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. 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 [24,26]. 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 [23] and program repair [15]. 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 $\text{ALWAYS}(X_1 \to \text{EVENTUALLY}\, grant)$ 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 [8,20].

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 [1,18]. 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]

[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

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

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 [19]. 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

different, and the goal is to find restrictions on the behavior of the clocks such that the automata satisfy certain properties.

construct the DWFAs $\mathit{reduce}(\mathcal{A}, t)$ and $\mathit{increase}(\mathcal{A}, t)$, 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 $\mathit{reduce}(\mathcal{A}, t) \subseteq \mathcal{B} \subseteq \mathit{increase}(\mathcal{A}, t)$.

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 viewing the Boolean product of $\mathcal{A}$ and $\mathcal{C}$ as a partially specified WFA.

Let us now return to parameterized weighted containment where a partial WFA $\mathcal{B}$ is given. Our solution to the problem is based on 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* [17] 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 [12,16,21] 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.

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

Due to the lack of space, some of the proofs and examples are omitted in this version and can be found in the full version, in the authors' homepages.

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 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 $w = w_1, \ldots, w_n \in \Sigma^*$ 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 $\Delta(r_i, w_{i+1}, r_{i+1})$. The run r is accepting iff $r_n \in F$. The value of the run, denoted $val(r, w)$, is the sum of costs of transitions it traverses. That is, $val(r, w) = \sum_{0 \leq i < n} \tau(\langle r_i, w_{i+1}, r_{i+1} \rangle)$. 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 $w \in \Sigma^*$, denoted $val(\mathcal{A}, w)$, as the value of the maximal-valued accepting run of $\mathcal{A}$ on w. That is, $val(\mathcal{A}, w) = max\{val(r, w) :$ r is an accepting run of $\mathcal{A}$ on $w\}$. 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 [1,18] 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 $val(\pi) \leq 0$ for every critical path π 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 [22].

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 the full version 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].

[4] In [22], 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.

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 [17] 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 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.

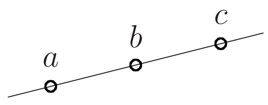

Fig. 1. A 1-dimensional polytope in $\mathbb{R}^2$.

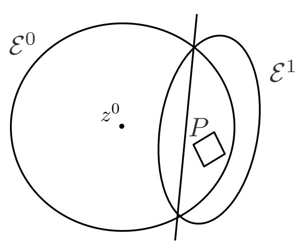

Fig. 2. An illustration of generating the ellipsoid $\mathcal{E}^1$, given that the center z^0 of the ellipsoid $\mathcal{E}^0$ violates an inequality. Note that the polytope P is contained in $\mathcal{E}^0$ and in $\mathcal{E}^1$. Also note that $\mathcal{E}^1$ contains the intersection between $\mathcal{E}^0$ 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 [12,16,21] 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 the full version 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. *[22] 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, [25] 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 $L(\mathcal{A}) \subseteq L(\mathcal{B}) \subseteq L(\mathcal{C})$. It is easy to see that the weighted sandwich problem is at least as hard as the Boolean one. Indeed, by defining all costs to be 0, we get an easy reduction between the two. In order to neutralize the difficulty of the Boolean aspect of the language, we define a pure-weighted sandwich problem, where $\mathcal{A}$ and $\mathcal{C}$ are such that $L(\mathcal{A}) = L(\mathcal{C})$, and we are looking for a minimal $\mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. Note that such a WFA $\mathcal{B}$ exists iff $\mathcal{A} \subseteq \mathcal{C}$. As we show now, all sandwich problems are difficult.

Theorem 3. *The Boolean, weighted, and pure-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 start with the Boolean case and 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.

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 the full version.

As noted above, hardness in the Boolean setting implies hardness in the weighted variant. For the pure-weighted variant, we go through the t-approximation problem for DWFAs, 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 $\mathcal{A}$. As detailed in Section 1, t-approximation can be easily reduced to the pure-weighted sandwich problem. We prove that t-approximation is NP-hard by a reduction from VC. 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 the full version. $\square$

We suggest a heuristic to cope with the complexity of the pure-weighted sandwich problem. Consider DWFAs $\mathcal{A}$ and $\mathcal{C}$ such that $L(\mathcal{A}) = L(\mathcal{C})$ and $\mathcal{A} \subseteq \mathcal{C}$. We start the search for $\mathcal{B}$ with a DFA $\mathcal{D}$ that has the state space $Q_{\mathcal{A}} \times Q_{\mathcal{C}}$. Note that since $L(\mathcal{A}) = L(\mathcal{C})$ and the automata are deterministic, we have that $(F_{\mathcal{A}} \times Q_{\mathcal{C}}) \cup (Q_{\mathcal{A}} \times F_{\mathcal{C}}) = F_{\mathcal{A}} \times F_{\mathcal{C}}$, which we define as the set of $\mathcal{D}$'s accepting states. We try to minimize $\mathcal{D}$ by iteratively searching for states that can be merged. Clearly, we cannot hope to obtain an automaton with fewer states than the one required for $L(\mathcal{B})$, thus candidates for merging are states that are merged in the standard DFA minimization algorithm. Consider two such states $q, q' \in Q_{\mathcal{D}}$. Let $\mathcal{D}'$ be the DFW obtained by merging q and q'. We label each transition of $\mathcal{D}'$ with a different variable and use parametric weighted containment in order to find a DWFA $\mathcal{B}$ on the structure of $\mathcal{D}'$ that satisfies $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{C}$. If we find, we continue with further mergings in $\mathcal{D}'$. Otherwise, we un-merge the states, and look for new candidates.

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. *[22] 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 $\texttt{SEP}$ 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 $\texttt{SEP}$.*

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.*

Remark 3. In [11], the authors define *functional weighted automata*, which are non-deterministic weighted automata in which all the accepting runs on a word have the same value. The authors show that in this model, containment is decidable: given two functional weighted automata $\mathcal{A}$ and $\mathcal{B}$, we check if $L(\mathcal{A}) \subseteq L(\mathcal{B})$, and then we check if for every word $w \in \Sigma^*$, we have $val(\mathcal{A}, w) \leq val(\mathcal{B}, w)$ by reasoning on the automaton $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. It is easy to extend the technique presented in this section to functional automata. Essentially, as in the case of deterministic automata, the construction of the separation oracle can be based on $\mathcal{S}_{\mathcal{A},\mathcal{B}}$. Accordingly, the computational bottleneck is the Boolean $L(\mathcal{A}) \subseteq L(\mathcal{B})$ check, making the PWC problem for functional automata PSPACE-complete.

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]

[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.

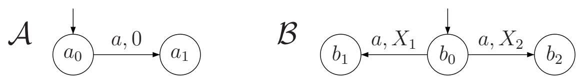

Fig. 3. An example in which there is no minimal assignment.

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 the full version. $\square$

For the upper bound, we first need the following result by Schrijver, relating the size of the inequalities that define a polytope P and the size of its vertices.

Theorem 7. *[22] 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 now rely on the fact that when Player 2 wins the weighted simulation game, he has a *memoryless winning strategy* [3]. Consequently, we can trim the two arenas and, as in the deterministic case, represent the set of assignments that are legal for these specific strategies by a k-dimensional polytope. Thus, we use Theorem 7 to show that if there is a legal assignment, then there is one of polynomial size. Now, if we assume that deciding weighted simulation can be done in polynomial time, we can conclude with the following. For the full proof see the full version.

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

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*, LNCS 7454, 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*, LNCS 5213, 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*, LNCS 510, pages 93–114, 1991.
11. E. Filiot, R. Gentilini, and J.-F. Raskin. Quantitative Languages Defined by Functional Automata. In *Proc. 23rd CONCUR*, LNCS 7454, pages 132–146, 2012.
12. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.
13. M. Grötschel, L. Lovász, and A. Schrijver. Corrigendum to our paper "the ellipsoid method and its consequences in combinatorial optimization". *Combinatorica*, 4(4), 1984.
14. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York, 1988.
15. B. Jobstmann, A. Griesmayer, and R. Bloem. Program repair as a game. In *Proc. 17th CAV*, LNCS 3576, pages 226–238, 2005.
16. R. Karp and C. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *Proc. 21st FOCS*, pages 1–9, 1980.
17. L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.
18. 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.
19. M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
20. 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.
21. 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.
22. A. Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.
23. A. Solar-Lezama, R.M. Rabbah, R. Bodík, and K. Ebcioglu. Programming by sketching for bit-streaming programs. In *PLDI*, pages 281–294, 2005.
24. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
25. M. Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In *SWAT*, LNCS 3111, pages 384–396, 2004.
26. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *I& C*, 115(1):1–37, 1994.