is an α-approximation of $\mathcal{A}$. Clearly, the OAS problem is harder than the AS problem, which gets as input both a WFA and the approximation factor α. We show the following.

Theorem 6.1 *For every fixed $k \in \mathbb{N}$, the OAS problem can be solved in polynomial time for tree-like k-ambiguous WFAs.*

Proof: Consider a tree-like k-ambiguous WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, q_0, \tau \rangle$ and let $G_{\mathcal{A}} = \langle V, E \rangle$. We proceed similarly to Theorem 5.8. We construct a linear program whose optimal solution corresponds to the minimal approximation factor for $\mathcal{A}$. The variables are X, and we define $X_D = \{x_t : t \in V\} \subseteq X$. Also, $x_\alpha \in X$. We introduce constraints so that an assignment $\nu : X_D \to \mathbb{R}$ corresponds to a distribution function D_ν. Note that the constraints are slightly different than the ones we use in Theorem 5.8 and they rely on the fact that $G_{\mathcal{A}}$ is a tree. First, $x_\alpha \geq 1$ and $x_s = 1$. Consider $q \in Q$ and $\sigma \in \Sigma$ such that the σ-choices of q are $t_1, \ldots, t_\ell$, where $\ell > 1$. Note the σ-choices of q share a common parent $t \in V$. We introduce a constraint $\sum_{1 \leq i \leq \ell} x_{t_i} = x_t$.

Consider an assignment $\nu : X_D \to \mathbb{R}$ that satisfies the constraints above. We define the corresponding distribution function D_ν as follows. For $v \in V \setminus \{s\}$ let $u \in V$ be the parent of v. We define $D_\nu(v) = \frac{\nu(v)}{\nu(u)}$. It is not hard to see that D_ν is a distribution function. This definition of D_ν has the following important property. Consider a run r of $\mathcal{A}$ that performs the nondeterministic choices $t_1, \ldots, t_\ell$, where for $1 \leq i < \ell$, the transition t_i is the parent of t_{i+1} in $G_{\mathcal{A}}$. Then, $\Pr[r]$ in $\mathcal{A}^{D_\nu}$ is $\prod_{1 \leq i \leq \ell} D_\nu(t_i) = \frac{1}{\nu(t_1)} \frac{\nu(t_1)}{\nu(t_2)} \cdots \frac{\nu(t_{\ell-1})}{\nu(t_{\ell-2})} \frac{\nu(t_\ell)}{\nu(t_{\ell-1})} = \nu(t_\ell)$.

As in Theorem 5.8, we construct the parameterized PWFA $\mathcal{A}^D = \langle \Sigma, Q, D, q_0, \tau \rangle$ with the following change. For $t \in \Delta$, if t is not a nondeterministic choices in $\mathcal{A}$, the definition does not change and we define $D(t) = 1$. Otherwise, let $t' \in V$ be the parent of t. We define $D(t') = \frac{x_{t'}}{x_t}$.

Next, we construct the parameterized DWFA $\mathcal{A}' = \langle \Sigma, S, \Delta', s_0, \tau' \rangle$ by applying the determinization construction of Theorem 5.4. We claim that the weights assigned by τ' are polynomials of degree at most one, i.e., they are linear functions. Consider a transition $t = \langle s, \sigma, s' \rangle \in \Delta'$, where $s = \{\langle q_1, p_1 \rangle, \ldots, \langle q_\ell, p_\ell \rangle\}$. Consider a run r on a word $w \in \Sigma^*$ that ends in s. Recall that for $1 \leq i \leq \ell$, we have $p_i = \Pr[\{r' \in runs(\mathcal{P}, w) : \mathit{last}(r') = q_i\}]$. Note that since $\mathcal{A}$ is tree-like, there is exactly one run $r' \in runs(\mathcal{P}, w)$ with $\mathit{last}(r') = q_i$. As in the above, let t' be the last probabilistic choice r' performs, then $\Pr[r'] = x_{t'}$. Next, recall that $\tau'(t) = \sum_{t' = \langle q_i, \sigma, q_i' \rangle \in st(s) \times \{\sigma\} \times st(s')} p_i \cdot D(t') \cdot \tau(t')$. If t' is not a nondeterministic choice, then $D(t') = 1$ and $p_i \cdot \tau(t')$ is linear. Otherwise, $D(t') = \frac{x_{t'}}{p_i}$ and $p_i \cdot D(t') \cdot \tau(t') = x_{t'} \cdot \tau(t')$ is linear.

Finally, we construct the parameterized WFA $\mathcal{A}'' = \langle \Sigma, Q \times S, \Delta'', \langle q_0, s_0 \rangle, \tau'' \rangle$, where $t = \langle \langle q, s \rangle, \sigma, \langle q', s' \rangle \rangle \in \Delta''$ iff $t_1 = \langle q, \sigma, q' \rangle \in \Delta$ and $t_2 = \langle s, \sigma, s' \rangle \in \Delta'$ in which case $\tau''(t) = x_\alpha \cdot \tau(t_1) - \tau'(t_2)$. Since $\tau'(t_2)$ is linear and $\tau(t_1) \in \mathbb{Q}$, we have that $\tau''(t)$ is linear.

We define the other variables in X as well as the other constraints as in Theorem 5.8. We note that the constraints are linear, their size is polynomial in the size of $\mathcal{A}$, and there are polynomial many constraints. Thus, we can find an assignment ν that minimizes x_α in polynomial time. Let $\alpha = \nu(x_\alpha)$ and D_ν as defined above. It is not hard to show that $\mathcal{A}^{D_\nu}$ α-approximates $\mathcal{A}$. Moreover, if there is a factor $\alpha \geq 1$ and a distribution function D such that $\mathcal{A}^D$ α-approximates $\mathcal{A}$, then similarly to Theorem 5.8 we can find an assignment ν that satisfies the constraints and has $\nu(x_\alpha) = \alpha$. Thus, the optimal solution of the system coincides with the minimal approximation factor, and we are done. ∎

Corollary 6.2 *The OAS problem for* 2*-ambiguous WFAs can be solved in polynomial time.*

References

[1] S. Almagor, U. Boker, and O. Kupferman. Formalizing and reasoning about quality. In *Proc. 40th ICALP*, LNCS 7966, pages 15 – 27, 2013.

[2] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. *TALG*, 6(2), 2010.

[3] B. Aminof, O. Kupferman, and R. Lampert. Formal analysis of online algorithms. In *Proc. 9th ATVA*, LNCS 6996, pages 213–227, 2011.

[4] B. Aminof, O. Kupferman, and R. Lampert. Rigorous approximated determinization of weighted automata. *TCS*, 480:104–117, 2013.

[5] G. Avni and O. Kupferman. Parameterized weighted containment. *TOCL*, 16(1):6, 2014.

[6] P. Azaria. *Introduction to Probabilistic Automata*. Academic Press, Inc., 1971.

[7] U. Boker, K. Chatterjee, T.A. Henzinger, and O. Kupferman. Temporal specifications with accumulative values. In *Proc. 26th LICS*, pages 43–52, 2011.

[8] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[9] N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proc. 15th ESA*, pages 253–264, 2007.

[10] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proc. STOC*, pages 460–467, 1988.

[11] K. Culik and J. Kari. Digital images and formal languages. *Handbook of formal languages, vol. 3: beyond words*, pages 599–616, 1997.

[12] M. Droste, W. Kuich, and H. Vogler (eds.). *Handbook of Weighted Automata*. 2009.

[13] M. Droste and G. Rahonis. Weighted automata and weighted logics with discounting. *TCS*, 410(37):3481–3494, 2009.

[14] N. Fijalkow, H. Gimbert, F. Horn, and Y. Oualhadj. Two recursively inseparable problems for probabilistic automata. In *Proc. 39th MFCS*, LNCS 8634, pages 267–278, 2014.

[15] H. Gimbert and Y. Oualhadj. Probabilistic automata on finite words: Decidable and undecidable problems. In *Proc. 37th ICALP*, LNCS 6199, pages 527–538, 2010.

[16] T.A. Henzinger. From Boolean to quantitative notions of correctness. In *Proc. 37th POPL*, pages 157–158, 2010.

[17] R.A. Howard. Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes. Vol 2, Courier Corporation, 2013.

[18] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[19] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. D. Sleator. Competitive snoopy caching. *Algorithmica*, 3:77–119, 1988.

[20] S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. Worrell. On the complexity of equivalence and minimisation for q-weighted automata. *LMCS*, 9(1), 2013.

[21] D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *IJAC*, 4(3):405–425, 1994.

[22] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003.

[23] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[24] M. Mohri, F.C.N. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

[25] M. O. Rabin. Probabilistic automata. *I&C*, 6:230–245, 1963.

[26] M.Y. Vardi. Probabilistic Linear-Time Model Checking: An Overview of the Automata-Theoretic Approach. *ARTS*, pages 265-276, 1999.