Parameterized Weighted Containment

Guy Avni and Orna Kupferman

School of Computer Science and Engineering, Hebrew University, Israel

Abstract. Partially-specified systems and specifications are used in formal methods such as stepwise design and query checking. The goal of these methods is to explore the design or refine it by examining possible completions of the missing information. Existing methods consider a setting in which the systems and their correctness are Boolean. In recent years there has been growing interest and need for quantitative formal methods, where systems may be weighted and specifications may be multi valued. Weighted automata, which map input words to a numerical value, play a key role in quantitative reasoning. Technically, every transition in a weighted automaton $\mathcal{A}$ has a cost, and the value $\mathcal{A}$ assigns to a finite word w is the sum of the costs on the transitions participating in the most expensive accepting run of $\mathcal{A}$ on w. We study *parameterized weighted containment*: given three weighted automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, with $\mathcal{B}$ being partial, the goal is to find an assignment to the missing costs in $\mathcal{B}$ so that we end up with $\mathcal{B}'$ for which $\mathcal{A} \leq \mathcal{B}' \leq \mathcal{C}$, where $\leq$ is the weighted counterpart of containment. We also consider a one-sided version of the problem, where only $\mathcal{A}$ or only $\mathcal{C}$ are given in addition to $\mathcal{B}$, and the goal is to find a minimal assignment with which $\mathcal{A} \leq B'$ or, respectively, a maximal one with which $\mathcal{B}' \leq \mathcal{C}$. We argue that both problems are useful in stepwise design of weighted systems as well as approximated minimization of weighted automata.
We show that when the automata are deterministic, we can solve the problems in polynomial time. Our solution is based on the observation that the set of legal assignments to k missing costs forms a k-dimensional polytope. The technical challenge is to find an assignment in polynomial time even though the polytope is defined by means of exponentially many inequalities. We do so by using a powerful mathematical tool that enables us to develop a divide-and-conquer algorithm based on a separation oracle for polytopes. For nondeterministic automata, the weighted setting is much more complex, and in fact even non-parameterized containment is undecidable. We are still able to study variants of the problems, where containment is replaced by simulation.

1 Introduction

The *automata-theoretic* approach uses the theory of automata as a unifying paradigm for system specification and verification [23,25]. By viewing computations as words (over the alphabet of possible assignments to variables of the system), we can view both the system and its specification as languages. Questions like satisfiability of specifications or their satisfaction can then be reduced to questions about automata and their languages.

The automata-theoretic approach has proven useful also in reasoning about *partially-specified* systems and specifications, where some components are not known or hidden. Partially-specified systems are used mainly in *stepwise design*: One starts with a system with "holes" and iteratively completes them in a way that satisfies some specification

[9,10]. Reasoning about partially-specified systems is useful also in automatic partial synthesis [22] and program repair [14]. From the other direction, partially-specified specifications are used for system exploration. In particular, in *query checking* [5], the specification contains variables, and the goal is to find an assignment to the variables with which the explored system satisfies the specification. For example, solutions to the query $\text{ALWAYS}(X_1 \rightarrow \text{EVENTUALLY} grant)$ assign to X_1 events that trigger a generation of a grant in the system. Missing information in the system or the specification can be easily encoded in an automaton that models it, and indeed algorithms for the above problems are based on partially specified automata (c.f., [4]).

Traditional automata accept or reject their input, and are therefore Boolean. In recent years, there is growing need and interest in quantitative reasoning. *Weighted finite automaton* (WFA, for short) map words to numerical values. Technically, every transition in a weighted automaton $\mathcal{A}$ has a cost, and the value that $\mathcal{A}$ assigns to a finite word w, denoted $val(\mathcal{A}, w)$, is the sum of the costs of the transitions participating in the most expensive accepting run of $\mathcal{A}$ on w. [1] Applications of weighted automata include formal verification, where they are used for the verification of quantitative properties [6], as well as text, speech, and image processing, where the weights of the automaton are used in order to account for the variability of the data and to rank alternative hypotheses [8,19].

In the Boolean setting, formal verification amounts to checking containment of the language of the system by the language of the specification. This makes the *language-containment* problem of great theoretical and practical interest. In the weighted setting, the analogous problem gets as input two weighted automata $\mathcal{A}$ and $\mathcal{B}$, and decides whether all the words w that are accepted by $\mathcal{A}$ are also accepted by $\mathcal{B}$ and $val(\mathcal{A}, w) \leq val(\mathcal{B}, w)$. We denote this by $\mathcal{A} \subseteq \mathcal{B}$. Weighted automata are much more complicated than Boolean ones. The source of the difficulty is the infinite domain of values that the automata may assign to words. In particular, the problem of weighted containment is in general undecidable [1,17]. Given the importance of the problem, researchers have studied decidable fragments and approximations of weighted containment. We know, for example, that weighted containment is decidable, in fact polynomial, for deterministic WFAs (DWFAs, for short). For general WFA, researchers have suggested a weighted variant of the simulation relation, which approximates weighted containment and is decidable [3,7].

In this paper, we introduce and study *parameterized weighted containment*: given three weighted automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, with $\mathcal{B}$ being partial, the goal is to find an assignment to the missing costs in $\mathcal{B}$ so that we end up with $\mathcal{B}'$ for which $\mathcal{A} \subseteq \mathcal{B}' \subseteq \mathcal{C}$. We also consider a one-bounded version of the problem, where only $\mathcal{A}$ or only $\mathcal{C}$ are given in addition to $\mathcal{B}$, and the goal is to find a minimal assignment with which $\mathcal{A} \subseteq \mathcal{B}'$ or, respectively, a maximal one with which $\mathcal{B}' \subseteq \mathcal{C}$.[2]

Before we describe the technical details of the problems and their solutions, let us argue for their usefulness with two applications.

---

[1] In general, weighted automata may be defined with respect to all semirings. For our application here, we consider WFAs over $\mathbb{Q}$, with the sum of the semi-ring being $+$ and its product being $\max$.

[2] An orthogonal research direction is that of *parametric real-time reasoning* [2]. There, the quantitative nature of the automata origins from real-time constraints, the semantics is very different, and the goal is to find restrictions on the behavior of the clocks such that the automata satisfy certain properties.

Example 1. Stepwise design Assume we have a weighted specification $\mathcal{C}$. Refining the specification to an implementation involves a refinement of its Boolean behavior, possibly extending its alphabet, and an assignment of values to the refined computations. When the values in $\mathcal{C}$ exhibit upper bounds on costs, we want the implementation $\mathcal{B}$ to satisfy $\mathcal{B} \subseteq \mathcal{C}$. It is relatively easy to refine the Boolean behavior of $\mathcal{C}$ and get an automaton whose language, when restricted to the joint alphabet, is contained in the language of $\mathcal{C}$. It is much harder to design the weighted behavior of $\mathcal{B}$. For this, we apply one-bound parameterized weighted containment: $\mathcal{C}$ is the specification, $\mathcal{B}$ is its Boolean refinement, we label its costs by variables, and we are looking for a maximal assignment for the variables with which $\mathcal{B}$ is contained in $\mathcal{C}$.

For a specific example, consider the problem of ranking contributors to user-generated sites (e.g., Wikipedia). A big challenge for these sites is to develop trust in users. We seek a WFA that distinguishes between good and bad edits. After a user performs an edit on the site, the WFA gives it a score, and decisions on blocking and promotion of users are based on these scores.

We assume that an edit is a sequence of words – these added by the user. We also assume we have a tool, which we refer to as the *mapper*, that, intuitively, performs a pre-processing that abstracts the edit the user performed. More formally, the mapper maps words to some fixed alphabet, which is the alphabet of the WFA. For example, a mapper might map the sentence "The dog bent uver." to the word "$the \cdot noun \cdot verb \cdot misspelledword$."

The WFA combines heuristics, each of which either identifies a positive linguistic feature of a sentence or a negative one. An example of a positive heuristic is: "a sentence in which the multiplicity of the subject matches that of the verb should get a score greater than $1/4$". An example of a negative heuristic is: "a sentence in which *the* appears before a verb should not get a score above $1/2$".

Devising a WFA that takes care of a single heuristic is simple. However, since the automata are weighted, combining them is complicated. Some variants of parameterized weighted containment are useful here: when we want to combine two positive heuristics, modeled by WFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, we seek a minimal WFA $\mathcal{B}$ such that both $\mathcal{A}_1 \subseteq \mathcal{B}$ and $\mathcal{A}_2 \subseteq \mathcal{B}$. This variant of the one-bound problem is useful also when both heuristics are negative. Combining a negative heuristic $\mathcal{A}$ and a positive one $\mathcal{C}$ then corresponds to the problem of finding a WFA $\mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$.

Example 2. DWFA approximated minimization Minimization of Boolean deterministic automata is a well-studied problem. For DWFAs, Mohri described a (complicated yet polynomial) minimization algorithm [18]. We argue that two-bound parameterized weighted containment can be used in order to simplify Mohri's algorithm and, which we find more exciting, enables also *approximated minimization*. There, given a DWFA $\mathcal{A}$ and a factor $t \in \mathbb{Q}$, we would like to construct a minimal automaton $\mathcal{B}$ that has the same language as $\mathcal{A}$ and assigns values within a factor of t from $\mathcal{A}$. Given $\mathcal{A}$, we first construct the DWFAs $reduce(\mathcal{A}, t)$ and $increase(\mathcal{A}, t)$, for whatever definitions of reduce and increase we are after; for example, we can take $-t$ and $+t$ as additive factors to the value, or we can take $\frac{1}{t}$ and t as multiplicative ones. We then use parameterized weighted containment in order to find $\mathcal{B}$ such that $reduce(\mathcal{A}, t) \subseteq \mathcal{B} \subseteq increase(\mathcal{A}, t)$.

In both examples above, we left all the components of the generated WFA $\mathcal{B}$ unspecified. When the user has an idea about $\mathcal{B}$'s Boolean behavior, as is typically the case in step-wise refinement, this Boolean behavior is a natural starting point. In Section 3.2, we study the case only $\mathcal{A}$ and $\mathcal{C}$ are given, and we seek a minimal $\mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. We show that the problem is NP-complete for DWFAs, and suggest a heuristic for finding $\mathcal{B}$ that is based on an "on-demand" generation of the Boolean product of $\mathcal{A}$ and $\mathcal{C}$ (yak yak).

Let us now return to parameterized weighted containment where a partial WFA $\mathcal{B}$ is given. Our solution to the problem is based of strong mathematical tools. We explain here briefly the general idea for the two-bound problem for DWFAs. Consider an input $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ to the problem. Assume that transitions in $\mathcal{B}$ are parameterized by variables from a set $\mathcal{X}$ of size k. Recall that we are looking for a legal assignment $f : \mathcal{X} \to \mathbb{Q}$; that is, one with which $\mathcal{A} \subseteq \mathcal{B}^f \subseteq \mathcal{C}$, where $\mathcal{B}^f$ is the DWFA obtained by replacing each variable $X \in \mathcal{X}$ by $f(X)$. We first show that the products $\mathcal{A} \times \mathcal{B}$ and $\mathcal{B} \times \mathcal{C}$ can be used in order to generate a set of inequalities that the variables have to satisfy. For that, we characterize *critical paths* in the products – it is necessary and sufficient to restrict the assignment of the variables in transitions along these paths in order to guarantee that f is legal. Each critical path induces an inequality and together the inequalities induces a convex polytope $P \subseteq \mathbb{R}^k$ that includes exactly all the legal assignments. Khachiyan's *Ellipsoid's method* [16] then enables us to find a point in this polytope or conclude that no legal assignment exists.

This is, however, not the end of the story. Unfortunately, the number of critical paths we have to consider is exponential, making a naive search for the solution exponential too. Examining Khachiyan's method one can see that it is not necessary to have an implicit list of inequalities that define the polytope P. Indeed, it was shown in [11,15,20] that it is sufficient to have a *separation oracle* for the polytope. That is, instead of a list of inequalities that define P, the input to the problem is an oracle that, given a point $p \in \mathbb{Q}^k$, either says that $p \in P$ or returns a half-space $H \subseteq \mathbb{Q}^k$ such that $p \notin H$ and $P \subseteq H$. We show that we can use the products $\mathcal{A} \times \mathcal{B}$ and $\mathcal{B} \times \mathcal{C}$ in order to define such a separation oracle, leading to polynomial-time a solution to the problem.

For the one-bound variant, we show that the induced polytope is *pointed*, and that the solution we are after is a *vertex* of it, leading to an actually simpler algorithm. For the case the automata are nondeterministic, we argue that the one-bound problem is not interesting, as a minimal/maximal solution need not exist. For the two bound problem, we approximate containment by simulation, and show that the problem is NP-hard. Also, a polynomial algorithm for deciding weighted simulation would imply that it is NP-complete. [3] Given the computational difficulty of handling nondeterministic WFAs in general, we view these results as good news: parameterized language containment can be solved in polynomial time for the deterministic setting, and its approximation by simulation is decidable in the nondeterministic one.

2 Preliminaries

2.1 Weighted Automata

A nondeterministic finite weighted automaton on finite words (WFA, for short) is a tuple $\mathcal{A} = \langle \Sigma, Q, \Delta, Q_0, F, \tau \rangle$, where Σ is an alphabet, Q is a set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is

[3] The best algorithm currently known for weighted simulation is in NP $\cap$ co-NP [3].

a transition relation, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of accepting states, and $\tau : \Delta \to \mathbb{Q}$ is a function that maps each transition to a rational value, which is the cost of traversing this transition. We assume that there are no redundant states in $\mathcal{A}$. That is, all states are not empty (an accepting state is reachable from them) and accessible (reachable from an initial state).

A run of $\mathcal{A}$ on a word $u = u_1, \ldots, u_n \in \Sigma^*$ is a sequence of states $r = r_0, r_1, \ldots, r_n$ such that $r_0 \in Q_0$ and for every $0 \leq i < n$ we have $\Delta(r_i, u_{i+1}, r_{i+1})$. The run r is accepting iff $r_n \in F$. The value of the run, denoted $val(r, u)$, is the sum of costs of transitions it traverses. That is, $val(r, u) = \sum_{0 \leq i < n} \tau(\langle r_i, u_{i+1}, r_{i+1} \rangle)$. Similarly, for a path π, which is a sequence of transitions, we define $val(\pi) = \sum_{e \in \pi} \tau(e)$. Since $\mathcal{A}$ is non-deterministic, there can be more than one run on a word. We define the value that $\mathcal{A}$ assigns to $u \in \Sigma^*$, denoted $val(\mathcal{A}, u)$, as the value of the maximal-valued accepting run of $\mathcal{A}$ on u. That is, $val(\mathcal{A}, u) = max\{val(r, u) : r \text{ is an accepting run of } \mathcal{A} \text{ on } u\}$. As in NFAs, the language of $\mathcal{A}$, denoted $L(\mathcal{A})$, is the set of words in Σ^* that $\mathcal{A}$ accepts.

We say that $\mathcal{A}$ is *deterministic* if $|Q_0| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, there is at most one state $q' \in Q$ such that $\Delta(q, \sigma, q')$. Note that a deterministic WFA (DWFA, for short) has at most one run on every word in Σ^*.

Weighted containment For two WFAs $\mathcal{A}$ and $\mathcal{B}$, we say that $\mathcal{A}$ is contained in $\mathcal{B}$, denoted $\mathcal{A} \subseteq \mathcal{B}$, iff $L(\mathcal{A}) \subseteq L(\mathcal{B})$ and for every word $w \in L(\mathcal{A})$ we have that $val(\mathcal{A}, w) \leq val(\mathcal{B}, w)$. It is shown in [1,17] that deciding containment for WFAs is undecidable.

Negativeness We say that a WFA $\mathcal{A}$ is *negative* if $val(\mathcal{A}, w) \leq 0$ for every word $w \in L(\mathcal{A})$. We say that a path π in $\mathcal{A}$ is a *critical* path iff it is either a simple path from an initial state to an accepting state or a simple cycle. Keeping in mind that all states in $\mathcal{A}$ are not empty and reachable from an initial state, it is not hard to prove the following characterization of negative DWFAs.

Proposition 1. *A DWFA $\mathcal{A}$ is negative iff $val(v) \leq 0$ for every critical path v in $\mathcal{A}$.*

Let $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, \Delta_\mathcal{A}, q_{0_\mathcal{A}}, F_\mathcal{A} \rangle$ and $\mathcal{B} = \langle \Sigma, Q_\mathcal{B}, \Delta_\mathcal{B}, q_{0_\mathcal{B}}, F_\mathcal{B} \rangle$ be two DWFAs. Consider the product $\mathcal{S}_{\mathcal{A},\mathcal{B}} = \langle \Sigma, Q_\mathcal{B} \times Q_\mathcal{A}, \Delta_{\mathcal{A},\mathcal{B}}, \langle q_{0_\mathcal{A}}, q_{0_\mathcal{B}} \rangle, F_\mathcal{A} \times F_\mathcal{B}, \tau_{\mathcal{A},\mathcal{B}} \rangle$, where $\Delta_{\mathcal{A},\mathcal{B}}$ is such that $t = \langle \langle u, v \rangle, \sigma, \langle u', v' \rangle \rangle \in \Delta_{\mathcal{A},\mathcal{B}}$ iff $t_\mathcal{A} = \langle u, \sigma, u' \rangle \in \Delta_\mathcal{A}$ and $t_\mathcal{B} = \langle v, \sigma, v' \rangle \in \Delta_\mathcal{B}$. We refer to the transitions $t_\mathcal{A}$ and $t_\mathcal{B}$ as the transitions that are mapped to t. Then, for every $t \in \Delta_{\mathcal{A},\mathcal{B}}$, we define $\tau_{\mathcal{A},\mathcal{B}}(t) = \tau_\mathcal{B}(t_\mathcal{B}) - \tau_\mathcal{A}(t_\mathcal{A})$, where $t_\mathcal{A}$ and $t_\mathcal{B}$ are mapped to t.

Assume that $L(\mathcal{A}) \subseteq L(\mathcal{B})$ and consider a word $w \in \Sigma^*$. Let $r = \langle r_0^\mathcal{A}, r_0^\mathcal{B} \rangle, \ldots, \langle r_{|w|}^\mathcal{A}, r_{|w|}^\mathcal{B} \rangle$ be the run of $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ on w. It is easy to see that $r_0^\mathcal{A}, \ldots, r_{|w|}^\mathcal{A}$ is the run of $\mathcal{A}$ on w and $r_0^\mathcal{B}, \ldots, r_{|w|}^\mathcal{B}$ is the run of $\mathcal{B}$ on w. Thus, by the definition of the weight function of $\mathcal{S}_{\mathcal{A},\mathcal{B}}$, it follows that $val(\mathcal{S}_{\mathcal{A},\mathcal{B}}, w) = val(\mathcal{A}, w) - val(\mathcal{B}, w)$. Hence, we have the following proposition:

Proposition 2. *Let $\mathcal{A}$ and $\mathcal{B}$ be two DWFAs such that $L(\mathcal{A}) \subseteq L(\mathcal{B})$. Then, $\mathcal{A} \subseteq \mathcal{B}$ iff $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ is not negative.*

2.2 Parameterized Weighted Containment

Consider a set of variables $\mathcal{X} = \{X_1, \ldots, X_k\}$. An *$\mathcal{X}$-parameterized WFA* is a WFA in which some of the costs are replaced by variables from $\mathcal{X}$. Thus, the weight function is of the form $\tau : \Delta \to \mathbb{Q} \cup \mathcal{X}$. Given an $\mathcal{X}$-parameterized WFA $\mathcal{A}$ and an assignment $f : \mathcal{X} \to \mathbb{Q}$ to the variables in $\mathcal{X}$, we obtain the WFA $\mathcal{A}^f$ by replacing every variable $X \in \mathcal{X}$ with the value $f(X)$. Formally, the components of the WFA $\mathcal{A}^f$ agree with these of $\mathcal{A}$ except for the weight function τ^f, which agrees with τ on all transitions $t \in \Delta$ with $\tau(t) \in \mathbb{Q}$, and is such that $\tau^f(t) = f(X)$ for all $t \in \Delta$ with $\tau(t) = X$, for some $X \in \mathcal{X}$. Note that a variable $X \in \mathcal{X}$ may appear in more than one transition of $\mathcal{A}$.

Definition 1. *We consider the following three variants of parameterized weighted containment (PWC, for short).*

- **Two bound PWC:** *Given WFAs $\mathcal{A}$ and $\mathcal{C}$, and an $\mathcal{X}$-parameterized WFA $\mathcal{B}$, find an assignment $f : \mathcal{X} \to \mathbb{Q}$ such that $\mathcal{A} \subseteq \mathcal{B}^f \subseteq \mathcal{C}$.*
- **Least upper bound PWC:** *Given a WFA $\mathcal{A}$ and an $\mathcal{X}$-parameterized WFA $\mathcal{B}$, find a minimal assignment $f : \mathcal{X} \to \mathbb{Q}$ such that $\mathcal{A} \subseteq \mathcal{B}^f$.*
- **Greatest lower bound PWC:** *Given a WFA $\mathcal{C}$ and an $\mathcal{X}$-parameterized WFA $\mathcal{B}$, find a maximal assignment $f : \mathcal{X} \to \mathbb{Q}$ such that $\mathcal{B}^f \subseteq \mathcal{C}$.*

The least-upper and greatest-lower bound variants are dual and we refer to them as *one-bound* PWC. Their definition uses the terms minimal and maximal, with the expected interpretation: an assignment f is minimal if decreasing the value it assigns to a variable results in a violation of the requirement that $\mathcal{A} \subseteq \mathcal{B}^f$. Formally, f is minimal if for every variable $X \in \mathcal{X}$ and every $\epsilon > 0$, the assignment f' that agrees with f on all variables $X \neq X' \in \mathcal{X}$ and $f'(X) = f(X) - \epsilon$ is such that $\mathcal{A} \not\subseteq \mathcal{B}^{f'}$. Note that without the minimality requirement, the upper-bound variant is trivial: for every variable $X \in \mathcal{X}$ we set $f(X)$ to be a very high value, for example, the maximal cost appearing in $\mathcal{A}$ times the size of $|\mathcal{A}| \cdot |\mathcal{B}|$. The definition of a maximal assignment is dual.

Solving parameterized weighted containment is clearly harder than solving weighted containment, and is therefore undecidable in general. We study two restrictions of the problem. In Section 3, we study the PWC problem where the automata are deterministic. As hinted in Proposition 2, containment is decidable for DWFAs. In Section 4 we study the PWC problem where the automata are nondeterministic, but we replace the containment relation with its approximating relation of *simulation* [3], which is decidable.

2.3 Geometry in $\mathbb{R}^k$

We briefly review some definitions on polytopes. For more details and intuition, see [21].

Polytopes A *convex polytope* is a set in $\mathbb{R}^k$ that is the intersection of a finite number of *half-spaces*. Thus, it can be defined as the set of points $p \in \mathbb{R}^k$ that are solutions to a system of linear inequalities $Ax \leq b$, where $A \in \mathbb{Q}^{m,k}$ is an $m \times k$ matrix of rationals, $b \in \mathbb{Q}^m$, and $m \in \mathbb{N}$ is the number of inequalities. For example, the system of inequalities $2x_1 + 3x_2 \leq 7$, $5x_1 \leq 3$, and $4x_2 \leq 0$ corresponds to the following representation:

$$\begin{pmatrix} 2 & 3 \\ 5 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 7 \\ 3 \\ 0 \end{pmatrix}$$

Dimension We say that the points $p_1, \ldots, p_m \in \mathbb{R}^n$ are *affinely independent* iff the vectors $p_2-p_1, p_3-p_1, \ldots, p_m-p_1$ are independent. The dimension of a convex polytope $P \subseteq \mathbb{R}^k$, denoted $dim(P)$, is defined to be $l \leq k$ iff the maximal number of affinely independent points in P is $l+1$.

For example, consider the line in Figure 1, which is a convex polytope in $\mathbb{R}^2$. We claim that the dimension of the polytope is 1. Indeed, points a and b in the polytope are affinely independent, as the single vector $b-a$ is linearly independent. On the other hand, points a, b, and c are not affinely independent, as $b-a$ and $c-a$ are linearly dependent.

We say that a polytope $P \subseteq \mathbb{R}^k$ is *full-dimensional* if its dimension is k. When P is not full-dimensional, it is contained in a hyper-plane of dimension less than k.

Vertices In 2-dimensions, a vertex is the meeting point of two edges. In k-dimensions, a vertex is the meeting point of k *faces*, which are the k-dimensional generalization of edges. We say that a polytope P is *pointed* if it has a vertex. In Appendix A.1 we define these notions formally.

Geometrical objects A k-dimensional *ball* is a generalization of the 2-dimensional circle. For $c \in \mathbb{R}^k$ (the center) and $r \in \mathbb{R}$ (the radius) we define the ball $B(c, r) = \{p \in \mathbb{R}^k : \sum_{1 \leq i \leq k}(p_i - c_i)^2 \leq r^2\}$. Consider a polytope $P = \{x \in \mathbb{R}^k : Ax \leq b\}$. We say that P is *bounded* iff there is a ball with a finite radius that contains it.

Consider an *invertible linear transformation* $L : \mathbb{R}^k \to \mathbb{R}^k$. For example, *rotation* is invertible but the transformation $L(p) = 0$, for every $p \in \mathbb{R}^k$, is not. An *ellipsoid* with center $0 \in \mathbb{R}^k$, is the implication of L on the unit ball $B(0, 1)$. That is, it is the set $L(B(0, 1)) = \{L(p) \in \mathbb{R}^k : p \in B(0, 1)\}$. An ellipsoid centered at $c \in \mathbb{R}^k$, is the translation of the set $L(B(0, 1))$ by c. That is, $Ell(c, L) = c + L(B(0, 1))$[4].

Volume Consider a set $S \subseteq \mathbb{R}^k$. We define the volume of S, denoted $vol(S)$, using the *Lebesgue measure*. The volume of a k-dimensional box $B = \{p \in \mathbb{R}^k : a_1 \leq p_1 \leq b_1, \ldots, a_k \leq p_k \leq b_k\}$ is $vol(B) = \prod_{1 \leq i \leq k}(b_i - a_i)$. Consider a collection of countably many boxes C such that $S \subseteq \bigcup_{B \in C} B$. We define $vol(S) = inf_C\{\sum_{B \in C} vol(B)\}$. An important observation is that the volume of a polytope that is not full-dimensional is 0. Generally, there are sets that are not Lebesque measurable. In this work, however, we only use convex sets, which are measurable.

Size of representation Consider a number $p/q \in \mathbb{Q}$. The *size* of p/q is, intuitively, the number of bits that are needed to represent it. Thus, we define the size of p/q, denoted $size(p/q)$, to be $1 + \lceil log(|p|+1) \rceil + \lceil log(|q|+1) \rceil$. We define the size of an inequality $\sum_{1 \leq i \leq k} a_i \cdot X_i \leq b$ to be $1 + \sum_{1 \leq i \leq k} size(a_i) + size(b)$.

The ellipsoid method In 1979, Khachiyan [16] introduced the first polynomial time algorithm for feasibility of linear programming. In this problem we get as input a polytope $P = \{x \in \mathbb{R}^k : Ax \leq b\}$, where $A \in \mathbb{Q}^{m \times k}$ and $b \in \mathbb{Q}^m$. Our goal is to find a point $p \in P$ or determine that P is empty. Let φ be the size of the maximal inequality that defines P, and let $\nu = 4k^2\varphi$. Also, let $R = 2^\nu$. We sketch the algorithm referred to

[4] In [21], an ellipsoid is defined as follows: $Ell(z, D) = \{p \in \mathbb{R}^k : (p-z)^T \cdot D^-1 \cdot (p-z) \leq 1\}$, where D is a positive definite matrix and $z \in \mathbb{R}^k$. The definition we use is equivalent to this definition.

as the *ellipsoid method* for bounded full-dimensional polytopes. We find a sequence of ellipsoids $\mathcal{E}^0, \mathcal{E}^1, \ldots, \mathcal{E}^N$ of decreasing volumes, such that for every $1 \leq i \leq N$ the ellipsoid $\mathcal{E}^i$ satisfies $P \subseteq \mathcal{E}^i$.

The initial ellipsoid $\mathcal{E}^0$ is the ball with center $0 \in \mathbb{R}^k$ and radius $R \in \mathbb{N}$. Using the radius R ensures that indeed $P \subseteq \mathcal{E}^0$. Assume that we found the ellipsoid $\mathcal{E}^i = Ell(z^i, L^i)$. We describe the $(i+1)$-th iteration of the algorithm. We test if $z^i \in P$. If it is, we are done. Otherwise, we find an inequality that z^i violates. Let $H \subseteq \mathbb{R}^k$ be the half-space that corresponds to the inequality we find. Next, using the half-space H, and the ellipsoid $\mathcal{E}^i$, we generate the ellipsoid $\mathcal{E}^{i+1}$ with the following properties: $\mathcal{E}^i \cap H \subseteq \mathcal{E}^{i+1}$ and $\mathcal{E}^{i+1}$ has a minimal volume. Moreover, we have that $vol(\mathcal{E}^{i+1})/vol(\mathcal{E}^i) \leq 1/e$. For a 2-dimensional example, consider Figure 2. Finally, it is guaranteed that if all the equations generated by the separation oracle are over $\mathbb{Q}$, so are all the points generated by the algorithm.

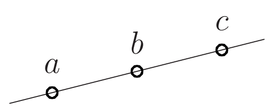

Fig. 1. A 1-dimensional polytope in $\mathbb{R}^2$.

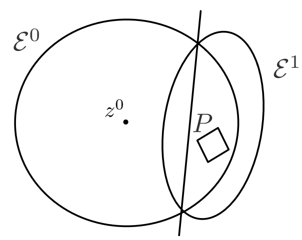

Fig. 2. An illustration of generating the ellipsoid $\mathcal{E}^1$, given that the center z^0 of the ellipsoid $\mathcal{E}^0$ violates an inequality. Note that the polytope P is contained in $\mathcal{E}^0$ and in $\mathcal{E}^1$. Also note that $\mathcal{E}^1$ contains the intersection between $\mathcal{E}^0$ and the half-space corresponding to the inequality.

The termination criterion also depends on φ as above. If P is not empty, then since it is full-dimensional, its volume is at least $2^{-\nu}$, where ν is polynomial in the representation size of φ. Since the volumes of the ellipsoids decrease exponentially, by selecting $N = poly(k, \nu)$, we have that $vol(\mathcal{E}^N) < 2^{-\nu}$. Thus, if $z^N \notin P$, we can conclude that P is empty, and we terminate.

In order to drop the assumption that the polytope is bounded, we use the following property. If the polytope P is not empty, there is a point $p \in P$ with $size(p) \leq 2^{\nu}$. Thus, we can use the polytope that is the intersection of P with the box $\{p \in \mathbb{R}^k : \forall 1 \leq i \leq k,\ 2^{\nu} \leq p_i \leq 2^{\nu}\}$, which is clearly bounded.

Symbolic ellipsoid method Examining Khachiyan's method one can see that it is not necessary to have the implicit list of inequalities that define the given polytope P. Indeed, it was shown in [11,15,20] that it is sufficient to have a *separation oracle* for P when P is full-dimensional. That is, instead of a full list of inequalities that define P, the input to the problem is an oracle that, given a point $p \in \mathbb{R}^k$, either says that $p \in P$ or returns a half-space $H \subseteq \mathbb{R}^k$ such that $p \notin H$ and $P \subseteq H$. Assuming we found the ellipsoid $\mathcal{E}^i = Ell(z^i, L^i)$, we use the oracle to check if z^i is in P. If it is, we are done, and otherwise, we get a half-space with which we construct the ellipsoid $\mathcal{E}^{i+1}$. Since we construct only polynomially many ellipsoids, we perform only polynomial many calls to the oracle. The runtime is thus polynomial in the runtime of the separation oracle, in k, and in the maximal representation size (aka φ) of the inequalities that define P. In Appendix A.2 we explain how we can work with a separation oracle even when the polytope is not full-dimensional. To conclude, we have the following.

Theorem 1. *[21] Consider a polytope $P \subseteq \mathbb{R}^k$, defined by linear inequalities over $\mathbb{Q}$ of size at most φ. Given a separation oracle* `SEP` *for P, it is possible to find a point in $P \cap \mathbb{Q}^k$ in time that is polynomial in k, φ, and the running time of* `SEP`.

3 The PWC Problem for Deterministic WFAs

In this section we show that both the two- and one-bound PWC problems can be solved in polynomial time when the input WFAs are deterministic.

3.1 The Two-Bound PWC Problem for Deterministic Automata

Recall that the input to the two-bound PWC problem are DWFAs $\mathcal{A}$ and $\mathcal{C}$ and an $\mathcal{X}$-parametrized DWFA $\mathcal{B}$. Our goal is to find a *legal assignment* for the variables in $\mathcal{X}$. That is, an assignment f such that $\mathcal{A} \subseteq \mathcal{B}^f \subseteq \mathcal{C}$.

From parameterized containment to a convex polytope Consider an input $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ to the two-bound problem. When the automata are deterministic, checking whether $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C})$ can be done in polynomial time. If Boolean containment does not hold, there is clearly no assignment as required. Thus, we assume that $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C})$.

Consider an assignment $f : \mathcal{X} \to \mathbb{Q}$. By Proposition 2, we have that f is legal iff $\mathcal{S}_{\mathcal{A},\mathcal{B}^f}$ and $\mathcal{S}_{\mathcal{B}^f,\mathcal{C}}$ are negative. Moreover, by Proposition 1, the latter holds iff all the critical paths in $\mathcal{S}_{\mathcal{A},\mathcal{B}^f}$ and $\mathcal{S}_{\mathcal{B}^f,\mathcal{C}}$ have a non-positive value. Thus, the set of critical paths in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ and $\mathcal{S}_{\mathcal{B},\mathcal{C}}$ induce necessary and sufficient restrictions on the possible values the variables can get in a legal assignment. Each critical path induces an inequality over the variables in $\mathcal{X}$, and together all critical paths induce a convex polytope that includes exactly all the legal assignments.

The above observation is the key to our algorithm, and we describe its details below for the product $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. The construction of inequalities induced by $\mathcal{S}_{\mathcal{B},\mathcal{C}}$ is similar. Consider a critical path π in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. We generate an inequality from π that corresponds to a restriction on legal assignment to the variables. Inequalities for $\mathcal{S}_{\mathcal{B},\mathcal{C}}$ are generated in a similar manner. Recall that the path π is a sequence of transitions in a DWFA that is the product of two DWFAs. For every $e = \langle e_\mathcal{A}, e_\mathcal{B} \rangle \in \pi$, let $c_e = \tau_\mathcal{A}(e_\mathcal{A}) - \tau_\mathcal{B}(e_\mathcal{B})$. Recall that $\tau_\mathcal{B}(e_\mathcal{B})$ can either be a number, in which case c_e is a number too, or a variable $X \in \mathcal{X}$, in which case c_e is of the form $c - X$ with $c = \tau_\mathcal{A}(e_\mathcal{A}) \in \mathbb{Q}$. We define the inequality $(\sum_{e \in \pi} c_e) \leq 0$. Clearly, it is possible to rewrite the inequality as $\sum_{1 \leq i \leq k} -l_i \cdot X_i + c \leq 0$, where $l_i \in \mathbb{N}$ is the number of times that $X_i \in \mathcal{X}$ appears in π and $c \in \mathbb{Q}$.

Remark 1. Let $n = max\{|Q_\mathcal{A}| \cdot |Q_\mathcal{B}|, |Q_\mathcal{B}| \cdot |Q_\mathcal{C}|\}$ and $M = max\{|\tau_\mathcal{A}(e_\mathcal{A}) - \tau_\mathcal{B}(e_\mathcal{B})|, |\tau_\mathcal{B}(e_\mathcal{B})|, |\tau_\mathcal{B}(e_\mathcal{B}) - \tau_\mathcal{C}(e_\mathcal{C})| : e_\mathcal{A} \in \Delta_\mathcal{A}, e_\mathcal{B} \in \Delta_\mathcal{B}, e_\mathcal{C} \in \Delta_\mathcal{C}, \tau(e_\mathcal{B}) \in \mathbb{Q}\}$. Since π is either acyclic or a simple cycle, its length is at most n. Since for every $1 \leq i \leq k$, we have that l_i is the number of times $X_i \in \mathcal{X}$ appears in π, then $0 \leq l_i \leq n$. Clearly, $|c| \leq Mn$. Thus, the size of every inequality we generate is at most $1 + \sum_{1 \leq i \leq k} size(n) + size(Mn) = O(log(nM))$.

Let $|\mathcal{X}| = k$. By the above, we think of an assignment to the variables as a point in $\mathbb{R}^k$, think of the inequalities as half-spaces in $\mathbb{R}^k$, and think of the set of legal assignments as a convex polytope in $\mathbb{R}^k$, namely the intersection of all the half-spaces that are generated from the critical paths in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ and $\mathcal{S}_{\mathcal{B},\mathcal{C}}$. We denote this polytope by $\mathcal{P} \subseteq \mathbb{R}^k$.

Efficient reasoning about the convex polytope A naive way to solve the two-bound problem is to generate all the inequalities from $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ and $\mathcal{S}_{\mathcal{B},\mathcal{C}}$ and solve the system of inequalities. Since, however, there can be exponentially many critical paths, the running time of such an algorithm would be at least exponential. In order to overcome this difficulty, we do not construct the induced polytope $\mathcal{P}$ implicitly. Instead, we devise a separation oracle for $\mathcal{P}$. By Theorem 1, this would enable us to find a point in $\mathcal{P} \cap \mathbb{Q}^k$ (or decide that $\mathcal{P}$ is empty) with only polynomial many calls to the oracle.

Recall that a separation oracle for $\mathcal{P}$ is an algorithm that, given a point $p \in \mathbb{R}^k$, either returns that $p \in \mathcal{P}$ or returns a half-space $H \subseteq \mathbb{R}^k$, represented by an inequality, that separates p from $\mathcal{P}$. That is, $\mathcal{P} \subseteq H$ and $p \notin H$.

We describe the separation oracle for $\mathcal{P}$. Given a point $p \in \mathbb{R}^k$, we check if $\mathcal{A} \subseteq \mathcal{B}^p \subseteq \mathcal{C}$. If the latter holds, we conclude that p is a legal assignment, and we are done. Otherwise, there is a word $w \in \Sigma^*$ such that $w \in L(\mathcal{A})$ and $val(\mathcal{A}, w) > val(\mathcal{B}^p, w)$, or $w \in L(\mathcal{B})$ and $val(\mathcal{B}^p, w) > val(\mathcal{C}, w)$. Using w, we find a critical path that p violates and we return the inequality induced by this path. Note that the run on w may not be a critical path: we know it is a path form an initial state to an accepting state, but this path may not be simple. We describe how to detect a critical path from w. Assume that w is such that $val(\mathcal{A}, w) > val(\mathcal{B}^p, w)$. The other case is similar. Since $\mathcal{A}$ and $\mathcal{B}$ are deterministic there is a single accepting run r of $\mathcal{S}_{\mathcal{A},\mathcal{B}^p}$ on w. If r is acyclic, then it is a critical path. Otherwise, we remove every non-positive cycle from r. Let r' be the obtained path in $\mathcal{S}_{\mathcal{A},\mathcal{B}^p}$. Clearly, $val(r') \geq val(r)$. If r' is acyclic, we found a critical path. Otherwise, since $val(r') \geq val(r) > 0$, there must be a positive valued cycle in r'. This cycle is a critical path, and we are done.

We can now use Theorem 1 and conclude with the following.

Theorem 2. *The two-bound PWC problem for DWFAs can be solved in polynomial time.*

Remark 2 (Speeding up the Separation Oracle). Reasoning about critical paths involves a calculation of distances in the graphs corresponding to the product automata and is done by solving the All-Pairs Shortest Path problem. As we update the ellipsoids, we also update costs in the product automata. There is much research in the field of *dynamic graph algorithms* (specifically, [24] suggests a fully-dynamic data-structure to solve the All-Pairs Shortest Path problem) that we can use here in order to speed up the running time of the separation oracle so that the time required for solving a distance query is proportional to the updates rather than to the automata.

3.2 When $\mathcal{B}$ is Not Given

An interesting variant of the two-bound PWC problem is one in which we are not given $\mathcal{B}$ and we seek a DWFA of a minimal size such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. One may start the search for $\mathcal{B}$ with a non-weighted version of the problem. That is, seek a minimal DFA $\mathcal{B}$ such that $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C})$. We can then turn $\mathcal{B}$ into a candidate DWFA by labeling all its transitions by variables. The corresponding decision problem, which we refer to as the *Boolean sandwich* problem, gets as input two DFAs $\mathcal{A}$ and $\mathcal{C}$ and index $k \in \mathbb{N}$ and decides whether there is a DFA $\mathcal{B}$ with k states such that $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C})$. In the *weighted sandwich* problem, the language of $\mathcal{B}$ is given by means of a DFA. The corresponding decision problem gets as input DWFAs $\mathcal{A}$ and $\mathcal{C}$, a DFA $\hat{\mathcal{B}}$ and an index $k \in \mathbb{N}$, and decides whether there is a DWFA $\mathcal{B}$ with $L(\mathcal{B}) = L(\hat{\mathcal{B}})$ such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. As we show now, however, both problems are difficult.

Theorem 3. *The Boolean-sandwich and the weighted-sandwich problems are NP-complete.*

Proof: Since the automata are deterministic, checking whether a given k-state automaton satisfies the Boolean or weighted sandwich requirements can be done in polynomial time. Thus, membership in NP in easy.

For the lower bound, we show a reduction from the vertex coloring problem (VC, for short). Recall that the input to the VC problem is $\langle G, k \rangle$, where G is a graph and $k \in \mathbb{N}$ is an index. We say that $\langle G, k \rangle \in$ VC iff there is a coloring of G's vertices in k colors such that two adjacent vertices are not colored in the same color.

In the Boolean case, consider an input $\langle G, k \rangle$ to the VC problem. We construct an input $\langle \mathcal{A}, \mathcal{C}, k+2 \rangle$ to the Boolean-sandwich problem. The idea is that $\mathcal{A}$ has a state representing every vertex in G. In order to construct $\mathcal{B}$ one must, intuitively, *merge* different states of $\mathcal{A}$. The automaton $\mathcal{C}$ enforces that merging two states can only be done if the corresponding vertices do not share a common edge, and thus the states of $\mathcal{B}$ correspond to a legal coloring of the vertices of G. For the details of the proof, see Appendix A.3.

In the weighted sandwich problem, we go through the the t-approximation problem for DWFAs, which is defined as follows: given a DWFA $\mathcal{A}$, a factor $t \in \mathbb{Q}$ such that $t > 1$, and an index $k \in \mathbb{N}$, we ask whether there is a DWFA $\mathcal{A}'$ with k states such that $L(\mathcal{A}) = L(\mathcal{A}')$ and for every word $w \in L(\mathcal{A})$, we have $1/t \cdot val(\mathcal{A}, w) \leq val(\mathcal{A}', w) \leq t \cdot val(\mathcal{A}, w)$. We say that $\mathcal{A}'$ t-approximates $\mathcal{A}$. Similar to the Boolean case above, given an input $\langle G, k \rangle$ to the VC problem, we construct a DWFA $\mathcal{A}$ with a state corresponding to every vertex in G. Constructing an approximating automaton for $\mathcal{A}$ is done by merging states. We construct $\mathcal{A}$ so that by merging two states whose corresponding vertices share an edge, there is a word in $L(\mathcal{A})$ that violates the t-approximation requirement. Thus, an approximating automaton for $\mathcal{A}$ corresponds to a legal coloring of G. For the details of the proof, see Appendix A.4. $\square$

We suggest a heuristic to cope with the complexity of the sandwich problems. Consider DWFAs $\mathcal{A}$ and $\mathcal{C}$ such that $L(\mathcal{A}) \subseteq L(\mathcal{C})$. Let $Q_{\mathcal{A}}$ and $Q_{\mathcal{C}}$ be the state spaces of $\mathcal{A}$ and $\mathcal{C}$, respectively. We start the search for $\mathcal{B}$ with a DFA $\mathcal{D}$ that has the state space $Q_{\mathcal{A}} \times Q_{\mathcal{C}}$. All the states in $F_{\mathcal{A}} \times Q_{\mathcal{C}}$ are accepting in $\mathcal{D}$, and we have a freedom to chose which states in $Q_{\mathcal{A}} \times F_{\mathcal{C}}$ are also accepting. With each such choice, we have $L(\mathcal{A}) \subseteq L(\mathcal{D}) \subseteq L(\mathcal{C})$. Thus, our goal is to decide which states should be defined as accepting, and also decide whether and how $\mathcal{D}$ can be minimized. Suppose we choose a set of accepting states. We iteratively search for states that can be merged in $\mathcal{D}$. The candidates for merging are the states that are merged in the regular DFA minimization algorithm. Consider two candidate states $q, q' \in Q_{\mathcal{D}}$. We construct a DFA $\mathcal{D}'$ by merging q and q'. Then, we check, using the algorithm of Theorem 2, if it is possible to find a DWFA $\mathcal{B}$ on the structure of $\mathcal{D}'$ that satisfies $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. We do this by setting a different variable for every transition in $\mathcal{D}'$. If it is, we continue to find new candidates for merging in $\mathcal{D}'$. Otherwise, we un-merge the states, and look for new candidates in $\mathcal{D}$. We may also modify our choice of accepting states.

3.3 The One-Bound PWC Problem for Deterministic Automata

Recall that the input to the one-bound PWC problem is a DWFA $\mathcal{A}$ and an $\mathcal{X}$-parameterized automaton $\mathcal{B}$, which we want to complete to a DWFW that either upper bounds $\mathcal{A}$ in a minimal way or lower bounds $\mathcal{A}$ in a maximal way. We focus here on the case we seek a least upper bound. The second case is similar. As in the two-bound case, we say that an assignment f is legal if it satisfies $\mathcal{A} \subseteq \mathcal{B}^f$.

As detailed below, it is technically simpler to assume that all the states in the DWFAs are accepting. Thus, in this model, the language of a DWFA is Σ^*. We can, however, use weights and encode rejecting states. For example, we can add to the alphabet a letter $\#$ that leaves all states to some state with either a bottom value, when we do not want the origin state to be considered accepting, or with value 0 when we want it to be accepting. We then restrict attention to prefixes of words that end after the first $\#$.

As in the two-bound problem, we view assignments as points in $\mathbb{R}^k$ and use inequalities induced by critical paths in order to define a polytope $\mathcal{P} \subseteq \mathbb{R}^k$ of legal assignments. We show that the polytope generated in this case is in full-dimensional. Intuitively, it follows from the fact that increasing a point by ϵ results in a point that is still in $\mathcal{P}$.

Lemma 1. *If $\mathcal{P} \neq \emptyset$, then $\mathcal{P}$ is full-dimensional.*

Unlike the two-bound case, here we are not looking for an arbitrary point in $\mathcal{P}$, but one that is a minimal assignment. We show that a vertex of $\mathcal{P}$ is such an assignment. Intuitively, it follows from the fact that points on a face F of $\mathcal{P}$ are minimal assignments with respect to the variables participating in the inequality corresponding to F. A vertex is the intersection of k faces, and thus, it corresponds to an assignment that is minimal with respect to all faces and hence also with respect to all variables.

Lemma 2. *A vertex in $\mathcal{P}$ is a minimal assignment.*

Recall that some of the inequalities that define $\mathcal{P}$ are induced by critical paths that are simple paths from the initial vertex to an accepting state. Since we assume that all states are accepting, prefixes of such critical paths are also critical. From the geometrical point of view, this implies the following.

Lemma 3. *If $\mathcal{P} \neq \emptyset$, then $\mathcal{P}$ is pointed.*

For full-dimensional pointed polytopes, Schrijver shows a strengthening of Theorem 1 that enables us to find vertices:

Theorem 4. *[21] Consider a full-dimensional pointed polytope $P \subseteq \mathbb{R}^k$, defined by linear inequalities over $\mathbb{Q}$ of size at most φ. Given a separation oracle* `SEP` *for P, it is possible to find a vertex of P in $\mathbb{Q}^k$, in time that is polynomial in k, φ, and the running time of* `SEP`.

By the lemmas above, Theorem 4 is applicable in the one-bound PWC problem and we conclude with the following.

Theorem 5. *The one-bounded problem can be solved in polynomial time.*

4 The PWC Problem for WFAs

In this section we study the one- and two-bound problems for WFAs. Recall that containment for WFAs is undecidable, making the decidability of the PWC hopeless. Consequently, we replace the containment order for WFAs by *weighted simulation* [3,7]. Simulation has been extensively used in order to approximate containment in the Boolean setting, and was recently used as a decidable approximation of containment in the weighted setting.

Let us explain the idea behind weighted simulation. Given two WFAs $\mathcal{A}$ and $\mathcal{B}$, deciding whether $\mathcal{A} \subseteq \mathcal{B}$ can be thought of as a two-player game of one round: Player 1, the Player whose goal it is to show that there is no containment, chooses a word w and a run r_1 of $\mathcal{A}$ on w. Player 2 then replies by choosing a run r_2 of $\mathcal{B}$ on w. Player 1 wins if r_1 is accepting and r_2 is not or if $val(r_1, w) > val(r_2, w)$. While this game clearly captures containment, it does not lead to interesting insights or algorithmic ideas about checking containment. A useful way to view simulation is as a "step-wise" version of the above game in which in each round the players proceed according to a single transition of the WFAs. More formally, $\mathcal{B}$ simulates $\mathcal{A}$, denoted $\mathcal{A} \leq \mathcal{B}$, if Player 2 has a strategy that wins against all strategies of Player 1: no matter how Player 1 proceeds in the WFA $\mathcal{A}$, Player 2 can respond in a transition so that whenever the run generated so far in $\mathcal{A}$ by Player 1 is accepting, so is the run generated by Player 2 in $\mathcal{B}$. Moreover, the cost of the run in $\mathcal{A}$ is smaller than the one in $\mathcal{B}$. For full details, see [3].

So, in the nondeterministic setting, we replace containment with simulation and seek, in the two-bound case, a valuation f such that $\mathcal{A} \leq \mathcal{B}^f \leq \mathcal{C}$, and in the one bound cases minimal and maximal assignments so that $\mathcal{A} \leq \mathcal{B}^f$ or $\mathcal{B}^f \leq \mathcal{C}$, respectively.

4.1 The One-Bound PWC Problem for Nondeterministic Automata

We argue that this version of the PWC problem is not very interesting, in fact it is not well defined as is, as there are cases in which we do not have even a minimal (or maximal) assignment.

Consider for example the WFAs in Figure 4.1. The candidates for minimal assignments for the variables in $\mathcal{B}$ are the ones that assign the value 0 to X_1 or X_2. However, an assignment f with $f(X_1) = 0$ can assign to X_2 an arbitrarily low value, and, symmetrically, an assignment with $f(X_2) = 0$ can assign an arbitrarily low value to X_1. Hence, there is no minimal assignment for the variables in $\mathcal{B}$.

Thus, the nondeterministic setting calls for a different definition of the one-bound PWC problem – one that considers alternative sets of variables whose values should be minimized. We do not find such definitions well motivated.[5]

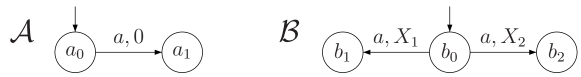

Fig. 3. An example in which there is no minimal assignment.

4.2 The Two-Bound PWC Problem for Nondeterministic Automata

We now turn to study the two-bound problem. As we shall show, here the set of legal assignments corresponds to a vertex polytope, so it is either empty or not and the problem is well defined. On the other hand, the complexity of the problem depends on the complexity of deciding weighted simulation, and is NP-complete even if one finds a polynomial algorithm for deciding weighted simulation (the best known algorithm for weighted simulation positions it in NP $\cap$ co-NP). We first show that the problem is NP-hard.

Theorem 6. *The two-bound PWC problem for WFAs is NP-hard.*

Proof: We prove that finding a satisfying assignment to a 3-SAT formula is easier than solving the two-bound PWC problem for WFAs. Consider an input formula ψ to the 3-SAT problem. The intuition behind the construction is as follows. For every variable in ψ there are two variables in $\mathcal{X}$. We construct $\mathcal{B}$ and $\mathcal{C}$, so that the simulation game corresponding to them guarantees that only one of the two variables in $\mathcal{X}$ that correspond to a variable in ψ can get a value greater than or equal to 1. Thus, an assignment to the variables in $\mathcal{X}$ corresponds to an assignment to the variables in ψ. The idea of the game that corresponds to $\mathcal{A}$ and $\mathcal{B}$ is to force the assignment to variables in ψ to be satisfying. That is, Player 1 challenges Player 2 with a clause. Player 2 in his turn chooses a literal that is satisfied in this clause. For the full proof see Appendix A.8. $\square$

It is shown by Schrijver in [21] that there is a connection between the size of the inequalities that define a polytope P and its vertices.

Theorem 7. *[21] Let $P \subseteq \mathbb{R}^k$ be a polytope that is defined by inequalities of size at most φ. Then, the size of each of its vertices is at most $4k^2\varphi$.*

We show that Theorem 7 implies the following upper bound:

Theorem 8. *A polynomial algorithm for solving simulation games implies that solving the two-bound nondeterministic PWC is in NP.*

[5] Having said that, the setting does suggest some very interesting theoretical problems, like deciding when a solution exists, and the relation between the observation above and the fact WFAs cannot always be determined.

Proof: Consider an assignment $f : \mathcal{X} \to \mathbb{Q}$. Since we assume that solving simulation games can be done in polynomial time, deciding whether $\mathcal{A} \leq \mathcal{B}^f \leq \mathcal{C}$, can also be done in polynomial time. Thus, we need to show that if there is a legal assignment, there is also one of polynomial size. Consider a legal assignment f. Thus, Player 2 wins both the simulation game that corresponds to $\mathcal{A}$ and $\mathcal{B}^f$, and the game that corresponds to $\mathcal{B}^f$ and $\mathcal{C}$. We show in [3] that this implies that Player 2 has a *memoryless winning strategy* in the two games. We show that similar to Section 3, it is possible to represent the set of assignments for which these strategies are winning, as a k-dimensional polytope that is defined by inequalities of polynomial size. Thus, Theorem 7 implies that there is a point with polynomial size in this polytope, which corresponds to a legal assignment with polynomial size. For the full proof see Appendix A.9 $\square$

Acknowledgments We thank Nati Linial for the helpful discussions and pointers.

References

1. S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *Proc 9th ATVA*, LNCS 6996, pages 482–491, 2011.
2. R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proc. 25th STOC*, pages 592–601, 1993.
3. G. Avni and O. Kupferman. Making weighted containment feasible: A heuristic based on simulation and abstraction. In *Proc. 23rd CONCUR*, pages 84–99, 2012.
4. G. Bruns and P. Godefroid. Temporal logic query checking. In *Proc. 16th LICS*, pages 409–420, 2001.
5. W. Chan. Temporal-logic queries. In *Proc. 12th CAV*, LNCS 1855 , pages 450–463, 2000.
6. K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. In *Proc. 17th CSL*, pages 385–400, 2008.
7. K. Chatterjee, L. Doyen, and T. A. Henzinger. Expressiveness and closure properties for quantitative languages. *LMCS*, 6(3), 2010.
8. K. Culik and J. Kari. Digital images and formal languages. *Handbook of formal languages, vol. 3: beyond words*, pages 599–616, 1997.
9. E.W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
10. L. Fix, N. Francez, and O. Grumberg. Program composition and modular verification. In *Proc. 18th ICALP*, pages 93–114, 1991.
11. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
12. M. Grötschel, L. Lovász, and A. Schrijver. Corrigendum to our paper "the ellipsoid method and its consequences in combinatorial optimization". *Combinatorica*, 4(4), 1984.
13. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York, 1988.
14. B. Jobstmann, A. Griesmayer, and R. Bloem. Program repair as a game. In *Proc. 17th CAV*, LNCS 3576, pages 226–238, 2005.
15. R. Karp and C. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *Proc. 21st FOCS*, pages 1–9, 1980.
16. L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.
17. D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–425, 1994.

18. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
19. 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.
20. M.W. Padberg and M. R. Rao. *The Russian Method and Integer Programming*. Working paper series. Salomon Brothers Center for the Study of Financial Institutions, 1980.
21. A. Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.
22. A. Solar-Lezama, R.M. Rabbah, R. Bodík, and K. Ebcioglu. Programming by sketching for bit-streaming programs. In *PLDI*, pages 281–294, 2005.
23. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
24. M. Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In *SWAT*, pages 384–396, 2004.
25. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I& C*, 115(1):1–37, 1994.

A Elaborations and Proofs

A.1 Geometry in $\mathbb{R}^k$

Vertices Consider a polytope $P = \{x \in \mathbb{R}^k : Ax \leq b\}$. A *face* of P is a set $\emptyset \neq F \subseteq P$ such that $F = \{x \in P : A'x = b'\}$, where $A'x \leq b'$ is a subsystem of $Ax \leq b$. A *minimal face* of P is a face which does not contain any other face. Thus, it is minimal with respect to inclusion. All the minimal faces of P have the same dimension, which is k minus the rank of A. For example, consider again the polytope in Figure 1. The polytope has only one face, which is also a minimal face. When the rank of A is k, a minimal face contains only one point, which is referred to as a *vertex*.

A.2 Using Separation Oracle in Non-Full-Dimensional Polytopes

In order to drop the requirement that the polytope is full-dimensional, Khachiyan suggested to work with the polytope $P^\epsilon = \{x \in \mathbb{R}^k : Ax \leq b + \epsilon\}$. He shows that there is an $\epsilon > 0$ such that P is empty iff P^ϵ is empty, and if P is non-empty, then P^ϵ is full-dimensional. Using this technique is not enough when the polytope is represented with a separation oracle. The reason is that finding a point in P^ϵ does not help us find a point in P.

The solution to this problem in the non-full-dimensional case is due to [12,13,15]. We sketch the algorithm, and its details can be found in Chapter 14 of [21] or Chapter 6 of [13]. We start as in the ellipsoid method: we find a sequence of ellipsoids of diminishing volume. If the polytope is full-dimensional, then, as in the ellipsoid method, there is a lower bound on the volume of the polytope. Thus, there is a bound on the length of the sequence. However, when the polytope is not full-dimensional, since the k-dimensional volume of the polytope is 0, the sequence of ellipsoids may not terminate. To overcome this difficulty, we take advantage of the fact that the ellipsoids get "flatter" in the direction of the hyper-plane that P is contained in. We use a tool dubbed *simultaneous diophantine approximation* to find the representation of the containing hyper-plane. Then, after finding it, we construct a new separation oracle for a polytope in $k-1$ dimensions, and continue recursively.

A.3 Proof of the Lower Bound for Boolean-Sandwiches

In order to show that Boolean sandwich problem is NP-hard, we show a reduction from VC. Recall that the input to the VC problem is $\langle G, k\rangle$, where $G = \langle V, E\rangle$ is a graph, $V = \{v_1, \ldots, v_n\}$ is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $k \in \mathbb{N}$ is an index. We say that $\langle G, k\rangle \in$ VC iff there is a coloring function $\chi : V \to \{1, \ldots, k\}$ such that two adjacent vertices are not colored in the same color. That is, for every $\langle v, v'\rangle \in E$, we have $\chi(v) \neq \chi(v')$. We assume that there are no self-loops and at least two vertices in G.

Consider an input $\langle G, k\rangle$ to the coloring problem. We construct two DFAs $\mathcal{A}$ and $\mathcal{C}$ such that there is a DFA $\mathcal{B}$ with $k+2$ states that satisfies $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$ iff $\langle G, k\rangle \in$ VC. We describe the intuition of the construction. The alphabet of the automata are the vertices V. The languages of $\mathcal{A}$ and $\mathcal{C}$ consist of words with two letters. The language of $\mathcal{A}$ is vv for every $v \in V$. For $v, v' \in V$, the word vv' is in the language of $\mathcal{C}$ iff $\langle v, v'\rangle \notin E$.

For ease of presentation, we assume that a word is accepted by a DFA iff the automaton has an accepting run on it. Thus, a state might not have an outgoing transition for every letter in the alphabet.

Formally, we define $\mathcal{A} = \langle \Sigma, Q_{\mathcal{A}}, q_{0_{\mathcal{A}}}, \delta_{\mathcal{A}}, F_{\mathcal{A}} \rangle$, where the alphabet Σ is V, the states $Q_{\mathcal{A}}$ are $\{q_{0_{\mathcal{A}}}, q_{acc}\} \cup V$, the initial state is $q_{0_{\mathcal{A}}}$, for every $v \in V$ we define the transition function $\delta_{\mathcal{A}} : Q_{\mathcal{A}} \times \Sigma \to Q_{\mathcal{A}}$ to be $\delta_{\mathcal{A}}(q_{0_{\mathcal{A}}}, v) = v$ and $\delta_{\mathcal{A}}(v, v) = q_{acc}$, and the set of accepting states $F_{\mathcal{A}}$ is the singleton $\{q_{acc}\}$. Clearly, a word $w \in \Sigma^*$ is in $L(\mathcal{A})$ iff $w = vv$ for some $v \in V$.

We define $\mathcal{C} = \langle \Sigma, Q_{\mathcal{C}}, q_{0_{\mathcal{C}}}, \delta_{\mathcal{C}}, F_{\mathcal{C}} \rangle$, where the states $Q_{\mathcal{C}}$ are $\{q_{0_{\mathcal{C}}}, q_{acc}\} \cup V$, the initial state is $q_{0_{\mathcal{C}}}$, for every $v, v' \in V$ we define the transition function $\delta_{\mathcal{C}} : Q_{\mathcal{C}} \times \Sigma \to Q_{\mathcal{C}}$ to be $\delta_{\mathcal{C}}(q_{0_{\mathcal{C}}}, v) = v$ and $\delta_{\mathcal{C}}(v, v') = q_{acc}$ iff $\langle v, v' \rangle \notin E$, and the set of accepting states $F_{\mathcal{C}}$ is the singleton $\{q_{acc}\}$. Clearly, a word $w \in \Sigma^*$ is in $L(\mathcal{C})$ iff it is of the form vv' and $\langle v, v' \rangle \notin E$. Note that since there are no self loops, for every $v \in V$ we have $vv \in L(\mathcal{C})$, and thus $\mathcal{A} \subseteq \mathcal{C}$. Clearly, the reduction is polynomial in the input.

We continue to prove the correctness of the reduction. If $\langle G, k \rangle \in$ VC, there is a coloring function $\chi : V \to \{1, \ldots, k\}$. We construct the DFA $\mathcal{B} = \langle \Sigma, \{q_{0_{\mathcal{B}}}, q_{acc}, 1 \ldots, k\}, q_{0_{\mathcal{B}}}, \delta_{\mathcal{B}}, \{q_{acc}\} \rangle$, where for every $v \in V$ we define $\delta_{\mathcal{B}}(q_{0_{\mathcal{B}}}, v) = \chi(v)$ and $\delta_{\mathcal{B}}(\chi(v), v) = q_{acc}$.

We claim that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. Consider a word $vv \in L(\mathcal{A})$. It is easy to see that there is an accepting run of $\mathcal{B}$ on vv, and thus $vv \in L(\mathcal{B})$ and $\mathcal{A} \subseteq \mathcal{B}$. We continue to prove that $\mathcal{B} \subseteq \mathcal{C}$. Clearly, $\mathcal{B}$ accepts only words with two letters. Consider a word $vv' \in L(\mathcal{B})$. We claim that $vv' \in L(\mathcal{C})$. That is, we prove that $\langle v, v' \rangle \notin E$. Since there is a transition from the state $\chi(v)$ to q_{acc} labeled v', we have $\chi(v) = \chi(v')$. Since χ is a legal coloring function, if $\chi(v) = \chi(v')$, then $\langle v, v' \rangle \notin E$, and we are done.

For the other direction, assume that there is a DFA $\mathcal{B}$ with $k+2$ states, such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. We construct a coloring function $\chi : V \to \{1, \ldots, k\}$ for G. Recall that the alphabet of $\mathcal{B}$ is V. We name the states of $\mathcal{B}$ with the numbers $0, 1, \ldots, k+1$, where 0 is the initial state and $k+1$ is accepting. Note that $\mathcal{B}$ must have at least one accepting state since $L(\mathcal{A}) \neq \emptyset$ implies that $L(\mathcal{B}) \neq \emptyset$. For $v \in V$, we define $\chi(v) = \delta_{\mathcal{B}}(0, v)$.

We claim that this is a legal coloring function with k colors. We claim that there is no $v \in V$ with $\chi(v) = k+1$ or $\chi(v) = 0$. Assume by way of contradiction that there is such a v. We distinguish between two cases. If $\chi(v) = k+1$, then by the definition of χ we have $\delta_{\mathcal{B}}(0, v) = k+1$. Since $k+1$ is an accepting state, we have $v \in L(\mathcal{B})$. However, since the word v has only one letter we have $v \notin L(\mathcal{C})$, and thus $L(\mathcal{B}) \not\subseteq L(\mathcal{C})$, which is a contradiction to our assumption. For the other case, we assume $\chi(v) = \delta_{\mathcal{B}}(0, v) = 0$. Consider a vertex $v' \neq v$. Since $v'v' \in L(\mathcal{A})$, we have $v'v' \in L(\mathcal{B})$. Since $\delta_{\mathcal{B}}(0, v) = 0$, we have $vv'v' \in L(\mathcal{B})$. Since the word $vv'v'$ has three letters, we have $vv'v' \notin L(\mathcal{C})$, and we reach a contradiction to the fact that $\mathcal{B} \subseteq \mathcal{C}$.

Next, we claim that for every $v, v' \in V$, if $\langle v, v' \rangle \in E$, then $\chi(v) \neq \chi(v')$. Assume by way of contradiction that there is an edge $\langle v, v' \rangle \in E$ and $\chi(v) = \chi(v')$. Recall that $\chi(v) = \delta_{\mathcal{B}}(0, v) = \delta_{\mathcal{B}}(0, v') = \chi(v')$. Since $vv \in L(\mathcal{A})$ we have that $\delta_{\mathcal{B}}(\chi(v), v)$ is accepting. In other words, $\delta_{\mathcal{B}}(\chi(v'), v)$ is accepting. Thus, the word $vv' \in L(\mathcal{B})$. However, $\langle v, v' \rangle \notin L(\mathcal{C})$, and thus we reach a contradiction to the fact that $\mathcal{B} \subseteq \mathcal{C}$, and we are done.

A.4 Weighted Sandwich and t-approximation of DWFAs

For two DWFAs $\mathcal{A}$ and $\mathcal{A}'$, and a factor number $t \in \mathbb{Q}$ such that $t > 1$, we say that $\mathcal{A}'$ t-approximates $\mathcal{A}$ iff their language is the same, i.e., $L(\mathcal{A}) = L(\mathcal{A}')$, and the weight the automata assign to every word in their language is within the factor t. That is, for every word $w \in L(\mathcal{A})$ it holds that $1/t \cdot val(\mathcal{A}, w) \leq val(\mathcal{A}', w) \leq t \cdot val(\mathcal{A}, w)$.

Formally, we define the t-approximation decision problem as follows:

Definition 2. *$\langle \mathcal{A}, k\rangle \in$ MIN-t-APPROX iff there exists a deterministic weighted automaton $\mathcal{A}'$ with k states such that $\mathcal{A}'$ t-approximates $\mathcal{A}$.*

We show that the weighted-sandwich problem can be reduced to the t-approximation problem. Consider an input $\langle \mathcal{A}, t, k\rangle$ to the t-approximation problem. We construct an input $\langle \mathcal{A}', \mathcal{C}', \hat{\mathcal{B}}, k\rangle$ to the weighted sandwich problem. The DWFA $\mathcal{A}'$ is the same as $\mathcal{A}$, only that the weight function is multiplied by $1/t$. Thus, we have $L(\mathcal{A}') = L(\mathcal{A})$ and for every word $w \in \Sigma^*$, we have $val(\mathcal{A}', w) = 1/t \cdot val(\mathcal{A}, w)$. The DWFA $\mathcal{C}'$ is the same as $\mathcal{A}$, only that the weight function is multiplied by t. Thus, we have $L(\mathcal{C}') = L(\mathcal{A})$ and for every word $w \in \Sigma^*$, we have $val(\mathcal{C}', w) = t \cdot val(\mathcal{A}, w)$. Finally, $\hat{B} = \mathcal{A}$. Clearly, a t-approximating automaton for $\mathcal{A}$ meets the weighted sandwich condition.

We continue to prove the following theorem:

Theorem 9. *MIN-t-APPROX is NP-complete.*

Proof: The upper bound is easy: Consider the following two DWFAs $1/t \cdot \mathcal{A}$ and $t \cdot \mathcal{A}$. For every word $w \in \Sigma^*$, the first has $val(1/t \cdot \mathcal{A}, w) = 1/t \cdot val(\mathcal{A}, w)$, and the second has $val(t \cdot \mathcal{A}, w) = t \cdot val(\mathcal{A}, w)$. Given a DWFA $\mathcal{A}'$ with k states, it is possible to check, in polynomial time, whether $1/t \cdot \mathcal{A} \subseteq \mathcal{A} \subseteq t \cdot \mathcal{A}$.

For the lower bound, we show a reduction from VC. Assume that $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We assume that there is some relation $<$ on the set of vertices.

Consider an input $\langle G, k\rangle$ to the VC problem, where $G = \langle V, E\rangle$. We construct an input $\langle \mathcal{A}, k + 2\rangle$ to the MIN-t-APPROX problem. The alphabet of $\mathcal{A}$ is $V \cup E$. The language of $\mathcal{A}$ has two types of words. One letter words, which consist of a vertex, and two letter words, which consist of a vertex and an edge. That is, $L(\mathcal{A}) = V \cup \{v \cdot e : v \in V, e \in E\}$. For every word $v \in V$, the value $\mathcal{A}$ assigns to v is 0. Consider a word $v \cdot e$, where $v \in V$ and $e = \langle v_1, v_2\rangle \in E$, where $v_1 < v_2$. There are three possible values for $v \cdot e$. If $v \neq v_1$ and $v \neq v_2$, then $val(\mathcal{A}, v \cdot e) = t$. Otherwise, if $v = v_1$, then $val(\mathcal{A}, v \cdot e) = 1$ and otherwise $v = v_2$ and $val(\mathcal{A}, v \cdot e) = t^2 + 1$.

Intuitively, in $\mathcal{A}$, there is a state that corresponds to every vertex. Coloring two vertices in the same color corresponds to merging their states. Consider two vertices $v, v' \in V$ that share a common edge $e = \langle v, v'\rangle \in E$ and $v < v'$. Recall that $val(\mathcal{A}, v \cdot e) = 1$ and $val(\mathcal{A}, v' \cdot e) = t^2 + 1$. We show that in an approximating automaton $\mathcal{A}'$ for $\mathcal{A}$, the states that correspond to v and v' cannot be merged. Intuitively, this is because merging them forces $\mathcal{A}'$ to assign to the words $v \cdot e$ and $v' \cdot e$, the same value. Since there is no value $1 \leq x \leq t^2 + 1$ such that $1/t \cdot (t^2 + 1) \leq x \leq 1 \cdot t$, we have a violation of the t-approximation requirement. Thus, we show that $\mathcal{A}$ has a t-approximating automaton with $k + 2$ states iff there is a coloring of G with k colors.

We continue to define the reduction formally. We construct a DWFA $\mathcal{A} = \langle \Sigma_{\mathcal{A}}, Q_{\mathcal{A}}, \Delta_{\mathcal{A}}, q_{in}, F_{\mathcal{A}}, \tau_{\mathcal{A}}\rangle$, where $\Sigma_{\mathcal{A}}$, $Q_{\mathcal{A}}$, $\Delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma_{\mathcal{A}} \times Q_{\mathcal{A}}$, $F_{\mathcal{A}} \subseteq Q_{\mathcal{A}}$, and $\tau_{\mathcal{A}} : \Delta_{\mathcal{A}} \to \mathbb{N}$ are defined as follows:

– $\Sigma_{\mathcal{A}} = V \cup E$.
– $Q_{\mathcal{A}} = \{q_{in}, q_{acc}\} \cup V$.
– For $v \in V$ and $e \in E$ we define $\Delta_{\mathcal{A}}(q_{in}, v, v)$ and $\Delta_{\mathcal{A}}(v, e, q_{acc})$.
– $F_{\mathcal{A}} = \{q_{acc}\} \cup V$.
– We define $\tau_{\mathcal{A}}(\langle q_{in}, v, v\rangle) = 0$. We assume that there is some order $<$ on the vertices. For every edge $e = \{v_1, v_2\}$, where $v_1 < v_2$ we define $\tau_{\mathcal{A}}(\langle v_1, e, q_{acc}\rangle) = 1$, $\tau_{\mathcal{A}}(\langle v_2, e, q_{acc}\rangle) = t^2+1$. For every vertex $v \neq v_1, v_2$ we define $\tau_{\mathcal{A}}(\langle v, e, q_{acc}\rangle) = t$.

The input to the MIN-t-APPROX problem is $\langle \mathcal{A}, k+2\rangle$.

Example 3. The graph on the left of Figure 4 is the input to the VC problem. We construct the automaton on the right as the input to the MIN-t-APPROX problem, where we set $t = 2$. We do not draw all the labels and weights out of lack of space. Since $e_1 = \langle v_1, v_2\rangle \in E$ and $v_1 < v_2$, we have $\tau_{\mathcal{A}}(\langle v_1, e_1, q_{acc}\rangle) = 1$. Since $e_4 = \langle v_3, v_4\rangle \in E$ and $v_3 < v_4$, we have $\tau_{\mathcal{A}}(\langle v_4, e_4, q_{acc}\rangle) = 5$. Since v_4 is not in $e_3 = \langle v_1, v_3\rangle$, then $\tau_{\mathcal{A}}(\langle v_4, e_3, q_{acc}\rangle) = 2$.

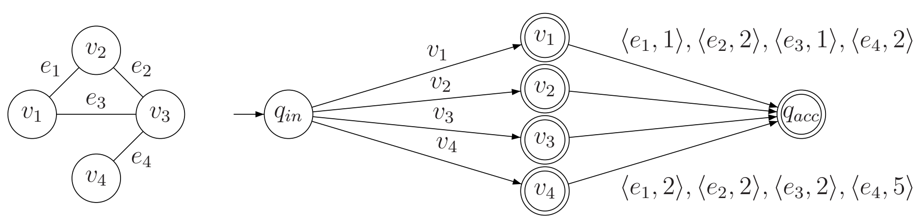

Fig. 4. On the left, we have the input graph. The reduction for a ratio of 2 outputs the automaton $\mathcal{A}$ on the right.

We continue to prove the correctness of the reduction. That is, we prove that $\langle G, k\rangle \in$ VC iff $\langle \mathcal{A}, k+2\rangle \in$ MIN-t-APPROX. For the first direction assume that $\langle G, k\rangle \in$ VC. Thus, there is a coloring function $\chi : V \to C$, where $|C| = k$. We construct an automaton $\mathcal{A}'$ that t-approximates $\mathcal{A}$ by, intuitively, merging states in $\mathcal{A}$ that correspond to vertices that are colored in the same color. Formally, the states of $\mathcal{A}'$ are $\{q_{in}, q_{acc}\} \cup C$. The structure of $\mathcal{A}'$ is the same as $\mathcal{A}$. That is, the edges are either of the form $\{q_{in}\} \times V \times C$ or $C \times E \times \{q_{acc}\}$, where there is a transition $\langle q_{in}, v, c\rangle$ iff v is colored in the color c, i.e., $\chi(v) = c$. For every color $c \in C$ and every $e \in E$, there is a transition $\langle c, e, q_{acc}\rangle \in \Delta_{\mathcal{A}'}$. Clearly, the automaton is deterministic. The accepting states are $C \cup \{q_{acc}\}$.

We continue to define the weight function of $\mathcal{A}'$. The weight of the transitions $\{q_{in}\} \times E \times C$ remains 0 in $\mathcal{A}'$. We define the weight of the edges of the form $C \times E \times \{q_{acc}\}$. For every state $c \in C$ and letter $e \in E$, we define the weight of every transition $\langle c, e, q_{acc}\rangle$ as follows:

– If there exists a vertex $v \in V$ with $\chi(v) = c$ such that $\tau_{\mathcal{A}}(\langle v, e, q_{acc}\rangle) = 1$, then $\tau_{\mathcal{A}'}(\langle c, e, q_{acc}\rangle) = t$.
– Otherwise, we define $\tau_{\mathcal{A}'}(\langle c, e, q_{acc}\rangle) = t + 1$.

We claim that $\mathcal{A}'$ t-approximates $\mathcal{A}$. First, we prove that $L(\mathcal{A}) = L(\mathcal{A}')$. Recall that $L(\mathcal{A}) = V \cup \{v \cdot e : v \in V, e \in E\}$. Consider a vertex $v \in V$. Since, there is a color $c \in C$ such that $\chi(v) = c$, there is a run of $\mathcal{A}'$ on v that ends in c, which is accepting. Thus, $v \in L(\mathcal{A}')$. Consider a vertex $v \in V$ and an edge $e \in E$. Since for every $c \in C$ there is a transition $\langle c, e, q_{acc}\rangle \in \Delta_{\mathcal{A}'}$, specifically for $\chi(v)$ there is such a transition. Thus, $v \cdot e \in L(\mathcal{A}')$. Clearly, no other words are accepted by $\mathcal{A}'$. We conclude that $L(\mathcal{A}) = L(\mathcal{A}')$.

We continue to prove that for every word $w \in L(\mathcal{A})$ we have $1/t \cdot val(\mathcal{A}, w) \leq val(\mathcal{A}', w) \leq t \cdot val(\mathcal{A}, w)$. Since for every word $v \in V$, we have $val(\mathcal{A}, v) = val(\mathcal{A}', v) = 0$, the words that are candidates to violate the approximation are of the form $v \cdot e$. Assume towards contradiction that there is such a violating word $w = v \cdot e$. That is, either $1/t \cdot val(\mathcal{A}, w) > val(\mathcal{A}', w)$ or $val(\mathcal{A}', w) > t \cdot val(\mathcal{A}, w)$.

We prove that both cases are not possible. Note that $\mathcal{A}'$ assigns, to two-letter words, either the value t or $t+1$. Recall that $\mathcal{A}$ assigns, to two-letter words, one of three values: 1, t, or $t^2 + 1$. Thus, if $val(\mathcal{A}', w) > t \cdot val(\mathcal{A}, w)$, then $val(\mathcal{A}, w) = 1$ and $val(\mathcal{A}', w) = t + 1$. Let q_{in}, c, q_{acc} be the run of $\mathcal{A}'$ on w. Recall that the weight of the transition $\langle q_{in}, v, c\rangle$ is 0. Thus, the weight of the transition $\langle c, e, q_{acc}\rangle$ is $t + 1$. Hence, by the definition of the weight function, for every state $u \in V$ with $\chi(u) = c$, the weight $\tau_{\mathcal{A}}(\langle u, e, q_{acc}\rangle)$ is either $t^2 + 1$ or t. Since we assume $\chi(v) = c$, this holds also for v. Thus, $val(\mathcal{A}, w)$ is either $t^2 + 1$ or t and not 1, which contradicts the assumption that $val(\mathcal{A}, w) = 1$.

We continue to prove that the second option is also not possible because it leads to a contradiction of the legality of the coloring χ. As in the above, if $1/t \cdot val(\mathcal{A}, w) > val(\mathcal{A}', w)$, then $val(\mathcal{A}, w) = t^2 + 1$ and $val(\mathcal{A}', w) = t$. That is, we find a vertex $v' \in V$ such that $\langle v, v'\rangle \in E$ and $\chi(v') = \chi(v) = c$, where recall that $w = v \cdot e$. Recall that by the definition of the weight function of $\mathcal{A}'$, since $\tau_{\mathcal{A}'}(\langle c, e, q_{acc}\rangle = t$, there is a vertex $v' \in V$ with $\tau_{\mathcal{A}}(\langle v, e, q_{acc}\rangle) = 1$. We claim that $\langle v, v'\rangle \in E$. Indeed, since $val(\mathcal{A}, w) = t^2 + 1$, we have $\tau_{\mathcal{A}}(\langle v, e, q_{acc}\rangle) = t^2 + 1$, and by the definition of the weight function in $\mathcal{A}$, there is an edge $\langle v, v'\rangle \in E$, and we are done.

We conclude that $\mathcal{A}'$ t-approximates $\mathcal{A}$, and that the number of states in $\mathcal{A}'$ is $k + 2$. Thus, $\langle \mathcal{A}, k\rangle \in$ MIN-t-APPROX.

We continue to prove the other direction of the reduction. Assume that $\langle \mathcal{A}, k\rangle \in$ MIN-t-APPROX. That is, there is a deterministic automaton $\mathcal{A}'$ with $k + 2$ states that t-approximates $\mathcal{A}$. We show that $\langle V, E, k\rangle \in$ VC by computing a coloring function $\chi : V \to \{1, \ldots, k\}$.

Since $\mathcal{A}'$ is deterministic, it has a single initial state q_{in}. We claim that for every $v \in V$, there is an outgoing transition from q_{in} labeled with v into an accepting state. This follows from the fact that $L(\mathcal{A}) = L(\mathcal{A}')$ and, $V \subseteq L(\mathcal{A})$. Since every word that starts with a letter $e \in E$ is not in $L(\mathcal{A})$, if there is an outgoing transition from q_{in} that is labeled with some $e \in E$, then it leads to an *empty* state. Thus, we can assume that no such outgoing transitions exist. That is, every outgoing transition from q_{in} is labeled with a letter in V.

We continue to prove that there are at most k outgoing transitions from q_{in}. Recall that the size of $\mathcal{A}'$ is at most $k + 2$. Thus, we prove that there is no self loop on q_{in} and that there is a state that does not have an incoming transition from q_{in}.

We start by proving that there is no self loop on q_{in}. Recall that since $V \subseteq L(\mathcal{A}')$, every outgoing transition from q_{in} leads to an accepting state. If for some letter $v \in V$

there is a self loop $\Delta_{\mathcal{A}'}(q_{in}, v, q_{in})$, then q_{in} is accepting in $\mathcal{A}'$, and moreover, $\mathcal{A}'$ accepts the words of the form v^*. This is a contradiction to the fact that $L(\mathcal{A}) = L(\mathcal{A}')$.

We continue to prove that there is a state that does not have an incoming transition from q_{in}. Assume by way of contradiction that there are $k+1$ outgoing transitions from q_{in}. We show that there is a three letter word that is accepted by $\mathcal{A}'$, which contradicts the fact that $L(\mathcal{A}) = L(\mathcal{A}')$. Consider two letters $v \in V$ and $e \in E$. Since $v \cdot e \in L(\mathcal{A})$, there is a run of $\mathcal{A}'$ on $v \cdot e$. Let q_{in}, q, q' be that run. Since by the above there is no self-loop on q_{in}, by our assumption, $\mathcal{A}'$ has $k+2$ states, and q_{in} has $k+1$ outgoing transitions, by the pigeon-hole principle, there is a letter $v' \in V$ such that $\langle q_{in}, v', q' \rangle \in \Delta_{\mathcal{A}'}$. Since for every $e \in E$, we have $v' \cdot e \in L(\mathcal{A})$, there is an accepting run q_{in}, q', q'' of $\mathcal{A}'$ on $v' \cdot e$. Thus, q_{in}, q, q', q'' is an accepting run of $\mathcal{A}'$ on $v \cdot e \cdot e$. We conclude that $\mathcal{A}'$ accepts a three-letter word, which contradicts the assumption that $L(\mathcal{A}) = L(\mathcal{A}')$.

We conclude that there are at most k outgoing transitions from q_{in}, and they are labeled with the letters V. Thus, we can define the following coloring function: for every $v \in V$ we define $\chi(v) = c$ such that $\Delta_{\mathcal{A}'}(q_{in}, v, c)$.

We continue to prove that this is a legal coloring. Assume by way of contradiction that there are two vertices that are connected by an edge and have the same color. That is, there exist $v, v' \in V$ with $\chi(v) = \chi(v') = c$ and $e = \{v, v'\} \in E$. We assume $v < v'$. Thus, by the definition of the weight function in $\mathcal{A}$, we have $\tau_{\mathcal{A}}(\langle v, e, q_{acc} \rangle) = 1$ and $\tau_{\mathcal{A}}(\langle v', e, q_{acc} \rangle) = t^2 + 1$. Thus, $val(\mathcal{A}, v \cdot e) = 1$ and $val(\mathcal{A}, v' \cdot e) = t^2 + 1$. Recall that since $\chi(v) = \chi(v') = c$, the run of $\mathcal{A}'$ on the words $v \cdot e$ and $v' \cdot e$ is q_{in}, c, q_{acc}. It is easy to see that $\tau_{\mathcal{A}'}(\langle q_{in}, v, c \rangle) = \tau_{\mathcal{A}'}(\langle q_{in}, v', c \rangle) = 0$. Let $\tau_{\mathcal{A}'}(\langle c, e, q_{acc} \rangle) = x$. Thus, $val(\mathcal{A}', v \cdot e) = val(\mathcal{A}', v' \cdot e) = x$. By the definition of t-approximation, we have $val(\mathcal{A}', v \cdot e) \leq t \cdot val(\mathcal{A}, v \cdot e)$. By evaluating the equation we get: $x \leq t \cdot 1$. Also, by the definition of t-approximation, we have $1/t \cdot val(\mathcal{A}, v' \cdot e) \leq val(\mathcal{A}', v' \cdot e)$. By evaluating the equation we get: $1/t \cdot (t^2 + 1) \leq x$. Combining the two equations we get $t^2 + 1 \leq t^2$, which is a contradiction.

We conclude that the coloring is a legal coloring with at most k colors, and thus $\langle V, E, k \rangle \in$ VC, and we are done.

$\square$

A.5 Proof of Lemma 1

Assume $\mathcal{P}$ is not empty. Thus, there is a point $p \in \mathcal{P}$. Consider point $p' \in \mathbb{R}^k$ for which the values are at least that of p. That is, for every $1 \leq i \leq k$, we have $p'_i \geq p_i$. Since there is only one bound in this problem, we claim that $p' \in \mathcal{P}$. Indeed, consider a critical path π in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. Since the values of p' are at least these of p, the value of π in $\mathcal{S}_{\mathcal{A},\mathcal{B}^{p'}}$ is at least its value in $\mathcal{S}_{\mathcal{A},\mathcal{B}^{p}}$, which is non-positive because $p \in \mathcal{P}$.

We generate $k+1$ affinely independent points in $\mathcal{P}$. For every $1 \leq i \leq k$, we define the point $p^i \in \mathbb{R}^k$ by increasing the value of p_i by $\epsilon > 0$. That is, for every $1 \leq j \leq k$ such that $j \neq i$ we have $p^i_j = p_j$, and $p^i = p_i + \epsilon$. By the above, $p \in \mathcal{P}$ implies that $p^i \in \mathcal{P}$. The points $p, p^1, \ldots, p^k$ are affinely independent because the vectors $p^1 - p, p^2 - p, \ldots, p^k - p$ are linearly independent (recall that the vectors $\langle 1, 0, \ldots, 0 \rangle, \langle 0, 1, 0, \ldots, 0 \rangle, \ldots, \langle 0, 0, \ldots, 0, 1 \rangle$ are a base for the vector space $\mathbb{R}^k$), and we are done.

A.6 Proof of Lemma 2

Recall that it is possible to represent $\mathcal{P}$ implicitly as $\{x \in \mathbb{R}^k : Ax \leq b\}$, where $k = |\mathcal{X}|$ and every row in the matrix A together with the corresponding value in B is an inequality that is generated from some critical path in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. Assume $v = \{x \in \mathcal{P} : A'x = b'\} \in \mathcal{P}$ is a vertex, where A' and b' are sub-systems of A and b. We prove that v is a minimal assignment. Assume towards contradiction that it is not. That is, there is an index $1 \leq i \leq k$ and $\epsilon > 0$, such that the assignment v' that is constructed by reducing the value of v_i by ϵ results in a legal assignment. Recall that since v is a vertex, the rank of A' is k. Thus, there is a row a in A' with $a_i \neq 0$. Also recall that the inequalities that define $\mathcal{P}$ are generated from paths in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. For $1 \leq i \leq k$, the entry a_i represents the number of times the variable X_i appears in the path from which the inequality that corresponds to a was generated. Thus, the values in a are all positive. Combining the above we have that $a \cdot v' = a \cdot v + \epsilon \cdot a_i > a \cdot v = b$. Thus, v' violates an inequality, which contradicts the fact that $v' \in \mathcal{P}$.

A.7 Proof of Lemma 3

Before proving the lemma, we define vertices in a different manner. Consider a polytope P. We define the set char.cone$P = \{y \in \mathbb{R}^k : \forall x \in P,\ x + y \in P\}$. Intuitively, the non-zero vectors in char.cone$P$ are the *infinite directions* of P. Some properties of char.cone$P$ are that $P +$char.cone$P = P$, and P is bounded iff char.cone$P = \{0\}$. Also, it is easy to see that $y \in$ char.cone$P$ iff there exists $x \in P$ such that for all $\lambda \geq 0$, we have $x + \lambda y \in P$. Recall that P is pointed if it has a vertex. We use the following property of char.cone$P$: the dimension of char.cone$P \cap -$char.cone$P$ is 0 iff P is pointed.

We continue to prove the lemma. Assume by way of contradiction that $\mathcal{P} \neq \emptyset$ is not pointed. Thus, there is a point $0 \neq p \in$ char.cone$\mathcal{P} \cap -$char.cone$\mathcal{P}$. Consider an index $1 \leq i \leq k$, such that $p_i \neq 0$. Recall that we assume that the variable $X_i \in \mathcal{X}$ is reachable in $\mathcal{B}$. Consider an acyclic path of transitions $\pi = e_1, \ldots, e_l$ in $\mathcal{S}_{\mathcal{A},\mathcal{B}}$ such that π ends with a transition with $X_i \in \mathcal{X}$. That is, $\tau_{\mathcal{B}}(e_l) = X_i$, and note that for every $1 \leq j < l$, we have $\tau_{\mathcal{B}}(e_j) \neq X_i$. Thus, e_l is the first transition in π in which the variable X_i appears. Let $\sum_{1 \leq j \leq k} -l_j X_j + c \leq 0$, be the inequality that corresponds to π. Assume WLog that $X_1, \ldots, X_{i-1} \in \mathcal{X}$ are the placeholders that appear in π before X_i in the order in which they appear in π. That is, there are prefixes $\pi^1, \pi^2, \ldots, \pi^{i-1}$ of π. The length of the subpath π^1 is $|\pi^1| = l^1 \leq l$, the subpath $\pi^1[1 : l^1 - 1]$ of π^1 (π^1 without the last transition) has no variables, and the value of the last transition in π^1 is $X_1 \in \mathcal{X}$. The length of the subpath π^2 is $|\pi^2| = l^2 > l^1$, the only variable that appears in the subpath $\pi^2[1 : l^2 - 1]$ of π^2 (π^2 without the last transition) is X_1, and the value of the last transition in π^2 is X_2. The subpaths $\pi^3, \ldots, \pi^{i-1}$, are defined similarly.

A crucial point in the proof is the following. Since π is an acyclic path, so are the subpaths $\pi^1, \ldots, \pi^{i-1}$. Since we assume all the states in the automata are accepting, the prefixes $\pi^1, \ldots, \pi^{i-1}$ are critical paths, and thus, each prefix generates an inequality that defines the polytope $\mathcal{P}$.

We prove by induction on i that for the point $p \in \mathbb{R}^k$, we have that $p_1, \ldots, p_i = 0$, and thus reach a contradiction to the assumption that $p_i \neq 0$. For the base case, we prove that $p_1 = 0$. Assume towards contradiction that $p_1 \neq 0$. Consider the subpath π^1 as above, that ends in X_1 and has no other variables. Let $-X_1 + c \leq 0$, be the inequality that corresponds to this path. Since $p \in$ char.cone$\mathcal{P}$, there is a point $q \in \mathcal{P}$ such that for every $\lambda \geq 0$, we have $q + \lambda p \in \mathcal{P}$. Let $\lambda = \frac{c-q_1}{p_1} - 1$, where recall that we assume $p_1 \neq 0$. Since $q \in \mathcal{P}$, we have $-q_1 + c \leq 0$. Since $q + \lambda p \in \mathcal{P}$, it follows that $0 \geq -(q_1 + \lambda p_1) + c = -(q_1 + p_1 \cdot \frac{c-q_1}{p_1} - 1) + c = 1$, which is a contradiction. Thus, we conclude that $p_1 = 0$.

For the induction step, assume that $p_1, \ldots, p_j = 0$. We prove that $p_{j+1} = 0$. Assume towards contradiction that $p_{j+1} \neq 0$. Consider the subpath π^{j+1} of π as above, and let $-l_1X_1 - l_2X_2 - \ldots - l_jX_j - X_{j+1} + c \leq 0$ be the inequality that corresponds to π^{j+1}. As in the base case, since $p \in$ char.cone$\mathcal{P}$, there is a point $q \in \mathcal{P}$ such that for every $\lambda \geq 0$, we have $q + \lambda p \in \mathcal{P}$. Let $\lambda = \frac{-l_1q_1 - \ldots - l_jq_j - q_{j+1} + c - 1}{p_{j+1}}$. Since $q \in \mathcal{P}$, we have that $-l_1q_1 - l_2q_2 - \ldots - l_jq_j - q_{j+1} + c \leq 0$. Since $q + \lambda p \in \mathcal{P}$, we have $-l_1(q_1 + \lambda p_1) - \ldots - l_j(q_j + \lambda p_j) - (q_{j+1} + \lambda p_{j+1}) + c \leq 0$. By the induction hypothesis $p_1, \ldots, p_j = 0$. Thus, $0 \geq -l_1q_1 - \ldots - l_jq_j - (q_{j+1} + \lambda p_{j+1}) + c = -l_1q_1 - \ldots - l_jq_j - (q_{j+1} + \frac{-l_1q_1 - \ldots - l_jq_j - q_{j+1} + c - 1}{p_{j+1}} \cdot p_{j+1}) + c = 1$, which is a contradiction, and we are done.

A.8 Proof of Theorem 6

Consider an input to the 3-SAT problem, which is a formula $\psi = \bigwedge_{1 \leq i \leq m} C_i$, where every clause C_i is of the form $l_1^i \vee l_2^i \vee l_3^i$ and every literal is variable x_j or its negation $\neg x_j$. We assume that there are n variables in ψ. In $\mathcal{X}$ there are $2n$ variables, two for every variable in ψ. We denote variables in ψ with lowercase x and variables in $\mathcal{X}$ with uppercase X. We sometimes refer to literals as variables in $\mathcal{X}$. That is, if we have that $l_r^i = x_j$ we refer to the corresponding variable $X_j \in \mathcal{X}$ as l_r^i, and similarly, if $l_r^i = \neg x_j$ we refer to the corresponding variable $\overline{X_j} \in \mathcal{X}$ as l_r^i. The reduction is described in Figure A.8. Clearly, it is polynomial in the size of φ. Note that all the states in $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ are accepting.

In the following, we prove that an assignment $f : \mathcal{X} \to \mathbb{Q}$ is legal iff it satisfies two properties:

- for every clause $1 \leq i \leq m$, there is a literal l_r^i, where $1 \leq r \leq 3$ for which $f(l_r^i) \geq 1$ and
- for every $1 \leq j \leq n$ we have $f(X_i) + f(\overline{X_i}) \leq 0$.

We show that the claim implies the correctness of the reduction. For the first direction, consider a legal assignment $f : \mathcal{X} \to \mathbb{Q}$. We claim that there is a satisfying assignment $g : \{x_1, \ldots, x_n\} \to \{0, 1\}$ for the variables in ψ. We define g as follows. For every variable x in ψ, if the corresponding variable $X \in \mathcal{X}$ has $f(X) \geq 1$ then $g(x) = 1$. Otherwise, by the second property we have $f(\overline{X}) \geq 1$ and we define $g(x) = 0$. By the first property, g is a legal assignment. For the other direction, given a satisfying assignment g to the variables in ψ, we construct an assignment f for the variables in $\mathcal{X}$. For every variable x in ψ, if $g(x) = 1$, we set $f(X) = 1$, and otherwise $g(x) = 0$ and we set $f(\overline{X}) = 1$. Clearly, the second property holds. Also, since the assignment is legal, the first property holds. Thus, by the claim, f is a legal assignment.

We continue to prove the claim. Consider an assignment $f : \mathcal{X} \to \mathbb{Q}$. We claim that f satisfies $\mathcal{A} \leq \mathcal{B}^f$ iff it satisfies the first property, namely that for every clause, there is a literal l_r^i for which $f(l_r^i) \geq 1$. Recall that in the simulation game that corresponds to $\mathcal{A}$

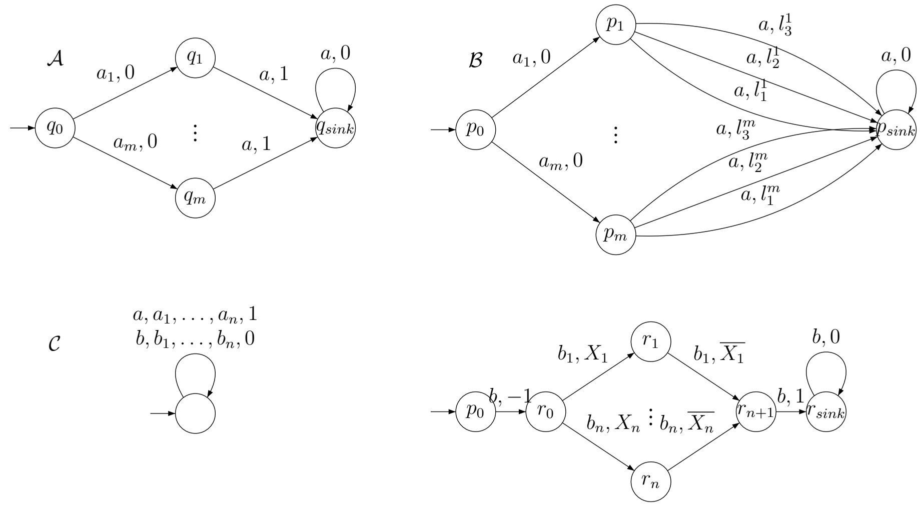

Fig. 5. The reduction from 3-SAT to weighted query checking for WFAs.

and $\mathcal{B}$, Player 1 chooses the letters and controls the moves of $\mathcal{A}$, and Player 2 reacts with moves in $\mathcal{B}$. Consider a Player 1 strategy and the outcome of the game after the first round of play. That is, Player 1 chooses a letter a_i and proceeds to q_i for some $1 \leq i \leq m$. Player 2, since he has no other option, reacts by choosing a_i and proceeding to p_i. At this point Player 1 must choose a and proceed to q_{sink}. Player 2 must react by choosing an edge with value at least 1. Otherwise, after his move, the game reaches a winning configuration for Player 1. Thus, Player 2 wins against this Player 1 strategy iff there is a literal l_r^i with $f(l_r^i) \geq 1$. We conclude that Player 2 wins the game, i.e., $\mathcal{A} \leq \mathcal{B}^f$, iff for every clause there is a literal l_r^i with $f(l_r^i) \geq 1$, and we are done.

For the other part of the proof, we prove that a valuation f satisfies $\mathcal{B}^f \leq \mathcal{C}$ iff for every $1 \leq j \leq n$ we have $f(X_j) + f(\overline{X_j}) \leq 0$. Note that the bottom part of the parameterized automaton $\mathcal{B}$ and the automaton $\mathcal{C}$ are deterministic. Also note, that in the simulation game corresponding to $\mathcal{B}$ and $\mathcal{C}$, Player 1 has no hope of winning by playing the letters $a, a_1, \ldots, a_n$. Also, if the game reaches r_{sink}, clearly Player 2 will win the game. Thus, we are interested in plays of length 4 where the Player 1 strategy uses the letters $b, b_1, \ldots, b_n$. Consider such a Player 1 strategy. That is, consider a Player 1 strategy in which he choses the letters b, b_i, b_i, b for some $1 \leq i \leq n$. The outcome of the game after 4 moves is $\tau_{\mathcal{B}}(\langle p_0, r_0 \rangle) + \tau_{\mathcal{B}}(\langle r_0, r_i \rangle) + \tau_{\mathcal{B}}(\langle r_i, r_{n+1} \rangle) + \tau_{\mathcal{B}}(\langle r_{n+1}, r_{sink} \rangle) = -1 + f(X_i) + f(\overline{X_i}) + 1$. Recall that Player 1 wins this prefix iff its value is positive, which is iff $f(X_i) + f(\overline{X_i}) > 0$. Thus, Player 2 wins the game, i.e., $\mathcal{B}^f \leq \mathcal{C}$, iff for every $1 \leq i \leq n$ we have that $f(X_i) + f(\overline{X_i}) \leq 0$, and we are done.

A.9 Proof of Theorem 8

Consider an input to the PWC problem, which is WFAs $\mathcal{A}$ and $\mathcal{C}$ and a parameterized WFA $\mathcal{B}$. Let n and M, as they are defined in Remark 1. We claim that if there is a point $p \in \mathbb{R}^k$ for which $\mathcal{A} \leq \mathcal{B}^p \leq \mathcal{C}$, then there is also a point $p' \in \mathbb{R}^k$ with size polynomial in n and M that satisfies the relation. Note that the claim implies the theorem because we can guess a point $q \in \mathbb{R}^k$ with polynomial size, and, since we assume a polynomial algorithm for solving simulation games, we can check, in polynomial time, if $\mathcal{A} \leq \mathcal{B}^q \leq \mathcal{C}$ by solving two simulation games: the first corresponds to $\mathcal{A}$ and $\mathcal{B}^q$ and the second to $\mathcal{B}^q$ and $\mathcal{C}$.

Before proving the claim, we briefly recall some properties of simulation games. Intuitively, a strategy is a recipe that, given the history of the play, tells the player how he should continue playing. A *memoryless strategy* is a strategy that does not consider the history of the play, but only the current position. We show in [3] that if Player 2 wins a simulation game, he has a winning memoryless strategy in the game. Such a strategy is a function $\rho_2 : V_2 \to V_1$. That is, given a Player 2 vertex $v \in V_2$ a memoryless Player 2 strategy ρ_2 tells Player 2 to move to $\rho_2(v)$.

Given a Player 2 memoryless strategy ρ_2, we can *trim* the game arena. For every vertex $v \in V_2$, we remove every edge that does not coincide with ρ_2. The outcome is a *one-player* game, where the single player is Player 1, the antagonist. Note that Player 1 wins the game iff one of three conditions hold: (1) there is a an acyclic path from the initial vertex to a good state, (2) there is a critical path with positive value, where a critical path, similar to the deterministic case, is an acyclic path from the initial vertex to a risky state, or a simple cycle that is both reachable from the initial state and has a risky state that is reachable from it.

We continue to prove the claim. Consider a point $p \in \mathbb{R}^k$ for which $\mathcal{A} \leq \mathcal{B}^p \leq \mathcal{C}$. Thus, Player 2 wins the simulation game that corresponds to $\mathcal{A}$ and $\mathcal{B}^p$ and he also wins the simulation game that corresponds to $\mathcal{B}^p$ and $\mathcal{C}$. Consider two Player 2 memoryless winning strategies ρ_2^1 in the first game and ρ_2^2 in the second game. We define two polytopes: $\mathcal{P}_1 \subseteq \mathbb{R}^k$ is the set of points for which the strategy ρ_2^1 is winning, and $\mathcal{P}_2 \subseteq \mathbb{R}^k$ is the set of points for which ρ_2^2 is winning. That is, for every point $p' \in \mathcal{P}_1$ we have that $\mathcal{A} \leq \mathcal{B}^{p'}$, and, moreover, ρ_2^1 is a winning Player 2 strategy in the corresponding simulation game, and similarly for $\mathcal{P}_2$.

We continue to define the polytopes. We define the polytope $\mathcal{P}_1$ and the definition for $\mathcal{P}_2$ is similar. We construct a *parameterized simulation game* that corresponds to $\mathcal{A}$ and $\mathcal{B}$, in the expected way. Then, we trim the arena according to the Player 2 strategy ρ_2^1. We claim that there are no acyclic paths that end in a good vertex for Player 1. Indeed, otherwise, ρ_2^1 would not be winning in the game corresponding to $\mathcal{A}$ and $\mathcal{B}^p$. Similar to Section 2, every critical path imposes a restriction on assignments. That is, consider a point $p' \in \mathcal{P}_1$ and such a path π in the parameterized game corresponding to $\mathcal{A}$ and $\mathcal{B}$. The value of π in the game corresponding to $\mathcal{A}$ and $\mathcal{B}^{p'}$ is not positive, because otherwise ρ_2^1 is not a winning strategy for Player 2. We define the polytope $\mathcal{P}_1$ to be the intersection of the inequalities that correspond to the restrictions.

Similarly to Section 2, the size of every inequality that defines $\mathcal{P}_1$ and $\mathcal{P}_2$ is polynomial in M and n. Consider the polytope $\mathcal{P}_1 \cap \mathcal{P}_2$. That is, the polytope that is defined by the inequalities that define $\mathcal{P}_1$ combined with those that define $\mathcal{P}_2$. Clearly, every inequality that defines $\mathcal{P}_1 \cap \mathcal{P}_2$ is polynomial in M and n. Note that every point $q \in \mathcal{P}_1 \cap \mathcal{P}_2$ satisfies $\mathcal{A} \leq \mathcal{B}^q$, because $q \in \mathcal{P}_1$, and also satisfies $\mathcal{B}^q \leq \mathcal{C}$, because $q \in \mathcal{P}_2$. Since $\mathcal{A} \leq \mathcal{B}^p \leq \mathcal{C}$ we have $p \in \mathcal{P}_1 \cap \mathcal{P}_2$. Thus, $\mathcal{P}_1 \cap \mathcal{P}_2 \neq \emptyset$. By Theorem 7 there is a point in $\mathcal{P}_1 \cap \mathcal{P}_2$ with size polynomial in M and n, and we are done.