Reasoning about Online Algorithms with Weighted Automata

Benjamin Aminof, Orna Kupferman, Robby Lampert
Hebrew University, School of Engineering and Computer Science, Jerusalem 91904, Israel
Email: {benj,orna,robil}@cs.huji.ac.il

December 2, 2008

Abstract

We describe an automata-theoretic approach for the competitive analysis of *online algorithms*. Our approach is based on *weighted automata*, which assign to each input word a cost in $\mathbb{R}^{\geq 0}$. By relating the "unbounded look ahead" of optimal offline algorithms with nondeterminism, and relating the "no look ahead" of online algorithms with determinism, we are able to solve problems about the competitive ratio of online algorithms, and the memory they require, by reducing them to questions about *determinization* and *approximated determinization* of weighted automata.

1 Introduction

In *formal verification*, we verify that a system has a desired property by checking whether a mathematical model of the system satisfies a formal specification of the property. Early work on formal verification handled finite-state hardware designs. Current work already copes with infinite-state software systems, complex distributed systems, and so on [2, 15], and is widely and successfully used in the industry [9]. An important feature of formal verification is that rather than reasoning only about input/output relations of terminating systems (for example, the output is the gcd of the two numbers in the input), it enables reasoning about *reactive systems*, which maintain an on-going interaction with their environment. For example, one can check that an operating system never reaches a deadlock, or that every request in some communication protocol is eventually acknowledged.

In this work we extend the scope of formal verification to reasoning about *online algorithms*. An online algorithm can be viewed as a reactive system: at each round, the environment issues a request, and the algorithm should process it. The sequence of requests is not known in advance, and the goal of the algorithm is to minimize the overall cost of processing all the requests in the sequence. Online algorithms for many problems have already been extensively studied for several decades, and have aroused much interest, both from a practical and a theoretical point of view [4].

While the interaction described above between an online algorithm and its environment is at the heart of formal verification, the questions that are traditionally answered by formal verification techniques are very different from those that are asked in the context of online algorithms. In formal verification a system is checked with respect to a given specification. On the other hand, the most interesting question about an online algorithm refers to its *competitive ratio*: the worst-case (with respect to all input sequences) ratio between the cost of the algorithm and the cost of an optimal solution (one that may be given by an *offline algorithm*, which knows the input sequence in advance).

While current formal verification techniques can check qualitative properties of an online algorithm (e.g., "whenever a request to a page is made, and this page is not in the cache, the page is brought into the cache") and even answer quantitative questions about it (e.g., "what is the maximal number of page faults within a window of k rounds?") [8], current techniques cannot refer to the optimal solution, and hence, they cannot reason about the competitive ratio. Likewise, while synthesis algorithms and tools successfully generate systems that satisfy a given specification [16], current synthesis algorithms cannot, for example, synthesize (or decide that there does not exist) online algorithms that are as good, or competitive with some given ratio, as a given offline algorithm.

Our approach to formally reasoning about online algorithms is based on *weighted finite automata* (WFAs, for short) [18, 20]. Essentially, we relate the "unbounded look ahead" of the optimal offline algorithm with nondeterminizm, and relate the "no look ahead" of online algorithms with determinism. This enables us to reduce questions about the competitive ratio of online algorithms to questions about *determinization* and *approximated determinization* of WFAs. Below we further elaborate on our approach and our results.

A WFA $\mathcal{A}$ induces a partial *cost* function from Σ^* to $\mathbb{R}^{\geq 0}$. Technically, each transition of $\mathcal{A}$ has a cost, the cost of a run is the sum of the costs of the transitions taken along the run, and the cost of a word w, denoted $cost(\mathcal{A}, w)$, is the minimum cost over all accepting runs on it (the cost is undefined if no run on the word is accepting). Consider an optimization problem P with requests in Σ. An algorithm for P can be viewed as a mapping of words in Σ^+ to a set of actions available to the algorithm [3]. For a finite set S of configurations, we say that an algorithm uses memory S if there is a regular mapping of Σ^* into S such that the algorithm behaves in the same manner on identical continuations of words that are mapped to the same configuration.

The set of online algorithms for P that use memory S induces a WFA $\mathcal{A}_P$, with alphabet Σ and state space S, such that the transitions of $\mathcal{A}_P$ correspond to actions of the algorithms and the cost of each transition is the cost of the corresponding action. We argue that many optimization problems have algorithms that use finite memory. We demonstrate this on the paging, k-server, ski-rental, load-balancing, and Δ-paid exchange static list accessing problems.

Given a finite sequence of requests $w \in \Sigma^*$, each run of $\mathcal{A}_P$ on w corresponds to a way of serving the requests in w by an algorithm with memory S. The set of all runs include all such ways, thus $cost(\mathcal{A}, w)$ is the cost of an optimal offline algorithm on w that uses memory S. On the other hand, an online algorithm has to process each request as soon as it arrives. Hence, an online algorithm corresponds to a deterministic automaton *embodied* in $\mathcal{A}_P$. Indeed, for every configuration $s \in S$ and request $\sigma \in \Sigma$, the algorithm suggests a particular way to process σ from s, inducing a single transition labeled σ from s.

Accordingly, there exists an online algorithm for P that performs as well as the optimal offline algorithm iff $\mathcal{A}_P$ embodies an equivalent deterministic automaton, in which case we say that $\mathcal{A}_P$ is *determinizable by pruning*. Similarly, there exists an α-competitive online algorithm for P, for $\alpha > 1$, iff $\mathcal{A}_P$ embodies a deterministic automaton $\mathcal{A}'_P$ that α-approximates $\mathcal{A}_P$ (the automaton $\mathcal{A}'_P$ accepts the same set of words as $\mathcal{A}_P$, and $cost(\mathcal{A}'_P, w) \leq \alpha \cdot cost(\mathcal{A}_P, w)$ for all words w in this set). Then, we say that $\mathcal{A}_P$ is *α-determinizable by pruning*.

Restricting the determinization procedure to automata embodied in $\mathcal{A}_P$ guarantees that transitions in the automaton still correspond to actions of an algorithm for P. An online algorithm, however, may require more memory than an offline algorithm for the same problem. For example, in the paging problem, an offline algorithm only has to remember in each round the set of pages that are in the cache, whereas known online algorithms that achieve the best competitive ratio are marking algorithms,

which remember, in addition, some order on the pages in the cache, or some other information. To address this point, we also consider a variant of determinization by pruning that allows a refinement of the state space of $\mathcal{A}_P$ before pruning it to a deterministic automaton. We show that such a refinement indeed corresponds to an extension of the memory used by the algorithm.

We study the problems of deciding whether a WFA is determinizable (or α-determinizable) by pruning, with and without refinement. The problems are, in fact, challenging already for the unweighted case, and we first solve them in this setting[1]. We show that the problem of deciding whether a WFA is determinizable by pruning can be solved in polynomial time. Our algorithm makes use of the local nature of pruning – each state should have, for each input letter $\sigma \in \Sigma$, a σ-transition that "covers all other σ-transitions". The local nature of pruning, however, cannot be used when considering approximation, and we show that the problem of deciding whether a WFA is α-determinizable by pruning, for $\alpha > 1$, is NP-complete. It follows that given an optimization problem P and a finite set S of configurations, the problem of deciding whether there is an online algorithm for P with configurations in S, that is as good as an offline algorithm with configurations in S, can be solved in polynomial time. On the other hand, the problem of deciding whether there is an online algorithm for P with configurations in S that is α-competitive, for a fixed $\alpha > 1$, with respect to an offline algorithm with configurations in S, is NP-complete.

The complications that approximation brings with it are carried over to the setting in which an extension of the memory is allowed. We prove that while extending the memory cannot help an online algorithm to perform as well as the offline algorithm (that is, if an offline algorithm uses memory S, and no 1-competitive online algorithm with configurations in S exists, then there is no 1-competitive online algorithm at all), memory may help in order to decrease a competitive ratio $\alpha > 1$ (that is, there are problems for which an offline algorithm uses configurations in S, no online algorithm with configurations in S is α-competitive, but there is an online algorithm with richer configurations that is α-competitive)[2].

In Section 6, we discuss the practical aspects of implementing our framework. In particular, we discuss symbolic approaches that cope with the large state space that our algorithms handle, and parametric methods, which allow to reason about a system with many identical processes by studying properties of one of the processes.

1.1 Related work

Our automata-theoretic approach for reasoning about online algorithms adopts and extends ideas from work done in the formal-verification community. An automata-theoretic approach for reasoning about systems and their specifications has been suggested in [23], and has been extensively studied and implemented since then. As discussed above, the known approach is not suitable for reasoning about online algorithms. Determinization of WFA is studied in [20], but the technique and the applications are different from those of determinization by pruning, which we study here.

The online-algorithms community has studied several abstract models for competitive analysis. The on-going interaction that takes place in online algorithms can be modeled, for example, by means of *games in strategic form* [17] and *request-answer games* [3]. Other work considers models for

[1] The problem of determinization by pruning is of interest also in the unweighted case. As described in [14], automata that are determinizable by pruning can be used in the process of synthesis and game solving.

[2] We note that while it is widely believed that a k-competitive online algorithm for the paging problem needs more memory than the optimal offline algorithm, this is not the case [11].

specific problems (e.g., [1] for paging). The model that is closest to our automata-theoretic approach is the one of *metrical task systems* [5, 19]. The expressive power and the applications of the various models are different, however, from our weighted automata.

2 Preliminaries

2.1 Weighted automata

Standard automata map words in Σ^* to either "accept" or "reject". A weighted automaton can be viewed as a partial function (defined only for accepted words) from Σ^* to $\mathbb{R}^{\geq 0}$. Formally, a *weighted finite automaton* (WFA, for short) is $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$, where Σ is a finite input alphabet, Q is a finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation, $c : \Delta \to \mathbb{R}^{\geq 0}$ is a cost function, $Q_0 \subseteq Q$ is a set of initial states, and $F \subseteq Q$ is a set of final states. A transition $d = \langle q, a, p \rangle \in \Delta$ (also written as $\Delta(q, a, p)$) can be taken when reading the input letter a, and it causes $\mathcal{A}$ to move from state q to state p with *cost* $c(d)$. The transition relation Δ induces a transition function $\delta : Q \times \Sigma \to 2^Q$ in the expected way. Thus, for a state $q \in Q$ and a letter $a \in \Sigma$, we have $\delta(q, a) := \{p : \Delta(q, a, p)\}$. We extend δ to sets of states, by letting $\delta(S, a) := \bigcup_{q \in S} \delta(q, a)$, and recursively to words in Σ^*, by letting $\delta(S, \varepsilon) = S$, and $\delta(S, u \cdot a) := \delta(\delta(S, u), a)$, for every $u \in \Sigma^*$ and $a \in \Sigma$. A WFA $\mathcal{A}$ may be nondeterministic in the sense that it may have many initial states, and that for some $q \in Q$ and $a \in \Sigma$, it may have $\Delta(q, a, p_1)$ and $\Delta(q, a, p_2)$, with $p_1 \neq p_2$. If $|Q_0| = 1$ and for every state $q \in Q$ and letter $a \in \Sigma$ we have $|\delta(q, a)| \leq 1$, then $\mathcal{A}$ is a *deterministic* weighted finite automaton (DWFA, for short).

For a word $w = w_1 \dots w_n \in \Sigma^*$, a run of $\mathcal{A}$ on w is a sequence $r = r_0 r_1 \dots r_n \in Q^+$, where $r_0 \in Q_0$ and for every $1 \leq i \leq n$, we have $\langle r_{i-1}, w_i, r_i \rangle \in \Delta$. The run r is accepting if $r_n \in F$. The word w is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The (unweighted) *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \{w : w \text{ is accepted by } \mathcal{A}\}$. For $q \in Q$, we denote by $\mathcal{A}^q$ the automaton $\mathcal{A}$ with the single initial state q. The cost of an accepting run is the sum of the weights of the transitions that constitute the run[3]. Formally, let $r = r_0 r_1 \dots r_n$ be an accepting run of $\mathcal{A}$ on w, and let $d = d_1 \dots d_n \in \Delta^*$ be the corresponding sequence of transitions. The cost of r is $cost(\mathcal{A}, r) = \sum_{i=1}^{n} c(d_i)$. The cost of w, denoted $cost(\mathcal{A}, w)$, is the minimal cost over all accepting runs of $\mathcal{A}$ on w. Thus, $cost(\mathcal{A}, w) = \min\{cost(\mathcal{A}, r) : r \text{ is an accepting run of } \mathcal{A} \text{ on } w\}$. For completeness, if $w \notin L(\mathcal{A})$ we set $cost(\mathcal{A}, w) = \infty$.

For two WFAs $\mathcal{A}_1$ and $\mathcal{A}_2$, and $\alpha \geq 1$, we say that $\mathcal{A}_1$ *α-approximates* $\mathcal{A}_2$ if $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ and for all words $w \in \Sigma^*$, we have $cost(\mathcal{A}_1, w) \leq \alpha \cdot cost(\mathcal{A}_2, w)$. When both $\mathcal{A}_1$ 1-approximates $\mathcal{A}_2$ and $\mathcal{A}_2$ 1-approximates $\mathcal{A}_1$, we say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent*.

[3] In general, a WFA may be defined with respect to any semiring $\langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1} \rangle$. The cost of a run is then the semiring product of the weights along it, and the cost of an accepted word is the semiring sum over all accepting runs on it. For the modeling of online algorithms, we focus on weighted automata defined with respect to the *min-sum semiring*, $\langle \mathbb{R}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0 \rangle$ (sometimes called the *tropical* semiring), as defined above. Also, some work assigns costs also to initial and accepting states. We do not need such costs for the modeling of online algorithms, and work with a definition that omits them.

2.2 Online algorithms

A *problem* associates with each possible input I a set $F(I)$ of feasible solutions. In an *optimization problem* (of cost minimization), each solution in $F(I)$ has a cost in $\mathbb{R}^{\geq 0}$, and the goal is to find a feasible solution that minimizes the cost.

An *online algorithm* for an optimization problem P is an algorithm that gets as input a finite sequence of requests, and has to process each request (and end up in a feasible solution) without knowing the requests yet to come. In contrast, an *offline algorithm* for P gets the entire sequence in advance, and its decisions as to how to process a request may depend on the requests yet to come.

Formally, if we denote by Σ the set of requests, and denote by A the set of actions that are available to the algorithm, then an online algorithm corresponds to a function $g : \Sigma^+ \rightarrow A$. The processing of an input sequence $\sigma_1 \ldots \sigma_n$ by g is then $g(\sigma_1)$, $g(\sigma_1\sigma_2)$, $g(\sigma_1\sigma_2\sigma_3), \ldots$. In typical optimization problems, there is a cost function $action\_cost : A \rightarrow \mathbb{R}^{\geq 0}$ that associates a cost with each action. The cost of processing an input sequence is the sum of the costs of the actions taken in order to process it. The performance of an online algorithm is typically worse than that of an offline algorithm for the same problem. For analyzing the performance of online algorithms we use *competitive analysis*, which compares the two performances.

For an online algorithm g and an input $w \in \Sigma^+$, let $g(w)$ denote the cost of processing w by g, and let OPT(w) denote the cost of processing w by the optimal offline algorithm. We say that an online algorithm g is *α-competitive* if there exists a constant β such that for all input sequences $w \in \Sigma^+$ we have that $g(w) \leq \alpha \cdot \text{OPT}(w) + \beta$. The *competitive ratio* of g is the smallest α for which g is $alpha$-competitive. In the rest of the paper we restrict attention to the multiplicative factor α and ignore the additive factor β, except for places where it is not immediately clear how to handle β.

Our analysis of online algorithms takes into account the extra memory that the online algorithm may require in order to compete with the offline algorithm. Formally, we have the following.

Definition 2.1 *For a set S of configurations, a competitive ratio $\alpha \geq 1$, and an integer $r \geq 0$, we say that an optimization problem P has* competitive ratio (α, r) with memory S*, if there is an online algorithm g for P that uses an extension of the memory S by r Boolean variables, and g is α-competitive with respect to an optimal offline algorithm that uses memory S.*

3 An Automata-Theoretic Approach to Reasoning about Online Algorithms

In this section we describe an automata-theoretic approach to reasoning about online algorithms. We first characterize optimization problems for which the approach can be applied, and argue that typical optimization problems satisfy our characterization. We then describe how, by modeling optimization problems by weighted nondeterministic automata, we can reduce reasoning about the competitive ratio and the memory required by online algorithms, to reasoning about determinization of such automata.

3.1 Finite-state online algorithms

Recall that an online algorithm corresponds to a function $g : \Sigma^+ \rightarrow A$ that maps sequences of requests (the history of the interaction so far) to an action to be taken. In general, the algorithm induces an

infinite state space, as it may be in different states after processing different input sequences in Σ^*. Indeed, modeling of online algorithms by request-answer games gives rise to games with infinitely many positions [3]. For a finite set S of configurations, we say that g *uses memory* S, if there is a regular mapping of Σ^* into S such that g behaves in the same manner on identical continuations of words that are mapped to the same configuration.

We model the set of online algorithms that use memory S and solve an optimization problem P with requests in Σ and actions in A, by a WFA $\mathcal{A}_P = \langle \Sigma, S, \Delta, c, S_0, S \rangle$, such that Δ and c describe transitions between configurations and their costs, and S_0 is a set of possible initial configurations. Formally, $\Delta(s, \sigma, s')$ if the set $A' \subseteq A$ of actions that process the request σ from configuration s by updating the configuration to s' is non-empty, in which case $c(\langle s, \sigma, s' \rangle) = \min_{a \in A'} action\_cost(a)$. Note that all the states of $\mathcal{A}_P$ are accepting. Thus, $\mathcal{A}_P$ assigns a cost to all sequences in Σ^*.

Many optimization problems have online algorithms that require finite memory, or have finite memory variants that are obtained by imposing natural bounds. We give a few examples below.

Example 3.1 [**The paging problem** [21]] In the *paging* problem we have a two-level memory hierarchy: A slow memory that contains n different pages, and a *cache* that contains at most k different pages (typically, $k << n$). Pages that are in the cache can be accessed at zero cost. If a request is made to access a page that is not in the cache, the page should be brought into the cache, at a cost of 1, and if the cache is full, some other page should first be evicted from the cache. The paging problem is, given a sequence of requested pages, to decide which page to evict whenever an eviction is needed. The goal is to minimize the total cost.

A paging problem P with parameters n and k induces a WFA $\mathcal{A}_P = \langle \Sigma, S, \Delta, c, S_0, S \rangle$, where $\Sigma = \{1, \ldots, n\}$ is the set of possible requests (page indices), $S = \{C \subseteq \{1, \ldots, n\} : |C| \leq k\}$ is a set of finite configurations, each describing the set of pages currently in the cache, Δ and c describe how (and at which cost) requests are served, and $S_0 = \{\emptyset\}$, indicating that the cache is initially empty. Thus, $\Delta(C, i, C')$ iff one of the following holds: (1) $i \in C$, in which case $C' = C$ and $c(\langle C, i, C' \rangle) = 0$, (2) $i \notin C$, $|C| < k$, and $C' = C \cup \{i\}$, in which case $c(\langle C, i, C' \rangle) = 1$, or (3) $i \notin C$, $|C| = k$, and there is $j \in C$ such that $C' = (C \setminus \{j\}) \cup \{i\}$, in which case $c(\langle C, i, C' \rangle) = 1$. Note that by the definition of S, a configuration stores only the set of pages currently in the cache, and there are no provisions for storing any extra information such as time-stamps, etc. A different automaton for the problem could have defined S in a way that allows the storage of such extra information. We will elaborate on this point in the sequel.

Example 3.2 [**The k-server problem** [19]] The paging problem can be viewed as a special case of the k-server problem. There, we have k servers in a metric space $M = \langle V, d \rangle$, where V is a set of points and $d : V \times V \to \mathbb{R}^{\geq 0}$ is a distance function. The input to the problem is a sequence of points, each point should be served by moving a server to it (if no server is there), and the goal is to minimize the sum of distances that the servers move.

A k-server problem P with parameters k and $M = \langle V, d \rangle$, for a finite set V, induces a WFA $\mathcal{A}_P = \langle V, V^k, \Delta, c, \{s_0\}, V^k \rangle$, where each state corresponds to a configuration of the servers (for simplicity, we allow several servers to cover the same point), Δ and c describe how (and at which cost) servers may move, and s_0 is an initial configuration defined by the problem. Thus, $\Delta(s, v, s')$ iff one of the following holds: (1) there is $1 \leq j \leq k$ such that $v = s(j)$, in which case $s' = s$ and $c(\langle s, i, s' \rangle) = 0$, or (2) $s(j) \neq v$ for all $1 \leq j \leq k$, there is $1 \leq j \leq k$ such that $v = s'(j)$ and for all

$l \neq j$, we have $s'(l) = s(l)$, in which case $c(\langle s, i, s' \rangle) = d(s(j), s'(j))$[4]

Example 3.3 [**The ski-rental problem** [22]] In the ski-rental problem someone goes on a ski vacation whose length is not known in advance. Each morning he has to decide between renting skis ($1 per day) and buying skis ($\$y$). The goal is to minimize the expense. Here, making the problem finite-state requires the introduction of a finite bound M on the length of the vacation. Note that since M may be bigger than y, the challenge of an algorithm that knows M and does not know the length of the vacation in advance is similar to the challenge of an algorithm that does not know M. Indeed, studies of the problem usually refer to its finite-leasing version, in which the bound M is part of the input [4].

A ski-rental problem P with parameters y and M induces a WFA $\mathcal{A}_P = \langle \{a\}, \{0, \ldots, M+1\}, \Delta, c, \{0\}, \{0, \ldots, M+1\} \rangle$, where $\Delta(s, a, s')$ iff (1) $0 \leq s < M$ and $s' = s + 1$, in which case $c(\langle s, a, s' \rangle) = 1$, (2) $0 \leq s < M$ and $s' = M + 1$, in which case $c(\langle s, a, s' \rangle) = y$, or (3) $s = s' = M + 1$, in which case $c(\langle s, a, s' \rangle) = 0$. Note that the alphabet of $\mathcal{A}_P$ is a singleton letter, as we only care whether the vacation ends (the input word ends too) or not (the next letter is read).

Example 3.4 [**The load-balancing problem** [4]] In the load-balancing problem there are m identical machines. The input to the problem is a sequence $j_1, j_2, ..., j_n$ of loads from a domain J, typically $J = \mathbb{R}^{>0}$, each representing a load of a job that should be processed. The problem is to allocate the jobs to the machines, and the goal is to minimize the total load on the most loaded machine (a.k.a. *makespan*). Here too, we assume that there is a finite bound M on the total load of a machine, and that the set J of possible loads is finite. Let $\mathcal{J}$ denote the set of all possible sums of numbers from J that are bounded by M.

A load-balancing problem P with parameters J and M induces a WFA $\mathcal{A}_P = \langle J, \mathcal{J}^m, \Delta, c, 0^m, \mathcal{J}^m \rangle$, where each state $s \in \mathcal{J}^m$ describes a load-assignment to the m machines, and $\Delta(s, j, s')$ if there is $1 \leq i \leq m$ such that $s'(i) = s(i) + j$ and for all $l \neq i$, we have $s'(l) = s(l)$. The cost of $\langle s, j, s' \rangle$ is $\max_{1 \leq i \leq m} \{s'(i) - s(i)\}$.

Example 3.5 [**The Δ-paid exchange static list accessing problem** [4]] In this problem we have a static (fixed) linked list of n items. Each request is for an element of the list to be accessed. A request to access the i-th element in the list necessitates the traversal of i links, which costs i. After servicing a request, the list may be rearranged in the hope of better servicing future requests. Rearranging the list can be done by a series of exchanges of two consecutive items. Each exchange costs $\Delta > 1$.

While attempts to model the problem with *metric task systems* fail [4], it is not hard to see that P with parameters n and Δ induces a WFA $\mathcal{A}_P = \langle \Sigma, S, S \times \Sigma \times S, c, S_0, S \rangle$, where $\Sigma = \{1, \ldots, n\}$ and S is the set of all $n!$ permutations of $\{1, \ldots, n\}$, representing all the possible arrangements of the elements in the list. The cost of a transition $\langle s, i, s' \rangle$ is $j + \Delta k$, where j is the position of i in the permutation s, and k is the minimal number of exchanges needed to transform the list from the ordering s to the ordering s'.

We note that while the size of $\mathcal{A}_P$ is bounded by $|S|^2 \cdot |\Sigma|$, its computation may be complex, as demonstrated by Example 3.5. Note, however, that the source of the complexity is the fact that

[4]Note that both in the paging problem and here, we restrict attention to *lazy* algorithms, which minimize the change of configurations so that only the current request is served. By [19], for every non-lazy algorithm, there exists a lazy one that performs at least as well.

we compressed all the internal steps of the algorithm into one transition. Instead, one can enrich the alphabet of $\mathcal{A}_P$ and encode each request as a sequence of letters, thus allowing $\mathcal{A}_P$ to process each request by a series of internal steps, avoiding such compressions.

3.2 Relating online algorithms and determinization by pruning

In this section we reduce problems concerning online algorithms to questions about weighted automata. We first need some definitions. For two WFAs $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$ and $\mathcal{A}' = \langle \Sigma, Q, \Delta', c', Q'_0, F \rangle$, we say that $\mathcal{A}$ *embodies* $\mathcal{A}'$ if $Q'_0 \subseteq Q_0$, $\Delta' \subseteq \Delta$, and c' agrees with c on Δ'. Thus, $\mathcal{A}'$ can be obtained from $\mathcal{A}$ by decreasing its nondeterminism. For a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$ and an integer $r \geq 0$, the *r-refinement of $\mathcal{A}$* is the WFA $\mathcal{A}_r$ obtained by refining the state space of $\mathcal{A}$ by r Boolean variables. Formally, $\mathcal{A}_r = \langle \Sigma, Q \times 2^{\{1,\ldots,r\}}, \Delta_r, c_r, Q_0 \times 2^{\{1,\ldots,r\}}, F \times 2^{\{1,\ldots,r\}} \rangle$, where each state $\langle q, f \rangle \in Q \times 2^{\{1,\ldots,r\}}$ maintains, in addition to the state q of $\mathcal{A}$, also a subset f of $\{1, \ldots, r\}$, corresponding to a truth assignment for the new variables. The transition relation Δ_r and the cost function c_r are the expected extensions of Δ and c. That is, for every $f, f' \in 2^{\{1,\ldots,r\}}$, we have that $\Delta_r(\langle q, f \rangle, a, \langle q', f' \rangle)$ iff $\Delta(q, a, q')$, in which case $c_r(\langle \langle q, f \rangle, a, \langle q', f' \rangle \rangle) = c(\langle q, a, q' \rangle)$. Thus, each state of $\mathcal{A}$ has 2^r isomorphic copies in $\mathcal{A}_r$.

Definition 3.6 *Consider a WFA $\mathcal{A}$, an approximation factor $\alpha \geq 1$, and an integer $r \geq 0$. We say that $\mathcal{A}$ is* (α, r)-determinizable by pruning *((α, r)-DBP, for short) if the r-refinement of $\mathcal{A}$ embodies a DWFA that α-approximates $\mathcal{A}$.*

Note that when $\alpha = 1$, the embodied DWFA is equivalent to $\mathcal{A}$. Also, when $r = 0$, no refinement takes place, and the embodied automaton has the same state space as $\mathcal{A}$. When $\mathcal{A}$ is $(1, 0)$-DBP, we say that $\mathcal{A}$ is DBP.

Let P be an optimization problem, and let $\mathcal{A}_P = \langle \Sigma, S, \Delta, c, S_0, S \rangle$ be a WFA for its algorithms that use memory S. Given a finite sequence of requests $w \in \Sigma^*$, each run of $\mathcal{A}_P$ on w corresponds to a way of serving the requests in w by an algorithm with configurations in S. The set of all runs include all such algorithms, thus the cost of w in $\mathcal{A}_P$ is the cost of w in an optimal offline algorithm that uses memory S. Indeed, the semantics of WFA over the tropical semiring, in which the cost of a word is the minimum cost of some run on it, guarantees that the cost would be calculated according to the best guess. On the other hand, an online algorithm has to process each request as soon as it arrives, without knowing the requests yet to arrive. Accordingly, an online algorithm that uses memory S corresponds to a DWFA embodied in $\mathcal{A}_P$. Indeed, for every configuration $s \in S$ of the problem and request $\sigma \in \Sigma$, the algorithm suggests a particular way to process σ from s, inducing a particular transition $\langle s, \sigma, s' \rangle \in \Delta$. Moreover, a refinement of $\mathcal{A}_P$ maintains the correspondence between its transitions and the actions of the algorithms (note that this correspondence is lost if we consider unrestricted determinization of $\mathcal{A}_P$). Hence, a DWFA embodied in a refinement of $\mathcal{A}_P$ corresponds to an online algorithm with an extended memory. Formally, we have the following.

Theorem 3.7 *Consider an online problem P and a set S of configurations. Let $\mathcal{A}_P$ be a WFA with state space S that models online algorithms for P that use memory S. For all $\alpha \geq 1$ and $r \geq 0$, the problem P has competitive ratio (α, r) with memory S iff $\mathcal{A}_P$ is (α, r)-DBP.*

4 Determinization and Approximated Determinization by Pruning

In this section we study the problem of determinization by pruning. We show that deciding whether a given WFA is DBP (the *DBP problem*, for short) can be done in polynomial time. On the other hand, deciding whether a given WFA is $(\alpha, 0)$-DBP, for $\alpha > 1$ (the *approximated DBP problem*, for short) is NP-complete. In both cases, when the answer is positive, returning a witness DWFA requires no extra cost.

We assume that a given WFA $\mathcal{A}$ has no useless states (that is, every state is reachable from at least one initial state, and at least one word is accepted from each state; otherwise we remove the state and its associated transitions).

4.1 Deciding determinization by pruning

The polynomial-time algorithm for the DBP problem is our most challenging technical result. For clarity, we first describe a polynomial-time algorithm for deciding whether a given NFA (that is, a WFA with no costs) is DBP (that is, embodies an equivalent DFA).

Theorem 4.1 *The DBP problem for NFAs can be solved in polynomial time.*

Proof: We describe a polynomial algorithm for solving DBP for NFA. The algorithm decides whether a given NFA is DBP, and in case the answer is positive, it also returns a description of the set of equivalent embodied DFAs.

Consider an NFA $\mathcal{A} = \langle \Sigma, Q, \Delta, Q_0, F \rangle$. We inductively define a sequence $H_0, H_1, \ldots \subseteq Q \times Q$ of relations as follows.

$$
\begin{aligned}
H_0 &= (F \times F) \cup (\ (Q \setminus F) \times Q), \text{ and for } i \geq 0, \\
H_{i+1} &= H_i \cap \{\langle q, q' \rangle : \text{for all } a \in \Sigma, \text{ if } \delta(q,a) \neq \emptyset, \text{ then there exists } v' \in \delta(q',a) \\
&\qquad \text{such that for all } v \in \delta(q,a) \text{ we have } H_i(v, v')\}.
\end{aligned}
$$

Intuitively, $H_i(q, q')$ means that there is a DFA $\mathcal{A}'$ embodied in $\mathcal{A}$ such that all the words of length at most i accepted from q in $\mathcal{A}$ are also accepted from q' in $\mathcal{A}'$. Since $H_0 \supseteq H_1 \supseteq H_2 \supseteq \ldots$, the sequence of relations eventually reaches a fixed-point, which we denote by H. For two states q and q', we say that q' *covers* q if $H(q, q')$. The relation H induces an NFA $\mathcal{A}^H = \langle \Sigma, Q, \Delta^H, Q_0^H, F \rangle$ embodied in $\mathcal{A}$, where $q_0 \in Q_0^H$ iff $q_0 \in Q_0$ and q_0 covers all the states in Q_0, and for every $q, v \in Q$ and $a \in \Sigma$, we have that $\Delta^H(q, a, v)$ iff $\Delta(q, a, v)$ and v covers all the states in $\delta(q, a)$. Note that the set Q_0^H may be empty, and that for some q and a it may be that $\delta^H(q, a) = \emptyset$ even though $\delta(q, a) \neq \emptyset$. We prove below that $Q_0^H \neq \emptyset$ iff $\mathcal{A}$ is DBP. We first prove that the relation H is transitive.

Lemma 4.2 *The relation H is transitive. That is, for $q, q', q'' \in Q$, if $H(q, q')$ and $H(q', q'')$ then $H(q, q'')$.*

Proof: We prove that for all $i \geq 0$, if $H_i(q, q')$ and $H_i(q', q'')$ then $H_i(q, q'')$. The proof proceeds by an induction on i.

First, if $q \in F$ then $q' \in F$, in which case $q'' \in F$, thus H_0 is transitive. Assume now that H_i is transitive, and let $q, q', q'' \in Q$ be such that $H_{i+1}(q, q')$ and $H_{i+1}(q', q'')$. We prove that $H_{i+1}(q, q'')$.

Let $a \in \Sigma$ be such that $\delta(q,a) \neq \emptyset$. Since $H_{i+1}(q,q')$, there exists a state $q'_a \in \delta(q',a)$ such that $H_i(q_a, q'_a)$ for all $q_a \in \delta(q,a)$. Since $H_{i+1}(q',q'')$ and $q'_a \in \delta(q',a)$, there exists a state $q''_a \in \delta(q'',a)$ such that $H_i(q'_a, q''_a)$. By the induction hypothesis, $H_i(q_a, q''_a)$ for all $q_a \in \delta(q,a)$. Since the above holds for all letters $a \in \Sigma$, it follows that $H_{i+1}(q,q'')$. □

Lemma 4.3 *The NFA $\mathcal{A}$ is DBP iff $Q_0^H \neq \emptyset$.*

Proof: Assume first that $\mathcal{A}$ is DBP. Let $\mathcal{A}' = \langle \Sigma, Q', \Delta', q'_0, F' \rangle$ be an equivalent DFA embodied in $\mathcal{A}$. We first prove (1) that for all words $w \in \Sigma^*$, if $\delta(Q_0, w) \neq \emptyset$, then there exist a state $q' = \delta'(q'_0, w)$. We then prove (2) that for such a word w, the corresponding state q' covers all the states in $\delta(Q_0, w)$. In particular, $\delta(q'_0, \epsilon) = q'_0$ covers all the states in $\delta(Q_0, \epsilon) = Q_0$, thus $q'_0 \in Q_0^H$.

In order to prove (1), let us recall that by the assumption at the beginning of this section, no state in $\mathcal{A}$ is empty. Let w be a word on which $\mathcal{A}$ has a run, let $q \in \delta(Q_0, w)$, and let z be a word accepted from q by $\mathcal{A}$; thus $w \cdot z \in L(\mathcal{A})$. Since $\mathcal{A}'$ is equivalent to $\mathcal{A}$, it must accept the word $w \cdot z$. Let r' be the accepting run of $\mathcal{A}'$ on $w.z$. Clearly, there is a prefix of r', which is a run of $\mathcal{A}'$ on w.

In order to prove (2), we prove that for all $i \geq 0$, for any word $w \in \Sigma^*$, if $q \in \delta(Q_0, w)$ and $q' = \delta'(q'_0, w)$, then $H_i(q, q')$. The proof proceeds by an induction on i.

For the induction base, recall that $H_0(q,q')$ iff $q \in F$ implies that $q' \in F$. Consider a state $q \in \delta(Q_0, w)$, and assume that $q \in F$. Then, $w \in L(\mathcal{A})$. Since $\mathcal{A}$ and $\mathcal{A}'$ are equivalent, and $\mathcal{A}'$ is deterministic, it must be that $\delta'(q'_0, w)$ is in F. Thus, $H_0(q, \delta'(q'_0, w))$.

For the induction step, we assume that for all words $w \in \Sigma^*$ and all states $q \in \delta(Q_0, w)$ the state $q' = \delta'(q'_0, w)$ is defined and satisfies $H_i(q, q')$, and prove that $H_{i+1}(q, q')$. Observe that in order to prove that $H_{i+1}(q, q')$ it is enough to prove that for every letter $a \in \Sigma$, the state $v' = \delta'(q', a)$ is such that for all $v \in \delta(q, a)$, we have $H_i(v, v')$. Note that $v' = \delta'(q'_0, w \cdot a)$, and consider a state $v \in \delta(q, a)$. Since $v \in \delta(Q_0, w \cdot a)$, then by the induction hypothesis, applied to the word $w \cdot a$, we have that $H_i(v, v')$, and we are done.

For the other direction, assume that Q_0^H is not empty. We claim that every maximal DFA that is embodied in $\mathcal{A}^H$ is equivalent to $\mathcal{A}$ (an embodied DFA is maximal if adding to it a transition would make it nondeterministic). Let $\mathcal{A}' = \langle \Sigma, Q, \delta', q'_0, F \rangle$ be such a DFA. Thus, $q'_0 \in Q_0^H$ and for all states $q \in Q$ and letters $a \in \Sigma$, we have $\delta'(q, a) \in \delta^H(q, a)$. We prove that $L(\mathcal{A}') = L(\mathcal{A})$.

Since $\mathcal{A}'$ is embodied in $\mathcal{A}^H$, which in turn is embodied in $\mathcal{A}$, it is clear that $L(\mathcal{A}') \subseteq L(\mathcal{A})$. In order to prove that $L(\mathcal{A}) \subseteq L(\mathcal{A}')$, we consider a word $w = w_1 w_2 \ldots w_n \in L(\mathcal{A})$ and prove that for every run $r = r_0 r_1 \ldots r_n$ of $\mathcal{A}$ on w, there is a run $s = s_0 s_1 \ldots s_n$ of $\mathcal{A}'$ on w, and that for all $0 \leq i \leq n$, we have $H(s_i, s_i)$ and $H(r_i, s_i)$. Since $H \subseteq H_0$, the latter implies that membership of r_n in F implies membership of s_n in F. Thus, if there is an accepting run r of $\mathcal{A}$ on w, then the run of $\mathcal{A}'$ on w is also accepting. The proof proceeds by an induction on i.

For $i = 0$, the construction of $\mathcal{A}'$ implies that $s_0 = q'_0 \in Q_0^H$. Therefore, by the definition of Q_0^H, the state s_0 covers all the states in Q_0, and in particular it covers r_0 and itself. For the induction step, assume that the induction hypothesis holds for $0 \leq i \leq n-1$. Since $r_{i+1} \in \delta(r_i, w_{i+1})$ then $\delta(r_i, w_{i+1}) \neq \emptyset$. Hence, since by the induction hypothesis s_i covers r_i. Since H is a fixed-point, then there exists a state $t \in \delta(s_i, w_{i+1})$ such that t covers all the states in $\delta(r_i, w_{i+1})$, and r_{i+1} among them. Similarly, since $\delta(s_i, w_{i+1}) \neq \emptyset$, and by the induction hypothesis s_i covers itself, then there exists a state $v' \in \delta(s_i, w_{i+1})$ such that v' covers all states in $\delta(s_i, w_{i+1})$. Thus, by the definition of Δ^H, the set $\delta^H(s_i, w_{i+1}) \neq \emptyset$. Since $\mathcal{A}'$ is maximal, there is a state $s_{i+1} \in \delta^H(s_i, w_{i+1})$, and we can

extend the run of $\mathcal{A}'$ to cover $w_1 w_2 \ldots w_{i+1}$. Since $s_{i+1} \in \delta^H(s_i, w_{i+1})$, it follows that s_{i+1} covers t and itself. Recall that t covers r_{i+1}, thus by the transitivity of H (Lemma 4.2), s_{i+1} covers r_{i+1}, and we are done. □

By Lemma 4.3, the problem of checking whether a given NFA $\mathcal{A}$ is DBP can be reduced to calculating H and checking whether there is a state $q_0 \in Q_0$ that covers all the states in Q_0.

Computing H_0 takes $O(|Q|^2)$ time, and this is also the upper bound on the size of H_0. To compute H_i from H_{i-1} we need to check for all pairs $\langle q, q' \rangle \in H_{i-1}$ and for all letters $a \in \Sigma$ whether there exists an a-successor of q' that covers all a-successors of q. The number of successors of both q and q' is bounded by $|\Delta|$. Thus, the number of required checks is bounded by $O(|H_{i-1}| \cdot |\Delta|^2) = O(|Q|^2 \cdot |\Delta|^2)$. The number of iterations executed until the fixed point is reached is bounded by the size of H_0. Thus, the overall number of checks in the whole computation is bounded by $O(|Q|^4 \cdot |\Delta|^2)$. This is clearly polynomial in the size of the input. □

Note that, like the algorithm for DFA minimization, our algorithm calculates a fixed-point over pairs of states. The fixed-point here, however, is different and more complicated, as it involves a universal requirement nested inside an existential requirement. We found the result to be quite surprising. Indeed, as we now show, a slightly different decision problem, which maintains the local flavor of determinization by pruning, is NP-hard. We say that an WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$ is *almost-deterministic* if for every $q \in Q$ and $a \in \Sigma$ we have $|\delta(q, a)| \leq 1$. Thus, unlike deterministic automata, $\mathcal{A}$ may have several initial states. The *almost-DBP* problem is then to decide, given a WFA, whether it embodies an equivalent almost deterministic automaton. Note that this amounts to asking whether the nondeterministic choices of $\mathcal{A}$ can be replaced by an initial choice, among finitely many options. In the context of online algorithms, it means there are finitely many online algorithms such that, for each input sequence, one of the algorithms performs as well as the offline algorithm. We now show that the almost-DBP problem is NP-hard already in the unweighted case.

Theorem 4.4 *The almost-DBP problem for NFAs is NP-hard.*

Proof: We describe a reduction from 3SAT to almost-DBP. Let θ be a 3CNF formula with m clauses over the variables $x_1, x_2, ..., x_n$. We construct an NFA $\mathcal{A}_\theta$ over the alphabet $\{0, 1, ..., m\}$, such that $\mathcal{A}_\theta$ is almost-DBP iff θ is satisfiable.

The NFA $\mathcal{A}_\theta$ has the form of a DAG with three levels. On the first level there are n initial states, corresponding to the n variables in θ. On the second level there are $2n$ states. Each variable x_i induces two states: i_{true} and i_{false}, corresponding to the two possible truth assignments to x_i. For each $1 \leq i \leq n$, there are transitions labeled 0 from the initial state i to both i_{true} and i_{false}. On the third level, there is a single accepting state. For each state i_{val} in the second level and letter $1 \leq j \leq m$, there is a transition labeled j from i_{val} to the accepting state iff assigning val to variable i satisfies clause j. For example, if the literal $\neg x_5$ appears in clause 2, then there is a transition labeled 2 from the state 5_{false} to the accepting state. The language of $\mathcal{A}_\theta$ is $\{0 \cdot j : 1 \leq j \leq m\}$. In Figure 1 we show the NFA corresponding to the formula $\theta = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_2)$.

We prove that θ is satisfiable iff $\mathcal{A}_\theta$ is almost-DBP. Assume first that θ is satisfiable. Let $f : \{1, ..., n\} \rightarrow \{\text{true}, \text{false}\}$ be a satisfying assignment to the variables of θ. We describe an almost-deterministic automaton $\mathcal{A}_\theta^f$ embodied in $\mathcal{A}_\theta$ such that the language of $\mathcal{A}_\theta^f$ contains the language of

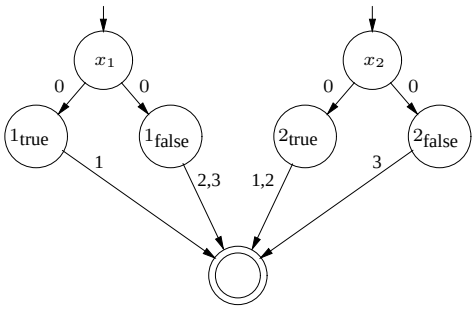

Figure 1: The NFA corresponding to $\theta = (x_1 \vee x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_2)$.

$\mathcal{A}_\theta$. Since $\mathcal{A}^f_\theta$ is embodied in $\mathcal{A}_\theta$, the containment in the other direction is trivial. Note that the only nondeterminism we have to resolve is in the transitions from each state i to i_{true} and i_{false}. Intuitively, each such choice corresponds to a truth assignment to x_i, and we let f chose. Formally, the transition function δ^f of $\mathcal{A}^f_\theta$ agrees with the transition function δ_θ of $\mathcal{A}$, except that for $1 \leq i \leq n$, we have $\delta^f(i, 0) = i_{f(i)}$. Since f is a satisfying assignment, then for every $1 \leq j \leq m$ there is a variable i_j whose assignment satisfies the clause j. Accordingly, there is an accepting run of $\mathcal{A}^f_\theta$ on the word $0 \cdot j$, starting at the initial state i_j.

For the other direction, assume that $\mathcal{A}_\theta$ embodies an equivalent almost-deterministic automaton $\mathcal{A}' = \langle \Sigma, Q, \delta', Q_0, F \rangle$. Let $f : \{1, ..., n\} \to \{\text{true}, \text{false}\}$ be such that $f(i) = \text{true}$ iff $\delta'(i, 0) = i_{\text{true}}$. Since $\mathcal{A}'$ is equivalent to $\mathcal{A}$, then for every $1 \leq j \leq m$, the word $0 \cdot j$ is accepted in $\mathcal{A}'$. By the definition of $\mathcal{A}_\theta$, there is a transition labeled j from a state i_{val} to the accepting state iff assigning val to i satisfies the clause j. Hence, by the definition of f, the truth assignment f satisfies θ. □

We now move to the DBP problem for WFA. Like the algorithm in the unweighted case, the polynomial algorithm we present below is based on a fixed-point calculation, that for each pair $\langle q, q' \rangle$ of states of $\mathcal{A}$, compares the behavior of embodied deterministic automata with initial state q' to the behavior of the nondeterministic automaton $\mathcal{A}^q$, over words of increasing length. The setting here, however, is much more difficult. First, the characterization associated with each pair is not Boolean: it is not enough to remember whether one can deterministically accept from q' the same words as from q — the characterization has to further refine this information and refer to the possibly different costs involved. Second, while in the unweighted setting it is clear that the calculation would reach a fixed-point in a polynomial number of steps, in the weighted setting it may well be that as the length of the words considered increases, so does the cost difference, and it is not clear how to force the calculation to reach a fixed-point.

Theorem 4.5 *The DBP problem can be solved in polynomial time.*

Proof: We first need some notations. For every $r \in \mathbb{R}$ we have $-\infty < r < \infty$, and we allow expressions of the form $\infty \pm r$, $-\infty \pm r$, $\infty + \infty$, and $(-\infty) + (-\infty)$, with the usual meaning. For every $i \geq 0$, let $\Sigma^{\leq i} = \{w \in \Sigma^* : |w| \leq i\}$. Let $\mathcal{A}$ and $\mathcal{A}'$ be two WFAs over the same alphabet Σ. Given a subset $S \subseteq \Sigma^*$, we define the cost difference between $\mathcal{A}$ and $\mathcal{A}'$ over S to be *costdiff*$(\mathcal{A}', \mathcal{A}, S) = \sup_{w \in S \cap L(\mathcal{A})}[cost(\mathcal{A}', w) - cost(\mathcal{A}, w)]$. Note that if $S \cap L(\mathcal{A}) = \emptyset$ then *costdiff*$(\mathcal{A}', \mathcal{A}, S) = -\infty$,

and that if there is a word $w \in S \cap L(\mathcal{A}) \setminus L(\mathcal{A}')$ then *costdiff*$(\mathcal{A}', \mathcal{A}, S) = \infty$. Also note that, unless both WFAs accept the empty language, $\mathcal{A}$ and $\mathcal{A}'$ are equivalent iff *costdiff*$(\mathcal{A}', \mathcal{A}, \Sigma^*) = 0$ and *costdiff*$(\mathcal{A}, \mathcal{A}', \Sigma^*) = 0$. Let $det(\mathcal{A})$ be the set of all deterministic WFAs embodied in $\mathcal{A}$, and let $det_0(\mathcal{A})$ be the set of all WFAs $\mathcal{A}'$ in $det(\mathcal{A})$ such that *costdiff*$(\mathcal{A}', \mathcal{A}, \Sigma^*) = 0$. Hence, $det_0(\mathcal{A})$ is exactly the set of all deterministic automata embodied in $\mathcal{A}$ that are equivalent to $\mathcal{A}$.

Consider a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$. Let $n = |Q|$. Our algorithm calculates a sequence of functions $f_0, f_1, \ldots, f_{2n^2-1} : Q \times Q \to \mathbb{R} \cup \{-\infty, \infty\}$, such that the following holds.

- For $0 \leq i \leq n^2 - 1$, the function $f_i(q, q')$ measures how well the state q' can deterministically simulate the state q, over words of length at most i. Formally, for every $\mathcal{A}' \in det_0(\mathcal{A})$ (if exists), and every pair of states $q, q' \in Q$ such that q' is reachable in $\mathcal{A}'$, we have that $f_i(q, q') =$ *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i})$. The value $-\infty$ is assigned to $f_i(q, q')$ when there are no words in $\Sigma^{\leq i}$ that can be accepted from q, and the value ∞ is assigned when there is a word that can be accepted from q but not from q'.
- For $n^2 \leq i \leq 2n^2 - 1$, the function $f_i(q, q')$ is similar, only that it takes cycles into account, and maps to ∞ pairs for which the cost difference has not stabilized yet, which indicates that it cannot be bounded.

The sequence of functions $f_0, f_1, \ldots, f_{2n^2-1}$ is defined as follows.

- At initialization:
$$f_0(q, q') = \begin{cases} -\infty & \text{if } q \notin F \\ 0 & \text{if } q \in F \text{ and } q' \in F \\ \infty & \text{if } q \in F \text{ and } q' \notin F, \end{cases}$$
- For $1 \leq i \leq n^2 - 1$:
$$f_i(q, q') = \max\{f_{i-1}(q, q'), \max_{a \in \Sigma} f_i(q, q', a)\}.$$
- For $n^2 \leq i \leq 2n^2 - 1$:
$$f_i(q, q') = \begin{cases} \infty & \text{if } \max_{a \in \Sigma} f_i(q, q', a) > f_{i-1}(q, q') \\ f_{i-1}(q, q') & \text{otherwise.} \end{cases}$$

In the above, for every $1 \leq i \leq 2n^2 - 1$ and $a \in \Sigma$, the function $f_i(q, q', a)$ is defined as follows.

$$f_i(q, q', a) = \min_{u' \in \rho_{i-1}(q', a)} \max_{u \in \delta(q, a)} f_{i-1}(u, u') + c(q', a, u') - c(q, a, u),$$

where the set $\rho_0(q', a) = \delta(q', a)$, and for $1 \leq i \leq 2n^2 - 1$, we have

$$\rho_i(q', a) = \{u' \in \rho_{i-1}(q', a) : \max_{u \in \delta(q', a)} [f_{i-1}(u, u') + c(q', a, u') - c(q', a, u)] \leq 0\}.$$

In the expression above for $f_i(q, q', a)$, in case that for all $u \in \delta(q, a)$ we have that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} = \emptyset$, we set $f_i(q, q', a) = -\infty$ (note that this also covers the case $\delta(q, a) = \emptyset$). In case there is $u \in \delta(q, a)$ such that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} \neq \emptyset$ and $\rho_{i-1}(q', a) = \emptyset$, we set $f_i(q, q', a) = \infty$.

Intuitively, a state $u' \in \rho_i(q', a)$ is a "witness" to the fact that $f_i(q', q', a) \leq 0$, i.e., to the fact that q' can deterministically simulate itself over words in $\Sigma^{\leq i}$ that start with a. Clearly, only such witnesses can be a-successors of q' in an embodied DWFA that is equivalent to $\mathcal{A}$.

We argue that the sequence of functions reaches a fixed-point in some iteration $1 \leq j \leq 2n^2 - 1$, and that $\mathcal{A}$ is DBP iff there is a state $q'_0 \in Q_0$ such that for every $q_0 \in Q_0$, we have that $f_j(q_0, q'_0) \leq 0$. Also, in case $\mathcal{A}$ is DBP, then every DWFA that consists of transitions that use the witnesses from the last iteration (that is, whose transition relation assigns successors according to ρ_j) is equivalent to $\mathcal{A}$. A detailed proof can be found in Appendix A. □

4.2 Deciding approximated determinization by pruning

We now turn to the approximated-DBP problem, and show that it is much harder. We first study the problem of approximation of a WFA by a given embodied DWFA.

Lemma 4.6 *Consider a WFA $\mathcal{A}$, an embodied DWFA $\mathcal{A}'$, and an approximation factor $\alpha \geq 1$. Deciding whether $\mathcal{A}'$ α-approximates $\mathcal{A}$ can be done in polynomial time.*

Proof: Let $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$ and $\mathcal{A}' = \langle \Sigma, Q', \Delta', c', q'_0, F' \rangle$. Consider first the case $\alpha = 1$. Then, the algorithm is similar to the one used for checking whether $\mathcal{A}$ is DBP, only that now $\mathcal{A}'$ is given. Accordingly, the functions f_i are defined for pairs in $Q \times Q'$, and when calculating $f_i(q, q', a)$, we only have to consider the given (if any) a-successor of q' in $\mathcal{A}'$, instead of the set $\rho_{i-1}(q', a)$. That is, $f_i(q, q', a)$ becomes: $f_i(q, q', a) = \max_{u \in \delta(q,a)} \{f_{i-1}(u, \delta'(q', a)) + c(q', a, \delta'(q', a)) - c(q, a, u)\}$.

Now, given $\alpha > 1$, we further modify the algorithm by scaling all the edges of $\mathcal{A}$ by α. More formally, we define a sequence of functions $g_0, g_1, \ldots, g_{2n^2-1} : Q \times Q' \to \mathbb{R} \cup \{-\infty, \infty\}$ that is similar to the sequence f_i, except that for every $1 \leq i \leq 2n^2 - 1$ and $a \in \Sigma$, we have

$$g_i(q, q', a) = \max_{u \in \delta(q,a)} [g_{i-1}(u, \delta'(q', a)) + c(q', a, \delta'(q', a)) - \alpha \cdot c(q, a, u)].$$

We argue that the sequence of functions reaches a fixed-point in some iteration $1 \leq j \leq 2n^2 - 1$, and that $\mathcal{A}'$ α-approximates $\mathcal{A}$ iff for every $q_0 \in Q_0$, we have that $g_j(q_0, q'_0) \leq 0$. The proof of this argument is essentially the same as the proof of Theorem 4.5 found in Appendix A, with the obvious modification that any reference to an automaton embodied in $\mathcal{A}$ is replaced with the given automaton $\mathcal{A}'$, and any reference to $\rho_i(q', a)$ or $\rho(q', a)$ is replaced with $\delta'(q', a)$.

It is worth noting that in order to also handle an additive factor $\beta > 0$, that is, in order to check if for all words w accepted by $\mathcal{A}$, we have $cost(\mathcal{A}, w) \leq \alpha \cdot cost(\mathcal{A}, w) + \beta$, all we have to do is to check whether for every $q_0 \in Q_0$, we have that $g_j(q_0, q'_0) \leq \beta$. □

Before we use Lemma 4.6 for solving the approximated DBP problem, we note its application in reasoning about online algorithms. Indeed, by Theorem 3.7, we have the following.

Corollary 4.7 *Consider an optimization problem P and a finite set S of configurations. Given an online algorithm g with memory S, and a competitive ratio $\alpha \geq 1$, the problem of deciding whether g is α-competitive with respect to an offline algorithm with memory S can be solved in polynomial time.*

In light of Lemma 4.6, one may be tempted to believe that by using the same ideas one can extend Theorem 4.5 to handle an approximation factor $\alpha > 1$. Unfortunately, as the next theorem shows, unless P=NP, this can not be the case. Essentially, the property that if $u' \in \rho_{i-1}(q', a)$ then u' is such that for every $q \in Q$ the minimum in the expression for $f_i(q, q', a)$ is achieved with u', which is crucial to proving Theorem 4.5, is no longer true when $\alpha > 1$.

Theorem 4.8 *The approximated-DBP problem is NP-complete.*

Proof: Membership in NP follows from Lemma 4.6. We prove NP-hardness by a reduction from 3-SAT. Given $\alpha > 1$ and a 3-SAT formula $\theta = \bigwedge_{j=1}^{m} C_j$ over the variables $x_1, \ldots, x_n$, we build a WFA $\mathcal{A}$ that is $(\alpha, 0)$-DBP iff θ is satisfiable. We assume without loss of generality that no clause in θ contains both a variable and its negation. The alphabet of $\mathcal{A}$ is $\Sigma = \{a\} \cup \{C_1, \ldots, C_m\}$, and $\mathcal{A}$ is given in Figure 2.

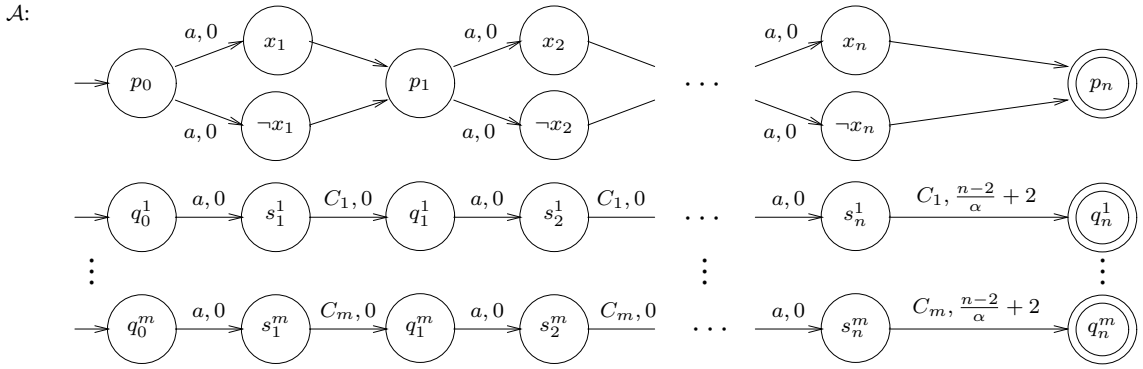

Figure 2: A WFA for a 3-SAT formula.

For every $1 \leq i \leq n$ and every literal $l_i \in \{x_i, \neg x_i\}$, the edge between l_i and p_i (which for lack of space is unlabeled in the figure) stands for m transitions, one for each of the letters $C_1, \ldots, C_m$, and the cost of a transition $\langle l_i, C_j, p_i \rangle$ is α if the literal $\neg l_i$ appears in the clause C_j, and is 1 otherwise. Recall that no clause in θ contains both a variable and its negation. Hence, for every $1 \leq i \leq n$ and every $1 \leq j \leq m$, at least one of the transitions $\langle x_i, C_j, p_i \rangle$, and $\langle \neg x_i, C_j, p_i \rangle$ costs 1. It follows that $\mathcal{A}$ with initial state p_0 accepts exactly all words of the form $(a \cdot (C_1 + \ldots + C_m))^n$ with cost n. In addition, $\mathcal{A}$ has m components such that for every $1 \leq j \leq m$, the DWFA $\mathcal{A}$ with initial state q_0^j accepts the word $(aC_j)^n$ with a lower cost (recall that $\alpha > 1$) of $(n-2)/\alpha + 2$. In the remainder of the proof we refer to words of the form $(aC_j)^n$ as *single-clause* words.

We now show that $\mathcal{A}$ is $(\alpha, 0)$-DBP iff θ is satisfiable. Observe that if $\mathcal{A}'$ is a deterministic automaton embodied in $\mathcal{A}$ such that $\mathcal{A}'$ accepts all the words that $\mathcal{A}$ accepts, then $\mathcal{A}'$ must have p_0 as its initial state, and for every $1 \leq i \leq n$ exactly one of the transition (p_{i-1}, a, x_i), and $(p_{i-1}, a, \neg x_i)$ is present in $\mathcal{A}'$. Indeed, $\mathcal{A}'$ induces an assignment to the variables $x_1, \ldots, x_n$, where x_i is true if $\mathcal{A}'$ has the transition $\langle p_{i-1}, a, x_i \rangle$, and false if it has the transition $\langle p_{i-1}, a, \neg x_i \rangle$. For every $1 \leq i \leq n$ let $l_i' \in \{x_i, \neg x_i\}$ be such that the transition $\langle p_{i-1}, a, l_i' \rangle$ is in $\mathcal{A}'$. Since the cost of all reachable transitions in $\mathcal{A}'$ on the letters $C_1, \ldots C_m$ is at most α, and a-transitions cost 0, we have that $\mathcal{A}'$ accepts every word of the form $(a \cdot (C_1 + \ldots + C_m))^n$ with cost at most αn. Thus, if w is not a single-clause word we have that $cost(\mathcal{A}', w) \leq \alpha \cdot cost(\mathcal{A}, w)$.

It remains to show that for every single-clause word $w = (aC_j)^n$, we have that $cost(\mathcal{A}', w) \leq \alpha \cdot cost(\mathcal{A}, w)$ iff the assignment induced by $\mathcal{A}'$ satisfies C_j. To see that, observe that the (single) run of $\mathcal{A}'$ on w is $p_0 \cdot l'_1 \cdot p_1 \cdots l'_n \cdot p_n$, and that for every $1 \leq i \leq n$ we have that $\neg l'_i$ is false in the assignment induced by $\mathcal{A}'$. Recall that the transition (l'_i, C_j, p_i) costs t if $\neg l'_i$ appears in C_j, but costs 1 otherwise. Let false(j) be the number of literals in C_j that were assigned the value false by $\mathcal{A}'$. It follows that $cost(\mathcal{A}', w) = n - \text{false}(j) + \text{false}(j)\alpha$. Since θ is a 3-SAT formula we have that false$(j) \leq 2$ iff $\mathcal{A}'$ satisfies C_j. Hence, for every $\alpha > 1$ we have that $cost(\mathcal{A}', w) \leq n - 2 + 2\alpha = \alpha \cdot cost(\mathcal{A}, w)$ iff $\mathcal{A}'$ satisfies C_j. □

We note that an adjustment to the costs of the transitions of the WFA used in the proof of Theorem 4.8 shows that the approximated-DBP problem is NP-hard already for an additive approximation factor (that is, when the embodied DWFA $\mathcal{A}'$ is such that there is $\beta \geq 0$ such that for all $w \in \Sigma^*$, we have $cost(\mathcal{A}', w) \leq \beta + cost(\mathcal{A}, w)$). The adjustment required is to assign a cost of $1 + \beta/n$ to the transitions that currently cost α, and a cost of $n + \beta(2/n - 1)$ to the transitions that currently cost $(n-2)/\alpha + 2$. A combination of a multiplicative factor α and an additive factor β, can be handled in the obvious way.

By Theorem 3.7 (and the fact that every WFA induces an online algorithm), we can conclude with the following.

Corollary 4.9 *Consider an optimization problem P and a finite set S of configurations. The problem of deciding whether P has competitive ratio $(\alpha, 0)$ with memory S can be solved in polynomial time for $\alpha = 1$, and is NP-complete for $\alpha > 1$.*

5 Determinization and Approximated Determinization by Refinement and Pruning

In this section we study the problem of determinization by refinement and pruning. We first show that extension of the memory is hopeless in an effort to be as good as an optimal offline algorithm.

Theorem 5.1 *For all integers $r \geq 0$, a WFA $\mathcal{A}$ is $(1, r)$-DBP iff it is $(1, 0)$-DBP.*

Proof: Clearly, if $\mathcal{A}$ is $(1, 0)$-DBP, then it is also $(1, r)$-DBP. We prove that if $\mathcal{A}$ is $(1, r)$-DBP for some $r \geq 0$, then it is also $(1, 0)$-DBP. For a refinement $\mathcal{A}_r$ of $\mathcal{A}$, we say that a DWFA $\mathcal{D}_r$, obtained by pruning $\mathcal{A}_r$, is *simple* if for each state q of $\mathcal{A}$ there is at most one subset $f \subseteq \{1, \ldots, r\}$ such that the state $\langle q, f \rangle$ is reachable in $\mathcal{D}_r$. If $\mathcal{A}_r$ can be pruned to a simple equivalent $\mathcal{D}_r$, then by omitting the $2^{\{1,\ldots,r\}}$ element of each state we get an equivalent deterministic pruning of $\mathcal{A}$, and we are done.

Assume now that no simple equivalent pruning exists; i.e., in every equivalent DWFA $\mathcal{D}_r$ that is obtained by pruning $\mathcal{A}_r$, there exists a state q and two subsets $f_1, f_2 \in 2^{\{1,\ldots,r\}}$, such that both $\langle q, f_1 \rangle$ and $\langle q, f_2 \rangle$ are reachable. Then, there must be a word t_1, accepted from $\langle q, f_1 \rangle$ with a certain cost and from $\langle q, f_2 \rangle$ with a higher cost (or not at all). Indeed, otherwise, we could have directed transitions that go to $\langle q, f_1 \rangle$ into $\langle q, f_2 \rangle$ and get an equivalent DWFA in which $\langle q, f_1 \rangle$ is unreachable. This change would not make $\mathcal{D}_r$ accept more words or accept some words with a different cost, since both the languages accepted from $\langle q, f_1 \rangle$ and from $\langle q, f_2 \rangle$ are contained in the language accepted from q in $\mathcal{A}$. Doing this repeatedly would result in a simple pruning.

Let h_2 be a word such that in $\mathcal{D}_r$ the state $\langle q, f_2 \rangle$ is reachable from the initial state by h_2. Clearly, in $\mathcal{A}$, the state q is reachable from the set of initial states with the word h_2 and the language of $\mathcal{A}$ from state q contains t_1 (with a cost equal to the acceptance cost of t_1 from $\langle q, f_1 \rangle$, or lower). Thus, $h_2 \cdot t_1 \in L(\mathcal{A})$ (with a cost that is lower than, or equal to, the sum of the weights of the transitions in a path of minimal cost from the set of initial states to q plus the cost of accepting t_1 from $\langle q, f_1 \rangle$). However, in $\mathcal{D}_r$, the word $h_2 \cdot t_1$ is accepted with a higher cost (or not accepted at all), and we have reached a contradiction. □

On the other hand, an extension of the memory may help in achieving a better competitive ratio:

Theorem 5.2 *For all $\alpha > 1$ and $r \geq 1$, there exists a WFA $\mathcal{A}$ that is (α, r)-DBP but not $(\alpha, r-1)$-DBP.*

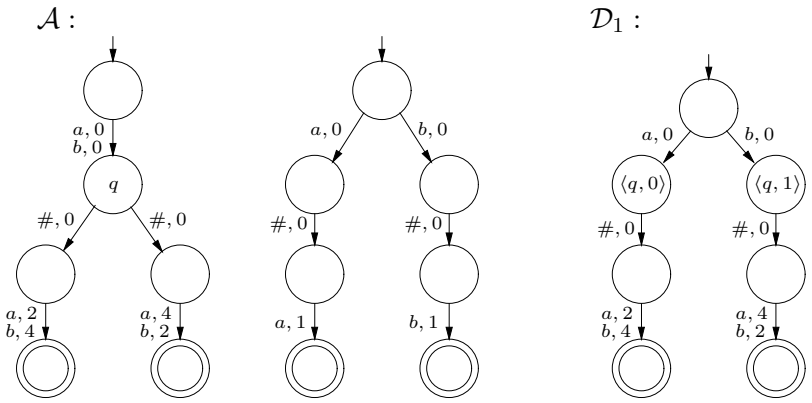

Figure 3: A WFA $\mathcal{A}$ and its refined 2-determinization by pruning $\mathcal{D}_1$.

Proof: The WFA $\mathcal{A}$ appearing in Figure 3 is $(2,1)$-DBP but not $(2,0)$-DBP. Note that the language of $\mathcal{A}$ consists of words of the form $x\#y$, for $x, y \in \{a, b\}$. The cost of an accepted word is 1 if $x = y$ (using the right part of $\mathcal{A}$), and is 2 otherwise (using the left part of $\mathcal{A}$). In the DWFA $\mathcal{D}_1$, obtained by pruning a 1-refinement of $\mathcal{A}$, the cost of an accepted word is 2 if $x = y$ and 4 otherwise. Note that a DBP for $\mathcal{A}$ cannot make use of the right part of $\mathcal{A}$, as words in which $x \neq y$ cannot be accepted by it.

The automaton $\mathcal{A}$ can be generalized for any $\alpha, r > 1$ to an automaton $\mathcal{A}_{\alpha,r}$ that is (α, r)-DBP but not $(\alpha, r-1)$-DBP. The automaton $A_{\alpha,r}$ is of size $O(r \cdot 2^r)$, it has a maximal branching degree of 2, and its language consists of all the words of the form $w\#^r v$, for $w, v \in \{a, b\}^r$, such that the cost of an accepted word is r if $w = v$ and is $\alpha \cdot r$ otherwise. For example, in Figure 4 we describe the WFA $\mathcal{A}_{4,3}$. Its right part accepts only words of the form $w\#^3 w$, for $w \in \{a, b\}^3$, at a cost of 3. Its left part accepts all words of the form $w\#^3 v$, for $w, v \in \{a, b\}^3$, at a cost of 12. Note that the nondeterminism of $\mathcal{A}_{4,3}$ lies in the choice of the initial state and in the #-transitions on the left part. In addition, note that after these #-transitions, for every word $w \in \{a, b\}^3$, there is a distinct branch that accepts it at a cost of 12, and the rest of the branches accept it at a cost of 14, 16, or 18. Thus, in order to be 4-competitive, a DWFA obtained by pruning a refinement of $\mathcal{A}_{4,3}$ should

have $2^3 = 8$ copies of the state q of $\mathcal{A}_{4,3}$. Such a DWFA assigns a cost of 12 to words in which $w = v$, and a cost of at most 18 to the rest of the words in the language. Indeed, after reading a prefix $w \in \{a, b\}^3$, the DWFA "knows" which branch it should take in order for the suffix to cost 12 when $v = w$. In case $v \neq w$, no matter to which branch we proceed, the cost would be at most 18. Since the original cost of the word is 12, it is within a factor of 4. Note that if we refine $\mathcal{A}_{4,3}$ by less than 3 variables, it cannot have 8 copies of the state q. Therefore, for any DWFA $\mathcal{D}$ obtained by pruning such a refinement, there exist two words $w, v \in \{a, b\}^3$, such that when $\mathcal{D}$ runs on each one of them it reaches the same copy of q. Thus, the runs of $\mathcal{D}$ on both $w\#^3$ and $v\#^3$ reach the same branch. But here, only one word is accepted at a cost of 12. Hence, at most one of the two words $w\#^3w$ and $v\#^3v$ can be accepted at a cost of 12, whereas at least one of them is accepted at a cost greater than 12, which is not 4-competitive.

In general, $\mathcal{A}_{\alpha,r}$ consists of two parts. The right part is deterministic, and accepts words of the form $w\#^r w$, for $w \in \{a, b\}^r$, at a cost of r. The left part is nondeterministic and accepts words of the form $w\#^r v$, for $w, v \in \{a, b\}^r$, at a cost of $\alpha \cdot r$. After its nondeterministic branches, the left part has 2^r branches, such that for every word $v \in \{a, b\}^r$ there is a distinct branch that accepts v at a cost of $\alpha \cdot r$ and accepts all other words in $\{a, b\}^r$ at a cost greater than $\alpha \cdot r$ but at most $\alpha^2 \cdot r$. This is achieved by generalizing the costs 4 and 6 in $\mathcal{A}_{4,3}$ by the costs α and β, respectively, for $\alpha < \beta \leq \alpha^2$. □

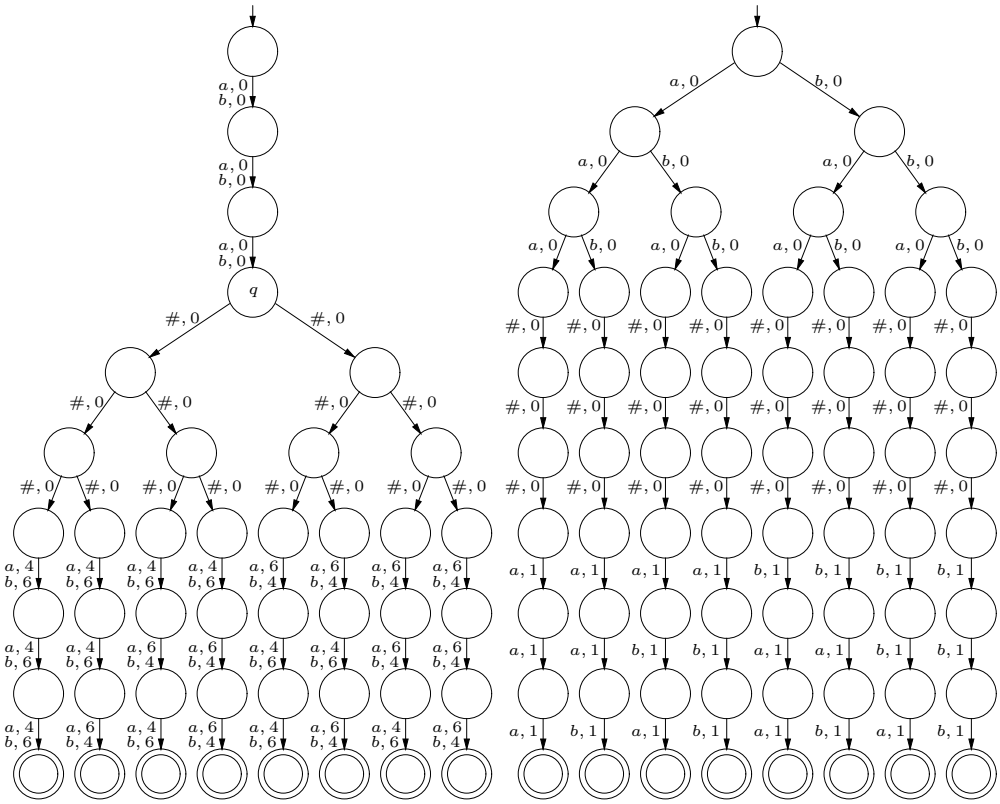

Figure 4: The WFA $\mathcal{A}_{4,3}$, which is $(4, 3)$-DBP but not $(4, 2)$-DBP.

Theorem 5.2 also follows from specific examples studied in the literature, showing that online algorithms that can store additional information can achieve better competitive ratios (for example, [6] shows a lower bound of 23/11 on the competitiveness of any deterministic trackless online algorithm for the 2-server problem[5]; whereas [10] shows that the competitive ratio of the Work Function Algorithm, which is also deterministic, but not trackless, for the 2-server problem is 2). Nonetheless, the proof of Theorem 5.2 serves to pinpoint the source of this phenomenon.

6 Discussion

The automata-theoretic approach we have described involves an explicit representation of the set S of configurations. One of the main challenges in formal verification is the need to cope with very big, often infinite, state spaces. *Symbolic reasoning* [7] is a leading approach for doing so. There, S is given symbolically (say, by a characteristic function), and the operations allowed to the verification algorithm are symbolic too. Since our algorithms are based on a fixed-point computation of a set of relations or functions, which are typically amenable to symbolic implementation, we are optimistic about adjusting them to the symbolic setting. Another challenge in our setting is that we would like to prove general properties of an online algorithm, rather than properties of instances corresponding to given parameters. This challenge is addressed in formal verification by means of *parametric reasoning* [13]. There, we reason about a system with many identical processes by studying properties of one of the processes. Parametric reasoning is, in general, undecidable. However, in the last decade there has been extensive research aimed at finding settings for which the problem is decidable, and on developing methods that are sound but incomplete. We are now examining their application to the setting of online algorithms. It is important to note that the field of formal verification has a history of successful implementations of algorithms with seemingly-infeasible complexity. For example, the tool MONA succesfuly decides the satisfiability of monadic second-order logic formulas – a problem whose complexity is non-elementary [12].

Finally, while we are able to decide whether a given online algorithm g has a given competitive ratio (Corollary 4.9), we left open the problem of finding the competitive ratio of g. Clearly, finding a finite upper bound on the competitive ratio would enable us to apply Corollary 4.9 and search for it. In the WFA formalism, this is reduced to finding, given a WFA $\mathcal{A}$, a finite bound γ such that $\mathcal{A}$ is $(\gamma, 0)$-DBP (or deciding that no such γ exists). We believe that such a bound can be found by analyzing the cost of cycles of $\mathcal{A}$, and we leave open the problem of doing it in polynomial time.

Acknowledgements We thank Marek Chrobak, Lawrence Larmore, and Nati Linial for helpful discussions.

References

[1] A.V. Aho, P.J. Denning, and J.D. Ullman. Principles of optimal page replacement. *Journal of the ACM*, 18(1):80–93, 1971.

[2] T. Ball, B. Cook, V. Levin, and S.K. Rajamani. Slam and static driver verifier: Technology transfer of formal methods inside microsoft. In *Integrated Formal Methods*, pages 1–20, 2004.

[5] Being memoryless is a stronger restriction than being trackless. Thus, this lower bound is valid also for memoryless algorithms.

[3] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *algorithmica*, 11(2):2–14, 1994.

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

[5] A. Borodin, N. Linial, and M.E. Saks. An optimal online algorithm for metrical task systems. In *Proc. 19th ACM Symp. on Theory of Computing*, pages 373–382, 1987.

[6] W.W. Bein and L.L. Larmore. Trackless Online Algorithms for the Server Problem. *Information Processing Letters*, 74:73–79, 2000.

[7] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[8] A. Chakrabarti, K. Chatterjee, T.A. Henzinger, O. Kupferman, and R. Majumdar. Verifying quantitative properties using bound functions. In *Proc. 13th CHARME*, LNCS 3725, pages 50–64. Springer, 2005.

[9] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[10] M. Chrobak and L.L. Larmore. The server problem and on-line games. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:11–64, 1991.

[11] M. Chrobak and L.L. Larmore. Private communication. 2008.

[12] J. Elgaard, N. Klarlund, and A. Möller. Mona 1.x: new techniques for WS1S and WS2S. In *Proc. 10th CAV*, LNCS 1427, pages 516–520, 1998.

[13] E.A. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *Proc. of the 17th Int. Conf. on Automated Deduction*, pages 236–255, 2000.

[14] T.A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th CSL*, LNCS 4207, pages 394–410. Springer, 2006.

[15] G.J. Holzmann. *The Spin Model Checker:primer and reference manual*. Addison-Wesley, 2004.

[16] B. Jobstmann, S. Galler, M. Weiglhofer, and R. Bloem. Anzu: A tool for property synthesis. In *Proc 19th CAV*, LNCS 4590, pages 258–262, 2007.

[17] H.W. Kuhn. Solvability and consistency for linear equations and inequalities. *The American Mathematical Monthly*, 63(4):217–232, 1956.

[18] W. Kuich and A.Salomaa. *Semirings, Automata, Languages*. EATCS Monographs on Theoretical Computer Science. Springer, 1986.

[19] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

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

[21] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[22] L. Rudolph. As described in a Hebrew Univeristy lecture, 1986.

[23] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

A Proof of Theorem 4.5

We first study a sequence of functions $\tilde{f}_0, \tilde{f}_1, \ldots : Q \times Q \to \mathbb{R} \cup \{-\infty, \infty\}$ that is very similar to the sequence $f_0, f_1, \ldots, f_{2n^2-1}$, except that we do not jump to infinity when a fixed-point is not reached after the n^2-th iteration. As a result, the sequence $\tilde{f}_0, \tilde{f}_1, \ldots$ may not reach a fixed-point, and an algorithm that calculates this sequence may never terminate. Nonetheless, studying $\tilde{f}$ is essential for our understanding of f.

Intuitively, $\tilde{f}_i(q, q')$ measures how well the state q' can deterministically simulate the state q, over words of length at most i. More precisely, we claim that for every automaton $\mathcal{A}' \in \det_0(\mathcal{A})$, and every pair of states $q, q' \in Q$, such that q' is reachable in $\mathcal{A}'$, we have that $\tilde{f}_i(q, q') = \textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i})$. The value $-\infty$ is assigned to $\tilde{f}_i(q, q')$ when there are no words in $\Sigma^{\leq i}$ that can be accepted from q, and the value ∞ is assigned when there is a word that can be accepted from q but not from q'. The sequence of functions $\tilde{f}_0, \tilde{f}_1, \ldots$ is defined as follows.

- At initialization:
$$\tilde{f}_0(q, q') = \begin{cases} -\infty & \text{if } q \notin F \\ 0 & \text{if } q \in F \text{ and } q' \in F \\ \infty & \text{if } q \in F \text{ and } q' \notin F, \end{cases}$$
- For $i > 0$:
$$\tilde{f}_i(q, q') = \max\{\tilde{f}_{i-1}(q, q'), \max_{a \in \Sigma} \tilde{f}_i(q, q', a)\}.$$

In the above, for every $i > 0$ and $a \in \Sigma$, the function $\tilde{f}_i(q, q', a)$ is defined as follows:

$$\tilde{f}_i(q, q', a) = \min_{u' \in \tilde{\rho}_{i-1}(q', a)} \max_{u \in \delta(q,a)} \tilde{f}_{i-1}(u, u') + c(q', a, u') - c(q, a, u).$$

Where the set $\tilde{\rho}_0(q', a) = \delta(q', a)$, and for $i > 0$:

$$\tilde{\rho}_i(q', a) = \{u' \in \tilde{\rho}_{i-1}(q', a) \mid \max_{u \in \delta(q', a)} \tilde{f}_{i-1}(u, u') + c(q', a, u') - c(q', a, u) \leq 0\}$$

In the expression above for $\tilde{f}_i(q, q', a)$, in case that for all $u \in \delta(q, a)$ we have that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} = \emptyset$, we set $\tilde{f}_i(q, q', a) = -\infty$ (note that this also covers the case $\delta(q, a) = \emptyset$). In case there is $u \in \delta(q, a)$ such that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} \neq \emptyset$, and $\tilde{\rho}_{i-1}(q', a) = \emptyset$, then we set $\tilde{f}_i(q, q', a) = \infty$.

Intuitively, a state $u' \in \tilde{\rho}_i(q', a)$ is a "witness" to the fact that $\tilde{f}_i(q', q', a) \leq 0$. We thus say that u' is a witness for $\tilde{f}_i(q', q', a)$. Obviously, if $\tilde{f}_i(q', q', a) > 0$ then $\tilde{\rho}_i(q', a) = \emptyset$. Note that for every $i \geq 0$, we have $\tilde{\rho}_i(q', a) \subseteq \delta(q', a)$. It follows that if $u' \in \tilde{\rho}_i(q', a)$, then $\tilde{f}_{i-1}(u', u') + c(q, a, u') - c(q, a, u') \leq 0$, implying that $\tilde{f}_{i-1}(u', u') \leq 0$. For notational convenience, given $q' \in Q$ and $a \in \Sigma$, we denote by $\tilde{\rho}(q, a)$, the set $\bigcap_{i \geq 0} \tilde{\rho}_i(q', a)$.

Observe that $\tilde{f}$ is monotonically increasing with i (i.e., for every $q, q' \in Q$, and every $i > 0$, we have that $\tilde{f}_i(q, q') \geq \tilde{f}_{i-1}(q, q')$), and that in the presence of loops in the automaton the sequence $\tilde{f}_0, \tilde{f}_1, \ldots$ may not reach a fixed-point. Let us start with a few easy observations.

Since Q and Σ are finite, and for every $q \in Q$ and every $a \in \Sigma$, the set $\tilde{\rho}_i(q, a)$ is finite and monotonically decreasing with i, there must be a point at which all the sets of witnesses have reached their minimal value. I.e.:

Lemma A.1 *There is an index $k \geq 0$ such that for every $i \geq k$, every $q \in Q$, and every $a \in \Sigma$, we have $\tilde{\rho}_i(q,a) = \tilde{\rho}(q,a)$.*

It is worth noting that $\tilde{f}_i$ may continue to evolve even after the sets of witnesses of all states have stabilized. Thus, Lemma A.1 does not imply that the sequence $\tilde{f}_0, \tilde{f}_1, \ldots$ ever reaches a fixed-point.

The following lemma shows that $\tilde{f}_i(q,q') = -\infty$ exactly when there are no words of length at most i that $\mathcal{A}$ can accept from q.

Lemma A.2 *For every $i \geq 0$, and $q, q' \in Q$, we have that $\tilde{f}_i(q,q') = -\infty$ iff $L(\mathcal{A}^q) \cap \Sigma^{\leq i} = \emptyset$.*

Proof: We prove the lemma by an induction on i. For $i = 0$, by the definition of $\tilde{f}_0$, we have $\tilde{f}_0(q,q') = -\infty$ iff $q \notin F$, i.e., iff no words of length 0 are accepted from q. For $i > 0$, assume that the lemma holds for $i-1$. By definition we have $\tilde{f}_i(q,q') = -\infty$ iff $\tilde{f}_{i-1}(q,q') = -\infty$ and for all $a \in \Sigma$ it holds that $\tilde{f}_i(q,q',a) = -\infty$. I.e., iff $\tilde{f}_{i-1}(q,q') = -\infty$, and for all $a \in \Sigma$, either there exists a $u' \in \tilde{\rho}_{i-1}(q',a)$ such that for every $u \in \delta(q,a)$ we have $\tilde{f}_{i-1}(u,u') = -\infty$, or that for every $u \in \delta(q,a)$ we have $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} = \emptyset$. Applying the induction hypothesis to all such $\tilde{f}_{i-1}(u,u')$ we get that $\tilde{f}_i(q,q') = -\infty$ iff $\tilde{f}_{i-1}(q,q') = -\infty$, and for all $a \in \Sigma$, $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} = \emptyset$ for every $u \in \delta(q,a)$. This means that $L(\mathcal{A}^q)$, as well as $L(\mathcal{A}^u)$, for every successor u of q, does not contain words of length at most $i-1$. Therefore, $L(\mathcal{A}^q)$ does not contain words of length at most i. □

The following proposition shows that if there is a word of length at most i that can be accepted from q but not from q', then $\tilde{f}_i(q,q') = \infty$, as needed to reflect the fact that q' can not simulate q, regardless of cost, over words of length at most i.

Proposition A.3 *For $i \geq 0$ and $q, q' \in Q$, if $(L(\mathcal{A}^q) \setminus L(\mathcal{A}^{q'}) \cap \Sigma^{\leq i}) \neq \emptyset$ then $\tilde{f}_i(q,q') = \infty$.*

Proof: We prove the proposition by an induction on i. For $i = 0$, $\varepsilon \in L(\mathcal{A}^q) \setminus L(\mathcal{A}^{q'})$ iff $q \in F$ and $q' \notin F$, and thus $\tilde{f}_0(q,q') = \infty$. For $i > 0$, assume that the proposition holds for $i-1$. If there is a word of length at most $i-1$ that is in $L(\mathcal{A}^q) \setminus L(\mathcal{A}^{q'})$, then by the induction hypothesis we have $\tilde{f}_{i-1}(q,q') = \infty$, and thus also $\tilde{f}_i(q,q') = \infty$. Otherwise, there is a word w of length i in $L(\mathcal{A}^q) \setminus L(\mathcal{A}^{q'})$. Let $w = a \cdot x$, where $a \in \Sigma$ and $x \in \Sigma^{i-1}$, and let $u \in \delta(q,a)$ be a state such that $x \in L(\mathcal{A}^u)$. Since for all $u' \in \delta(q',a)$ we have $x \notin L(\mathcal{A}^{q'})$, by the induction hypothesis we have $\tilde{f}_{i-1}(u,u') = \infty$ for all $u' \in \delta(q',a)$. Thus, by the definition of $\tilde{f}$, in any case we have $\tilde{f}_i(q,q',a) = \infty$, and hence $\tilde{f}_i(q,q') = \infty$. □

Recall that by our intuition, $\tilde{f}_i(q,q,0) \leq 0$ indicates that the state q can deterministically simulate itself, without any cost penalty, over words of length at most i. This in turn implies that the witnesses for this fact must be able to do the same over words of length at most $i-1$, etc. The next proposition formalizes this intuition.

Proposition A.4 *For every state $q \in Q$, letters $a, b \in \Sigma$, and state $u \in \tilde{\rho}(q,a)$, if $\delta(u,b) \neq \emptyset$ then $\tilde{\rho}(u,b) \neq \emptyset$.*

Proof: We have to show that if $\delta(u,b) \neq \emptyset$ then for every $i \geq 0$ we have $\tilde{\rho}_i(u,b) \neq \emptyset$. For $i = 0$, by definition, $\tilde{\rho}_0(u,b) = \delta(u,b)$. For $i > 0$, since $u \in \tilde{\rho}(q,a)$ then in particular $u \in \tilde{\rho}_{i+1}(q,a)$. Hence, as we observed earlier, $\tilde{f}_i(u,u) \leq 0$. It follows that $\tilde{f}_i(u,u,b) \leq 0$. Assume first that for all $v \in \delta(u,b)$, we have that $L(\mathcal{A}^v) \cap \Sigma^{\leq i-1} = \emptyset$. Since by our assumption $\delta(u,b) \neq \emptyset$, we can take some $v' \in \delta(u,b)$. Hence, by Lemma A.2, we have that $\tilde{f}_{i-1}(v,v') = -\infty$ for all $v \in \delta(u,b)$, implying that $v' \in \tilde{\rho}_i(u,b)$. Assume now that there is a state $v' \in \delta(u,b)$ such that $L(\mathcal{A}^v) \cap \Sigma^{\leq i-1} \neq \emptyset$. Recall that $\tilde{f}_{i+1}(u,u,b) \leq 0$. Hence, by the definition of $\tilde{f}_{i+1}$, it must be that $\tilde{\rho}_i(u,b) \neq \emptyset$. □

Definition A.5 *Given a WFA $\mathcal{A}$, we say that a WFA $\mathcal{A}' = \langle \Sigma, Q, \Delta', c, q'_0, F \rangle$ is* good *for $\tilde{f}_i$, if $\mathcal{A}'$ is a DWFA embodied in $\mathcal{A}$, and for every reachable state $q' \in Q$, and every $a \in \Sigma$, the following two properties hold:*

- $\delta'(q',a) \in \tilde{\rho}_i(q',a)$.
- *If $\delta(q',a) \neq \emptyset$ then $\delta'(q',a) \neq \emptyset$.*

If $\mathcal{A}'$ is good for $\tilde{f}_i$ for every $i \geq 0$, we simply say that $\mathcal{A}'$ is good for $\tilde{f}$. Note that since $j \leq i$ implies that $\tilde{\rho}_i(q',a) \subseteq \tilde{\rho}_j(q',a)$, we have:

Corollary A.6 *If $\mathcal{A}'$ is good for $\tilde{f}_i$, then $\mathcal{A}'$ is good for $\tilde{f}_j$, for every $j \leq i$.*

It follows that the set of automata that are good for $\tilde{f}_i$ is monotonically decreasing with i. Since $\mathcal{A}$ has finitely many embodied automata, this sequence of sets must reach a minimal value. Hence:

Corollary A.7 *There is an index $k \geq 0$ such that for every $i \geq k$, if $\mathcal{A}'$ is good for $\tilde{f}_i$ then $\mathcal{A}'$ is good for $\tilde{f}$.*

It is not hard to see that Proposition A.4 implies the following corollary:

Corollary A.8 *If $\mathcal{A}'$ is good for $\tilde{f}$, then for every $q' \in Q$ such that q' is reachable in $\mathcal{A}'$, and for every $a \in \Sigma$, if $u' \in \tilde{\rho}(q',a)$ then there is an automaton $\mathcal{A}''$ that is good for $\tilde{f}$, in which u' is reachable.*

In Lemma A.12 we show that every automaton in $det_0(\mathcal{A})$ is good for $\tilde{f}$. Our objective now is to prove that for every word w of length at most i, the value of $\tilde{f}_i(q,q')$ is an upper bound on the cost difference between accepting w from q' in any automaton that is good for $\tilde{f}_i$, and the cost of accepting w from q in $\mathcal{A}$. To this aim, we first show that $\tilde{f}_i$ satisfies a form of a transitivity inequality. Informally, this inequality claims that the cost difference incurred in simulating q by q' is at most that incurred in simulating q by some other state p, plus that of simulating p by q'. To get a feel for why this is true, take some $i > 0$, and assume that by some good fortune there is a letter $a \in \Sigma$ and a-successors r, s', v' of q, p, q' (respectively), such that $\tilde{f}_i(q,p) = \tilde{f}_i(q,p,a) = \tilde{f}_{i-1}(r,s') + c(p,a,s') - c(q,a,r)$, $\tilde{f}_i(p,q') = \tilde{f}_i(p,q',a) = \tilde{f}_{i-1}(s',v') + c(q',a,v') - c(p,a,s')$, and $\tilde{f}_i(q,q') = \tilde{f}_i(q,q',a) = \tilde{f}_{i-1}(r,v') + c(q',a,v') - c(q,a,r)$. I.e., the minimums and maximums in all three expressions for $\tilde{f}_i(q,p,a)$, $\tilde{f}_i(p,q',a)$ and $\tilde{f}_i(q,q',a)$ are achieved with the same successors r, s', v'. It follows that $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') = (\tilde{f}_{i-1}(r,s') + c(p,a,s') - c(q,a,r)) + (\tilde{f}_{i-1}(s',v') + c(q',a,v') - c(p,a,s')) = \tilde{f}_{i-1}(r,s') + \tilde{f}_{i-1}(s',v') + c(q',a,v') - c(q,a,r)$. By inductively applying the same kind of reasoning (and good fortune) to $\tilde{f}_{i-1}(r,s') + \tilde{f}_{i-1}(s',v')$, we can deduce that $\tilde{f}_{i-1}(r,s') +$

$\tilde{f}_{i-1}(s',v') = \tilde{f}_{i-1}(r,v')$, and thus $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') = \tilde{f}_{i-1}(r,v') + c(q',a,v') - c(q,a,r) = \tilde{f}_i(q,q',a) = \tilde{f}_i(q,q')$. Obviously, in the general case we can not expect to be so fortunate, and it is not always the case that $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') = \tilde{f}_i(q,q')$. However, using similar reasoning and careful handling of the minimums and maximums in the expressions for $\tilde{f}_i(q,p)$, $\tilde{f}_i(p,q')$, and $\tilde{f}_i(q,q')$, we can show the following:

Proposition A.9 *For every $i \geq 0$, and every $q,p,q' \in Q$, if $\tilde{f}_i(q,p)$ and $\tilde{f}_i(p,q')$ are in $\mathbb{R}$, then so is $\tilde{f}_i(q,q')$, and $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') \geq \tilde{f}_i(q,q')$.*

Proof: We prove the proposition by an induction on i. For $i = 0$, it must be that $p,q,q' \in F$, and $\tilde{f}_i(q,p) = \tilde{f}_i(p,q') = \tilde{f}_i(q,q') = 0$. For $i > 0$, assume that the proposition holds for $i-1$.

Assume first that $\tilde{f}_i(q,q') = \tilde{f}_{i-1}(q,q')$. We claim that $\tilde{f}_{i-1}(q,p), \tilde{f}_{i-1}(p,q') \in \mathbb{R}$, and thus by the monotonicity of $\tilde{f}$ and the induction hypothesis we have that $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') \geq \tilde{f}_{i-1}(q,p) + \tilde{f}_{i-1}(p,q') \geq \tilde{f}_{i-1}(q,q') = \tilde{f}_i(q,q')$. We now prove that indeed $\tilde{f}_{i-1}(q,p), \tilde{f}_{i-1}(p,q') \in \mathbb{R}$. Note that since $\tilde{f}$ is monotonically increasing, and $\tilde{f}_i(q,p), \tilde{f}_i(p,q') \in \mathbb{R}$, it is enough to show that $\tilde{f}_{i-1}(q,p) \neq -\infty$, and $\tilde{f}_{i-1}(p,q') \neq -\infty$. Note that since $\tilde{f}_i(q,p) \neq -\infty$ then by Lemma A.2 also $\tilde{f}_i(q,q') \neq -\infty$. Since by our assumption $\tilde{f}_{i-1}(q,q') = \tilde{f}_i(q,q')$, then also $\tilde{f}_{i-1}(q,q') \neq -\infty$, and by Lemma A.2, $L(\mathcal{A}^q) \cap \Sigma^{\leq i-1} \neq \emptyset$, and thus also $\tilde{f}_{i-1}(q,p) \neq -\infty$. To see that $\tilde{f}_{i-1}(p,q') \neq -\infty$, observe that since $L(\mathcal{A}^q) \cap \Sigma^{\leq i-1} \neq \emptyset$, and $\infty \neq \tilde{f}_i(q,p) \geq \tilde{f}_{i-1}(q,p)$, by Proposition A.3 it must be that $L(\mathcal{A}^p) \cap \Sigma^{\leq i-1} \neq \emptyset$. Thus, by Lemma A.2, we have that $\tilde{f}_{i-1}(p,q') \neq -\infty$.

Assume now that $\tilde{f}_i(q,q') > \tilde{f}_{i-1}(q,q')$. Hence, there is a letter $a \in \Sigma$ such that $\tilde{f}_i(q,q') = \tilde{f}_i(q,q',a)$. Note that $\tilde{f}_i(q,q') \neq -\infty$, and thus, there is a state $u \in \delta(q,a)$ such that $L(\mathcal{A}^q) \cap \Sigma^{\leq i-1} \neq \emptyset$. This, together with the fact that $\tilde{f}_i(q,p) \neq \infty$, imply that $\tilde{\rho}_{i-1}(p,a) \neq \emptyset$. Furthermore, by Proposition A.3, for some $r \in \tilde{\rho}_{i-1}(p,a)$ we have that $L(\mathcal{A}^r) \cap \Sigma^{\leq i-1} \neq \emptyset$. Since $\tilde{f}_i(p,q') \neq \infty$ it follows that also $\tilde{\rho}_{i-1}(q',a) \neq \emptyset$. The above observations show that all the $\min$ and $\max$ expressions in the remainder of the proof range over non-empty sets. By definition we have:

$$\tilde{f}_i(q,p) \geq \tilde{f}_i(q,p,a) = \min_{s \in \tilde{\rho}_{i-1}(p,a)} \max_{r \in \delta(q,a)} \tilde{f}_{i-1}(r,s) + c(p,a,s) - c(q,a,r).$$

By fixing $s' \in \tilde{\rho}_{i-1}(p,a)$ to be some state for which the minimum above is achieved, we get that for every $r \in \delta(q,a)$ the following holds:

$$\tilde{f}_i(q,p) \geq \tilde{f}_{i-1}(r,s') + c(p,a,s') - c(q,a,r) \tag{1}$$

By definition:

$$\tilde{f}_i(p,q') \geq \tilde{f}_i(p,q',a) = \min_{v \in \tilde{\rho}_{i-1}(q',a)} \max_{s \in \delta(p,a)} \tilde{f}_{i-1}(s,v) + c(q',a,v) - c(p,a,s).$$

By fixing $v' \in \tilde{\rho}_{i-1}(q',a)$ to be some state for which the minimum above is achieved and limiting our attention to s' we get that:

$$\tilde{f}_i(p,q') \geq \tilde{f}_{i-1}(s',v') + c(q',a,v') - c(p,a,s') \tag{2}$$

Note that since $\tilde{f}_i(q,p), \tilde{f}_i(p,q') \in \mathbb{R}$, inequalities 1 and 2 above imply that $\tilde{f}_{i-1}(r,s'), \tilde{f}_{i-1}(s',v') \neq \infty$. Hence, we are allowed to combine inequalities 1 and 2 (without fear of mixing ∞ and $-\infty$) and get that for every $r \in \delta(q,a)$:

$$\tilde{f}_i(q,p) + \tilde{f}_i(p,q') \geq \tilde{f}_{i-1}(r,s') - c(q,a,r) + \tilde{f}_{i-1}(s',v') + c(q',a,v') \tag{3}$$

On the other hand, recall that a was chosen such that :

$$\tilde{f}_i(q,q') = \tilde{f}_i(q,q',a) = \min_{v \in \tilde{\rho}_{i-1}(q',a)} \max_{r \in \delta(q,a)} \tilde{f}_{i-1}(r,v) + c(q',a,v) - c(q,a,r)$$

By limiting our attention to v', and taking some r for which $\tilde{f}_{i-1}(r,v') + c(q',a,v') - c(q,a,r)$ is maximal, we get that:

$$\tilde{f}_{i-1}(r,v') + c(q',a,v') - c(q,a,r) \geq \tilde{f}_i(q,q') \tag{4}$$

Observe that if $\tilde{f}_{i-1}(r,s'), \tilde{f}_{i-1}(s',v') \in \mathbb{R}$ we can apply the induction hypothesis and obtain that $\tilde{f}_{i-1}(r,s') + \tilde{f}_{i-1}(s',v') \geq \tilde{f}_{i-1}(r,v')$. By substituting the above into Inequality 3 and combining the result with Inequality 4 we obtain that $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') \geq \tilde{f}_i(q,q')$. Thus, to complete the proof we just need to show that indeed $\tilde{f}_{i-1}(r,s'), \tilde{f}_{i-1}(s',v') \in \mathbb{R}$. Recall that $\tilde{f}_{i-1}(r,s'), \tilde{f}_{i-1}(s',v') \neq \infty$. Hence, it is enough to show that $\tilde{f}_{i-1}(r,s'), \tilde{f}_{i-1}(s',v') \neq -\infty$. Recall that $\tilde{f}_i(q,q') > -\infty$, and thus, by Inequality 4, also $\tilde{f}_{i-1}(r,v') > -\infty$. By Lemma A.2 we have that $L(\mathcal{A}^r) \cap \Sigma^{\leq i-1} \neq \emptyset$, and thus also $\tilde{f}_{i-1}(r,s') \neq -\infty$. To see that $\tilde{f}_{i-1}(s',v') \neq -\infty$, recall that $\tilde{f}_{i-1}(r,s') \neq \infty$. Thus, by Proposition A.3, since $L(\mathcal{A}^r) \cap \Sigma^{\leq i-1} \neq \emptyset$ we get that $L(\mathcal{A}^{s'}) \cap \Sigma^{\leq i-1} \neq \emptyset$. By Lemma A.2 it follows that $\tilde{f}_{i-1}(s',v') \neq -\infty$, which completes the proof. □

The following lemma shows that $\tilde{f}_i$ is an upper bound on the cost difference, over words of length at most i, between any automaton $\mathcal{A}'$ that is good for $\tilde{f}_i$, and $\mathcal{A}$. It is not hard to see from the definition of $\tilde{f}_i$ that the lemma holds if $\mathcal{A}'$ is such that for every pair of states $q, q' \in Q$ and letter $a \in \Sigma$, if u' is the a-successor of q' in $\mathcal{A}'$ then the minimum in the expression for $\tilde{f}_i(q,q',a)$ is achieved with u'. However, it may not be immediately clear why the minimum in the expression for $\tilde{f}_i(q,q',a)$ is indeed achieved with u'. Informally, the argument goes as follows. Assuming by induction that the lemma holds for $i-1$, and since $\mathcal{A}'$ is good for $\tilde{f}_i$ implies that $u' \in \tilde{\rho}_i(q',a)$, it follows that u' can simulate any a-successor v' of q' with a cost difference that can be completely offset by the difference in the costs of the transition from q' to u' and the transition from q' to v'. Hence, from the point of view of q', u' can simulate v' without any penalty. By proposition A.9, it follows that from the point of view of q', u' is as good as v' in simulating any a-successor u of q. Since this is true for every a-successor v' of q', it must be that the minimum in the expression for $\tilde{f}_i(q,q',a)$ is achieved with u'. A more formal argument follows.

Lemma A.10 *For $i \geq 0$, if $\mathcal{A}'$ is good for $\tilde{f}_i$, then for all $q, q' \in Q$, such that q' is reachable in $\mathcal{A}'$, if $\tilde{f}_i(q,q') \in \mathbb{R} \cup \{\infty\}$ then costdiff$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i}) \leq \tilde{f}_i(q,q')$.*

Proof: Since $\tilde{f}_i(q,q') \neq -\infty$, by Lemma A.2 we have that $L(\mathcal{A}^q) \cap \Sigma^{\leq i} \neq \emptyset$. Let w be some word in $L(\mathcal{A}^q) \cap \Sigma^{\leq i}$. We have to prove that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) \leq \tilde{f}_i(q,q')$. Consider first the case where $\tilde{f}_i(q,q') = \infty$. Since $w \in L(\mathcal{A}^q)$ we have that $cost(\mathcal{A}^q, w) \in \mathbb{R}$. Hence, $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w)$ is well defined and $\leq \infty$.

Consider now the case where $\tilde{f}_i(q,q') \in \mathbb{R}$. We prove this case by an induction on i. For $i = 0$ we have $w = \epsilon$, and by the definition of $\tilde{f}_0$ we have $\tilde{f}_0(q,q') \in \mathbb{R}$ iff both q and q' are in F. Hence, $cost(\mathcal{A}^q, \epsilon) = cost(\mathcal{A}'^{q'}, \epsilon) = \tilde{f}_0(q,q') = 0$. For the induction step, we assume that the lemma holds for $i-1$, and prove it for i. Consider first the empty word. By Corollary A.6, $\mathcal{A}'$ is also good for $\tilde{f}_0$. Hence, by the induction hypothesis, we have that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) \leq \tilde{f}_0(q,q')$. Since $\tilde{f}$

is monotonically increasing, $\tilde{f}_0(q,q') \leq \tilde{f}_i(q,q')$. Assume now that $|w| > 0$, and let $w = ax$, where $a \in \Sigma$ and $x \in \Sigma^{\leq i-1}$. Since $w \in L(\mathcal{A}^q)$ we have that $cost(\mathcal{A}^q, w) \in \mathbb{R}$. We can thus let $u \in \delta(q,a)$ be the successor of q in a run of $\mathcal{A}^q$ on w costing exactly $cost(\mathcal{A}^q, w)$. By definition we have

$$\tilde{f}_i(q,q') \geq \tilde{f}_i(q,q',a) = \min_{v' \in \tilde{\rho}_{i-1}(q',a)} \max_{\bar{u} \in \delta(q,a)} \tilde{f}_{i-1}(\bar{u},v') + c(q',a,v') - c(q,a,\bar{u}).$$

Since $\tilde{f}_i(q,q') \in \mathbb{R}$, and u is such that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} \neq \emptyset$, by the definition of $\tilde{f}$ it must be that $\tilde{\rho}_{i-1}(q',a) \neq \emptyset$. We can thus choose $v' \in \tilde{\rho}_{i-1}(q',a)$ for which the minimum in the above expression is attained. By limiting our attention to u we get:

$$\tilde{f}_i(q,q') \geq \tilde{f}_i(q,q',a) \geq \tilde{f}_{i-1}(u,v') + c(q',a,v') - c(q,a,u) \tag{5}$$

Recall that q' is reachable in $\mathcal{A}'$, and that $\emptyset \neq \tilde{\rho}_{i-1}(q',a) \subseteq \delta(q',a)$. Hence, since $\mathcal{A}'$ is good for $\tilde{f}_i$, there is a state $u' \in \tilde{\rho}_i(q',a)$, such that $u' = \delta'(q',a)$. By the definition of $\tilde{\rho}_i(q',a)$ we have:

$$0 \geq \tilde{f}_{i-1}(v',u') + c(q',a,u') - c(q',a,v') \tag{6}$$

Since $\tilde{f}_i(q,q') \in \mathbb{R}$, by Inequality 5 we have $\tilde{f}_{i-1}(u,v') \neq \infty$. By Inequality 6, $\tilde{f}_{i-1}(v',u') \neq \infty$. Hence, we can add inequalities 5 and 6 (without fear of mixing ∞ and $-\infty$) and get:

$$\tilde{f}_i(q,q') \geq \tilde{f}_{i-1}(u,v') - c(q,a,u) + \tilde{f}_{i-1}(v',u') + c(q',a,u') \tag{7}$$

In preparation to applying Proposition A.9, we now show that $\tilde{f}_{i-1}(u,v')$ and $\tilde{f}_{i-1}(v',u')$ are both in $\mathbb{R}$. Recall that $\tilde{f}_{i-1}(u,v'), \tilde{f}_{i-1}(v',u') \neq \infty$. Since $x \in L(\mathcal{A}^u)$, Lemma A.2 implies that $\tilde{f}_{i-1}(u,v') \neq -\infty$. Since $\tilde{f}_{i-1}(u,v') \neq \infty$, by Proposition A.3 we have that x is also in $L(\mathcal{A}^{v'})$. Hence, by Lemma A.2, $\tilde{f}_{i-1}(v',u') \neq -\infty$. We can now apply Proposition A.9 and obtain that $\tilde{f}_{i-1}(u,v') + \tilde{f}_{i-1}(v',u') \geq \tilde{f}_{i-1}(u,u')$. Inequality 7 thus becomes:

$$\tilde{f}_i(q,q') \geq \tilde{f}_{i-1}(u,u') - c(q,a,u) + c(q',a,u') \tag{8}$$

Note that by our choice of u and u' it follows that $cost(\mathcal{A}^q, w) = cost(\mathcal{A}^u, x) + c(q,a,u)$ and $cost(\mathcal{A}'^{q'}, w) = cost(\mathcal{A}'^{u'}, x) + c(q',a,u')$. Hence, by Inequality 8, to show that $\tilde{f}_i(q,q') \geq cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w)$, we just have to show that $\tilde{f}_{i-1}(u,u') \geq cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x)$. Recall that since $x \in L(\mathcal{A}^u) \cap \Sigma^{\leq i-1}$, by Lemma A.2 we have that $\tilde{f}_{i-1}(u,u') \in \mathbb{R} \cup \{\infty\}$. Since by Corollary A.6, $\mathcal{A}'$ is also good for $\tilde{f}_{i-1}$, and since q' is reachable in $\mathcal{A}'$ implies that so is u', we can apply the induction hypothesis to $\tilde{f}_{i-1}(u,u')$ and obtain that $\tilde{f}_{i-1}(u,u') \geq cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x)$. □

The following lemma shows that $\tilde{f}_i$ is a lower bound on the cost difference, over words of length at most i, between any automaton that is good for $\tilde{f}_i$, and $\mathcal{A}$.

Lemma A.11 *For $i \geq 0$, if $\mathcal{A}'$ is good for $\tilde{f}_i$, then for all $q, q' \in Q$, such that q' is reachable in $\mathcal{A}'$, if $\tilde{f}_i(q,q') \in \mathbb{R} \cup \{\infty\}$ then $\tilde{f}_i(q,q') \leq$ costdiff$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i})$.*

Proof: We prove the lemma by induction on i. For $i = 0$, we have that $\tilde{f}_0(q, q') \in \mathbb{R}$ iff both q and q' are in F, implying that $cost(\mathcal{A}'^{q'}, \epsilon) = cost(\mathcal{A}^q, \epsilon) = \tilde{f}_0(q, q') = 0$. In case $\tilde{f}_0(q, q') = \infty$, then $q \in F$ but $q' \notin F$. Hence, $cost(\mathcal{A}'^{q'}, \epsilon) - cost(\mathcal{A}^q, \epsilon) = \infty - 0 = \infty$.

For the induction step, assume that the lemma holds for $i - 1$. By the definition of $\tilde{f}_i$ we have $\tilde{f}_i(q, q') = \max\{\tilde{f}_{i-1}(q, q'), \max_{a\in\Sigma} \tilde{f}_i(q, q', a)\}$. Assume first that $\tilde{f}_i(q, q') = \tilde{f}_{i-1}(q, q')$. By the induction hypothesis, there is a word w of length at most $i - 1$ (hence at most i) such that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) \geq \tilde{f}_i(q, q')$. Assume now that $\tilde{f}_i(q, q') > \tilde{f}_{i-1}(q, q')$. Thus, there exists $a \in \Sigma$ such that:

$$\tilde{f}_i(q, q') = \tilde{f}_i(q, q', a) = \min_{v'\in\tilde{\rho}_{i-1}(q',a)} \max_{u\in\delta(q,a)} \tilde{f}_{i-1}(u, v') + c(q', a, v') - c(q, a, u).$$

Observe that by the definition of $\tilde{f}_i$, since $\tilde{f}_i(q, q', a) = \tilde{f}_i(q, q') \neq -\infty$, there is a state $u \in \delta(q, a)$ such that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} \neq \emptyset$. Consider first the case where $\delta'(q', a) = \emptyset$. Let $w = a \cdot x$, where x is some word in $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1}$. It follows that $cost(\mathcal{A}^q, w) \in \mathbb{R}$, and $cost(\mathcal{A}'^{q'}, w) = \infty$. Thus, $\textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i}) = \infty \geq \tilde{f}_i(q, q')$. Consider now the case where there is a state $u' = \delta'(q', a)$. Since $\mathcal{A}'$ is good for $\tilde{f}_i$, and q' is reachable in $\mathcal{A}'$, then $u' \in \tilde{\rho}_i(q', a)$. Recall that by definition $\tilde{\rho}_i(q', a) \subseteq \tilde{\rho}_{i-1}(q', a)$, and thus u' belongs to the set over which the minimum in the expression above for $\tilde{f}_i(q, q', a)$ is taken. It follows that there is a state $u \in \delta(q, a)$ such that $\tilde{f}_i(q, q') \leq \tilde{f}_{i-1}(u, u') + c(q', a, u') - c(q, a, u)$. Since by our assumption $\tilde{f}_i(q, q') \in \mathbb{R} \cup \{\infty\}$, then by the last inequality also $\tilde{f}_{i-1}(u, u') \in \mathbb{R} \cup \{\infty\}$, and we can apply the induction hypothesis and obtain that there is a word x of length at most $i - 1$ such that $\tilde{f}_{i-1}(u, u') \leq cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x)$. Combining the last two inequalities involving $\tilde{f}_{i-1}(u, u')$ we get that $\tilde{f}_i(q, q') \leq cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x) + c(q', a, u') - c(q, a, u)$. Observe that since $\mathcal{A}'$ is deterministic but $\mathcal{A}'$ is nondeterministic we have that $cost(\mathcal{A}'^{u'}, x) + c(q', a, u') = cost(\mathcal{A}'^{q'}, a \cdot x)$, and $cost(\mathcal{A}^u, x) + c(q, a, u) \geq cost(\mathcal{A}^q, a \cdot x)$. It follows that $\tilde{f}_i(q, q') \leq cost(\mathcal{A}'^{q'}, a \cdot x) - cost(\mathcal{A}^q, a \cdot x) \leq \textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i})$, which completes the proof. □

Lemma A.12 *If $\mathcal{A}' \in det_0(\mathcal{A})$ then $\mathcal{A}'$ is good for $\tilde{f}$.*

Proof: We first prove that for every reachable state q in $\mathcal{A}'$, and every letter $a \in \Sigma$, if $\delta(q, a) \neq \emptyset$ then $\delta'(q, a) \neq \emptyset$. To see that, note that since $\mathcal{A}'$ is a DWFA embodied in $\mathcal{A}$ and equivalent to $\mathcal{A}$, it must be that $L(\mathcal{A}^q) = L(\mathcal{A}'^q)$. Since $\mathcal{A}$ has no useless states we must have that $\delta(q, a) \neq \emptyset$ implies that $\delta'(q, a) \neq \emptyset$. It remains to show that for every $i \geq 0$, every reachable state q in $\mathcal{A}'$, and every letter $a \in \Sigma$, we have that $\delta'(q, a) \in \tilde{\rho}_i(q, a)$. We prove this by an induction on i. The case $i = 0$ is true by definition. For $i > 0$, we assume that $\mathcal{A}'$ is good for $\tilde{f}_{i-1}$, and we have to show that if $u' = \delta'(q, a)$ then:

$$\max_{u\in\delta(q,a)} \tilde{f}_{i-1}(u, u') + c(q, a, u') - c(q, a, u) \leq 0$$

Assume by way of contradiction that there is a state $u \in \delta(q, a)$ such that $\tilde{f}_{i-1}(u, u') + c(q, a, u') - c(q, a, u) > 0$. It follows that $\tilde{f}_{i-1}(u, u') \in \mathbb{R} \cup \{\infty\}$. Since u' is reachable in $\mathcal{A}'$, and by the induction hypothesis $\mathcal{A}'$ is good for $\tilde{f}_{i-1}$, we can apply Lemma A.11 and obtain that there is a word $x \in L(\mathcal{A}^u) \cap \Sigma^{\leq i-1}$ such that $\tilde{f}_{i-1}(u, u') \leq cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x)$. Combining the last two inequalities involving $\tilde{f}_{i-1}(u, u')$ we get: $cost(\mathcal{A}'^{u'}, x) - cost(\mathcal{A}^u, x) + c(q, a, u') - c(q, a, u) > 0$. Observe that since $\mathcal{A}'$ is deterministic but $\mathcal{A}'$ is nondeterministic we have $cost(\mathcal{A}'^{u'}, x) + c(q, a, u') = cost(\mathcal{A}'^q, a \cdot x)$, and $cost(\mathcal{A}^u, x) + c(q, a, u) \geq cost(\mathcal{A}^q, a \cdot x)$. It follows that $cost(\mathcal{A}'^q, a \cdot x) -$

$cost(\mathcal{A}^q, a \cdot x) > 0$, which is a contradiction since $\mathcal{A}' \in det_0(\mathcal{A})$ implies that for every reachable state q in $\mathcal{A}'$, and every word $w \in \Sigma^*$, we have $cost(\mathcal{A}'^q, w) \leq cost(\mathcal{A}^q, w)$. □

Recall that the sequence of functions $\tilde{f}_0, \tilde{f}_1, \ldots$ may not reach a fixed-point. Hence, an algorithm that calculates the sequence $\tilde{f}_0, \tilde{f}_1, \ldots$ may never terminate. However, as the next proposition shows, if $i \geq n^2$, an increase in the value of $\tilde{f}_i(q, q')$ compared to $\tilde{f}_{i-1}(q, q')$ indicates that for every $\mathcal{A}'$ that is good for $\tilde{f}_i$ there are cycles in $\mathcal{A}$ and $\mathcal{A}'$ that can be pumped to create longer and longer words with an ever increasing cost difference between $\mathcal{A}'^{q'}$ and $\mathcal{A}^q$. This implies that q' can not deterministically simulate q with a bounded cost difference, and suggests that we can assign the value ∞ to this pair already at stage i. This is exactly what our algorithm which calculates the sequence $f_0, f_1, \ldots, f_{2n^2-1}$ (instead of $\tilde{f}_0, \tilde{f}_1, \ldots$) does.

Proposition A.13 *For $i \geq n^2$, if $\mathcal{A}'$ is good for $\tilde{f}_i$, then for every $q, q' \in Q$, such that q' is reachable in $\mathcal{A}'$, if $\tilde{f}_i(q, q') > \tilde{f}_{i-1}(q, q')$ then costdiff$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \infty$.*

Proof: We first need the following notation. Given a WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$, and two states $p, q \in Q$, a *path* in $\mathcal{A}$ from p to q is a finite sequence of states $\pi = \pi_0, \pi_1, \cdots, \pi_{m-1}$ such that $\pi_0 = p$, $\pi_{m-1} = q$, and for every $0 \leq i < m-1$, we have that $\pi_{i+1} \in \bigcup_{a \in \Sigma} \delta(\pi_i, a)$. If $\pi_0 \in \bigcup_{a \in \Sigma} \delta(\pi_{m-1}, a)$ then π is a *cycle*.

Assume then that $\tilde{f}_i(q, q') > \tilde{f}_{i-1}(q, q')$. It follows that $\tilde{f}_i(q, q') \in \mathbb{R} \cup \{\infty\}$. Hence, by Lemma A.11, there is a word $w \in \Sigma^{\leq i}$ such that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) \geq \tilde{f}_i(q, q')$. By Corollary A.6, $\mathcal{A}'$ is also good for $\tilde{f}_{i-1}$, and thus by Lemma A.10, $\tilde{f}_{i-1}(q, q') \geq$ *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i-1})$. Combining the last three inequalities we get that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) >$ *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i-1})$, which implies that $|w| = i$. If *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i}) = \infty$, we are done. Otherwise, it must be that $w \in L(\mathcal{A}^q) \cap L(\mathcal{A}'^{q'})$. Let $r = r_0, r_1, \cdots, r_i$ and $r' = r'_0, r'_1, \cdots, r'_i$ be accepting runs of minimal cost of $\mathcal{A}^q$ and $\mathcal{A}'^{q'}$, respectively, on w. Since $i \geq n^2$, there must be a pair of indices $0 \leq j < k \leq i$, such that $r_j = r_k$ and $r'_j = r'_k$. Let $w = xyz$, where $x = w_1 \cdots w_j$, $y = w_{j+1} \cdots w_k$, and $z = w_{k+1} \cdots w_i$. Observe that x and z may be empty, and that since $j < k$, it must be that $|y| > 0$. Also note that $\mathcal{A}^q$ can traverse x along $r_0 \cdots r_j$, traverse y along the cycle $C = r_j, \cdots, r_{k-1}$, and traverse z along $r_k \cdots r_i$. Similarly, $\mathcal{A}'^{q'}$ can traverse x along $r'_0 \cdots r'_j$, traverse y along the cycle $C' = r'_j, \cdots, r'_{k-1}$, and traverse z along $r'_k \cdots r'_i$.

By removing from r a traversal of C, and from r' a traversal of C', we derive accepting runs s and s' of $\mathcal{A}^q$ and $\mathcal{A}'^{q'}$, respectively, on the word xz. Since $|w| = i$, and $|y| > 0$, it follows that $|xz| < i$. Recall that $\mathcal{A}'$ is also good for $\tilde{f}_{i-1}$. Thus, by Lemma A.10, we have that $\tilde{f}_{i-1}(q, q') \geq cost(\mathcal{A}'^{q'}, xz) - cost(\mathcal{A}^q, xz)$. Recall that $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) \geq \tilde{f}_i(q, q') > \tilde{f}_{i-1}(q, q')$. Hence, $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}^q, w) > cost(\mathcal{A}'^{q'}, xz) - cost(\mathcal{A}^q, xz)$. Rearranging, we get $(cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}'^{q'}, xz)) - (cost(\mathcal{A}^q, w) - cost(\mathcal{A}^q, xz)) > 0$. Since $\mathcal{A}'^{q'}$ is deterministic, the run s' is the only run it has on xz, and thus $cost(\mathcal{A}'^{q'}, w) - cost(\mathcal{A}'^{q'}, xz) = c(r') - c(s') = c(C')$. On the other hand, since $\mathcal{A}^q$ is nondeterministic, s may not be a run of minimal cost over xz, and thus $cost(\mathcal{A}^q, w) - cost(\mathcal{A}^q, xz) \geq c(r) - c(s) = c(C)$. Combining the last three observations we get that $c(C') - c(C) > 0$.

For every $m > 0$, by adding m more traversals of the cycle C to r, we get an accepting run s_m of $\mathcal{A}^q$ on the word $xy^{m+1}z$. Similarly, by adding m more traversals of the cycle C' to r', we get an accepting run s'_m of $\mathcal{A}'^{q'}$ on the same word. It is not hard to see, using similar arguments to the ones used above, that $cost(\mathcal{A}'^{q'}, xy^{m+1}z) = c(s'_m) = c(r') + m \times c(C')$, and that $cost(\mathcal{A}^q, xy^{m+1}z) \leq$

$c(s_m) = c(r) + m \times c(C)$. Hence, $cost(\mathcal{A}'^{q'}, xy^{m+1}z) - cost(\mathcal{A}^q, xy^{m+1}z) \geq c(r') - c(r) + m \times (c(C') - c(C))$. Since this is true for arbitrarily large m, and we showed that $c(C') - c(C) > 0$, then $\mathit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \infty$. □

We are now ready to analyze the sequence of functions $f_0, f_1, \ldots, f_{2n^2-1}$, that our algorithm actually calculates. Recall that f_i is identical to $\tilde{f}_i$, except that for $i \geq n^2$ if $\max_{a\in\Sigma} f_i(q, q', a) > f_{i-1}(q, q')$, we set $f_i(q, q') = \infty$. For convenience, we recall the definition of the sequence $f_0, f_1, \ldots$ below. Also, to aid in the proof, the definition is extended to cover also indices i above $2n^2 - 1$.

- At initialization:
$$f_0(q, q') = \begin{cases} -\infty & \text{if } q \notin F \\ 0 & \text{if } q \in F \text{ and } q' \in F \\ \infty & \text{if } q \in F \text{ and } q' \notin F, \end{cases}$$
- For $1 \leq i \leq n^2 - 1$:
$$f_i(q, q') = \max\{f_{i-1}(q, q'), \max_{a\in\Sigma} f_i(q, q', a)\}.$$
- For $i \geq n^2$:
$$f_i(q, q') = \begin{cases} \infty & \text{if } \max_{a\in\Sigma} f_i(q, q', a) > f_{i-1}(q, q') \\ f_{i-1}(q, q') & \text{otherwise.} \end{cases}$$

In the above, for every $i \geq 0$ and $a \in \Sigma$, the function $f_i(q, q', a)$ is defined as follows.

$$f_i(q, q', a) = \min_{u'\in\rho_{i-1}(q',a)} \max_{u\in\delta(q,a)} f_{i-1}(u, u') + c(q', a, u') - c(q, a, u),$$

where the set $\rho_0(q', a) = \delta(q', a)$, and for $i > 0$, we have

$$\rho_i(q', a) = \{u' \in \rho_{i-1}(q', a) : \max_{u\in\delta(q',a)} f_{i-1}(u, u') + c(q', a, u') - c(q', a, u) \leq 0\}.$$

Observe that the definitions of "witnesses", and automata "good for", that we defined for $\tilde{f}$, are easily carried over to f. We first show that it is enough to calculate (at most) the functions $f_0, f_1, \ldots, f_{2n^2-1}$, as the sequence $f_0, f_1, \ldots$ reaches a fixed-point within at most $2n^2$ iterations.

Proposition A.14 *There is some $j < 2n^2$ such that $f_i = f_j$ for every $i \geq j$.*

Proof: Note that for every $i \geq n^2$, if the value of $f_i(q, q')$ differs from that of $f_{i-1}(q, q')$ then it attains the maximal value of ∞. Hence, since there are n^2 pairs of states, the sequence f_i must reach a fixed-point within $2n^2$ iterations. □

It is easy to see that f, just like $\tilde{f}$, is monotonically increasing with i. Below we show that f shares with $\tilde{f}$ the ability to precisely quantify the cost difference between $\mathcal{A}$ and deterministic automata embodied in $\mathcal{A}$. However, unlike $\tilde{f}_i$ which considers words of length at most i, f_i may also consider words of unbounded length. We start by making a couple of easy observations.

Proposition A.15 *for every $i \geq 0$, and every $q, q' \in Q$, we have $f_i(q,q') \geq \tilde{f}_i(q,q')$. Furthermore, if $f_i(q,q') \neq \tilde{f}_i(q,q')$, then $f_i(q,q') = \infty$.*

Proof: Immediate from the definitions of $\tilde{f}$ and f. □

Lemma A.16 *For every $i \geq 0$, and $q, q' \in Q$, we have that $f_i(q,q') = -\infty$ iff $L(\mathcal{A}^q) \cap \Sigma^{\leq i} = \emptyset$.*

Proof: Since $\mathcal{A}$ has n states, for every $i > n$ we have that $L(\mathcal{A}^q) \cap \Sigma^{\leq i} = \emptyset$ iff $L(\mathcal{A}^q) \cap \Sigma^{\leq n} = \emptyset$. Since $\tilde{f}_i$ and f_i coincide for $i \leq n$, the result follows from Lemma A.2. □

Proposition A.13 shows that if $\mathcal{A}'$ is good for $\tilde{f}_i$ and q' is reachable in $\mathcal{A}'$, then for $i \geq n^2$ an increase in the value of $\tilde{f}_i(q,q')$ compared to $\tilde{f}_{i-1}(q,q')$ indicates that *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \infty$. This observation was the intuition behind our definition of f. Note, however, that since the value of f_i depends on f_{i-1} and not on $\tilde{f}_{i-1}$, it is not clear that the above analysis carries over to f. In particular, Lemma A.9 which served a crucial role in the proof of Lemma A.10, and therefore also in the proof of the pumping argument of Proposition A.13, is not true for f. Recall that Lemma A.9 states that if $\tilde{f}_i(q,p), \tilde{f}_i(p,q') \in \mathbb{R}$, then so is $\tilde{f}_i(q,q')$, and $\tilde{f}_i(q,p) + \tilde{f}_i(p,q') \geq \tilde{f}_i(q,q')$. However, for f, if $i \geq n^2$ it may be that $f_i(q,p), \tilde{f}_i(p,q') \in \mathbb{R}$, but that $f_i(q,q')$ was bumped up to ∞. Fortunately, as the next lemma shows, even after many iterations where f and $\tilde{f}$ may have attained different values for many pairs, if f sets a value of ∞ to a certain pair, then its decision is justifiable, since $\tilde{f}$ would also (in future iterations) grow without bounds, or attain the value ∞. Thus, in a sense, "at the limit" $\tilde{f}$ and f behave in the same way.

Lemma A.17 *Given $i \geq 0$, and $q, q' \in Q$, such that $f_i(q,q') = \infty$. If q' is reachable in some $\mathcal{A}'$ that is good for $\tilde{f}$, then for every $m \in \mathbb{N}$ there is an index $k_m \geq 0$ such that $\tilde{f}_k(q,q') > m$ for every $k \geq k_m$.*

Proof: Observe that since $\tilde{f}$ is monotonically increasing it is enough to show that for every $m \in \mathbb{N}$ there is an index $k_m \geq 0$ such that $\tilde{f}_{k_m}(q,q') > m$. We prove the lemma by an induction on i. For $i < n^2$, $f_i(q,q') = \infty$ implies that $\tilde{f}_i(q,q') = \infty$. Assume now that $i \geq n^2$, and that the lemma holds for $i-1$. Note that if $\tilde{f}_i(q,q') = \infty$, we are done, and that $\tilde{f}_i(q,q') = -\infty$ is impossible by Lemmas A.2 and A.16. We thus assume that $\tilde{f}_i(q,q') \in \mathbb{R}$. Also note that if $f_i(q,q') = f_{i-1}(q,q')$ then the lemma holds by the induction hypothesis. Hence, from now on we also assume that $f_{i-1}(q,q') < f_i(q,q')$.

We now prove that if there is an index $k \geq i$ such that $\tilde{f}_k(q,q') > \tilde{f}_{k-1}(q,q')$ then the lemma holds. Assume for now that such a k exists (we will later show that indeed it does). By our assumption, there is an automaton $\mathcal{A}'$ that is good for $\tilde{f}$, in which q' is reachable. Since in particular $\mathcal{A}'$ is good for $\tilde{f}_k$, and since $k \geq n^2$, then by Proposition A.13 we have *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \infty$. It follows that for every $m \in \mathbb{N}$ there is an index $k_m \geq 0$, such that *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq j}) > m$ for every $j \geq k_m$. On the other hand, by Lemma A.10, for every such j we have that $\tilde{f}_j(q,q') \geq$ *costdiff*$(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq j})$, which completes our argument. It remains to show that indeed there is an index $k \geq i$ such that $\tilde{f}_k(q,q') > \tilde{f}_{k-1}(q,q')$. Observe that since $\tilde{f}$ is monotonically increasing it is enough to find an index $k \geq i$ such that $\tilde{f}_k(q,q') \neq \tilde{f}_{i-1}(q,q')$.

If $\tilde{f}_i(q,q') \neq \tilde{f}_{i-1}(q,q')$ then we are done. Assume then that $\tilde{f}_i(q,q') = \tilde{f}_{i-1}(q,q')$. Recall that by our assumption $f_{i-1}(q,q') < f_i(q,q')$. Thus, by definition, we have that there is an

$a \in \Sigma$ such that $f_i(q,q',a) > f_{i-1}(q,q')$. Assume first that there is an index $k \geq i$ for which $\tilde{f}_k(q,q',a) \geq f_i(q,q',a)$. Since, by definition, $\tilde{f}_k(q,q') \geq \tilde{f}_k(q,q',a)$, the previous inequalities imply that $\tilde{f}_k(q,q') > f_{i-1}(q,q')$. Thus, by Proposition A.15, we have that $\tilde{f}_k(q,q') > \tilde{f}_{i-1}(q,q')$, and we are done.

Assume now that for all $k \geq i$ we have $\tilde{f}_k(q,q',a) < f_i(q,q',a)$. Let us first prove that $\tilde{\rho}(q',a) \subseteq \rho_i(q',a)$. Observe that $\tilde{\rho}(q',a) \subseteq \tilde{\rho}_i(q',a)$, and that Proposition A.15 implies that $\rho_i(q',a) \subseteq \tilde{\rho}_i(q',a)$. Hence, it is enough to show that $(\tilde{\rho}_i(q',a) \setminus \rho_i(q',a)) \cap \tilde{\rho}(q',a) = \emptyset$. Assume by way of contradiction that there is a $u' \in (\tilde{\rho}_i(q',a) \setminus \rho_i(q',a)) \cap \tilde{\rho}(q',a)$. Since u' is in $\tilde{\rho}_i(q',a)$ but not in $\rho_i(q',a)$, there must be a state $v' \in \delta(q',a)$ such that $f_{i-1}(v',u') \neq \tilde{f}_{i-1}(v',u')$. Thus, by Proposition A.15, $f_{i-1}(v',u') = \infty$. Recall that $\mathcal{A}'$ is good for $\tilde{f}$, that q' is reachable in $\mathcal{A}'$, and that $u' \in \tilde{\rho}(q',a)$. Hence, by Corollary A.8, there is an $\mathcal{A}''$ that is good for $\tilde{f}$, in which u' is reachable. We can thus apply the induction hypothesis to $f_{i-1}(v',u')$, and obtain that there is an index $k \geq 0$ such that $\tilde{f}_k(v',u') > c(q',a,v') - c(q',a,u')$. But this is a contradiction since $u' \in \tilde{\rho}(q',a)$ implies that $u' \in \tilde{\rho}_k(q',a)$, and thus $\tilde{f}_k(v',u') + c(q',a,u') - c(q',a,v') \leq 0$. It follows that our claim that $\tilde{\rho}(q',a) \subseteq \rho_i(q',a)$ is true.

We are now ready to show that there is an index $k \geq i$ such that $\tilde{f}_k(q,q') > \tilde{f}_{i-1}(q,q')$. Given $k \geq i$, let $MIN_k = \{u' \in \tilde{\rho}_{k-1}(q',a) \mid \max_{u \in \delta(q,a)} \tilde{f}_{k-1}(u,u') + c(q',a,u') - c(q,a,u) = \tilde{f}_k(q,q',a)\}$, be the set of states for which the minimum in the expression for $\tilde{f}_k(q,q',a)$ is achieved. We claim that for every $k \geq i$ the set MIN_k is not empty. To see that, recall that $f_i(q,q',a) > f_{i-1}(q,q')$, and thus $f_i(q,q',a) \neq -\infty$. It follows, by the definition of $f_i(q,q',a)$, that there is $u \in \delta(q,a)$ such that $L(\mathcal{A}^u) \cap \Sigma^{\leq i-1} \neq \emptyset$. Thus, if $\tilde{\rho}_{k-1}(q',a) = \emptyset$, then by definition $\tilde{f}_k(q,q',a) = \infty$. But since $k \geq i$, by our assumption $\tilde{f}_k(q,q',a) < f_i(q,q',a)$, which is a contradiction. Hence, for every $k \geq i$ we have $MIN_k \neq \emptyset$. Since $\mathcal{A}$ has only finitely many states, there is a state u' such that $u' \in MIN_k$ for infinitely many k's. Observe that it follows that $u' \in \tilde{\rho}(q',a)$. Let $t \geq i$ be such that $u' \in MIN_t$. Recall that since $t \geq i$, by our assumption $f_i(q,q',a) > \tilde{f}_t(q,q',a)$. Thus:

$$\begin{aligned} f_i(q,q',a) &= \min_{v' \in \rho_{i-1}(q',a)} \max_{u \in \delta(q,a)} f_{i-1}(u,v') + c(q',a,v') - c(q,a,u) \\ &> \tilde{f}_t(q,q',a) \\ &= \max_{u \in \delta(q,a)} \tilde{f}_{t-1}(u,u') + c(q',a,u') - c(q,a,u) \end{aligned}$$

Recall that we showed that $\tilde{\rho}(q',a) \subseteq \rho_i(q',a)$, that $\rho_i(q',a) \subseteq \rho_{i-1}(q',a)$, and that $u' \in \tilde{\rho}(q',a)$. It follows that $u' \in \rho_{i-1}(q',a)$. Hence, the inequality above implies that there is a state $u \in \delta(q,a)$, such that $f_{i-1}(u,u') > \tilde{f}_{t-1}(u,u')$. Since $t \geq i$, and $\tilde{f}$ is monotonically increasing, then $\tilde{f}_{t-1}(u,u') \geq \tilde{f}_{i-1}(u,u')$. It follows that $f_{i-1}(u,u') > \tilde{f}_{i-1}(u,u')$, and thus by Proposition A.15, $f_{i-1}(u,u') = \infty$. Recall that $\mathcal{A}'$ is good for $\tilde{f}$, that q' is reachable in $\mathcal{A}'$, and that $u' \in \tilde{\rho}(q',a)$. Hence, by Corollary A.8, there is an $\mathcal{A}''$ that is good for $\tilde{f}$ in which u' is reachable. We can thus apply the induction hypothesis to $f_{i-1}(u,u')$ and get that for every $h \in \mathbb{N}$ there is an index $l_h \geq 0$ such that $\tilde{f}_l(u,u') > h$ for every $l \geq l_h$. Recall that by our assumptions $\tilde{f}_i(q,q') \in R$, and $\tilde{f}_i(q,q') = \tilde{f}_{i-1}(q,q')$. Thus, we can choose an h such that $h \geq \tilde{f}_{i-1}(q,q') - (c(q',a,u') - c(q,a,u))$. Since $u' \in MIN_k$ for infinitely many k's, we can find an index $k_h > l_h$, such that $u' \in MIN_{k_h}$. It

follows that:

$$\begin{aligned}\tilde{f}_{k_h}(q,q') \geq \tilde{f}_{k_h}(q,q',a) &= \max_{v\in\delta(q,a)} \tilde{f}_{k_h-1}(v,u') + c(q',a,u') - c(q,a,v) \\ &\geq \tilde{f}_{k_h-1}(u,u') + c(q',a,u') - c(q,a,u) \\ &> h + c(q',a,u') - c(q,a,u) \geq \tilde{f}_{i-1}(q,q')\end{aligned}$$

Which completes the proof. □

The next lemma shows that once the calculation of f reaches a fixed-point, the set of automata that are good for f at this fixed-point is exactly the set of automata that are good for $\tilde{f}$.

Lemma A.18 *If j is a fixed-point index of f, then an automaton $\mathcal{A}'$ embodied in $\mathcal{A}$ is good for f_j iff it is good for $\tilde{f}$.*

Proof: It is not hard to see that since j is a fixed-point index of f, then $\mathcal{A}'$ is good for f_j iff it is good for f. Hence, we can prove instead that $\mathcal{A}'$ is good for f iff it is good for $\tilde{f}$. For the first direction, assume that $\mathcal{A}'$ is good for f. Looking at the definition of $\tilde{\rho}(q,a)$, one can see that Proposition A.15 implies that for every $a \in \Sigma$ and every $q \in Q$, we have that $\rho(q,a) \subseteq \tilde{\rho}(q,a)$. Hence, $\mathcal{A}'$ is also good for $\tilde{f}$. For the other direction, assume that $\mathcal{A}'$ is good for $\tilde{f}$, and let q' be a state reachable in $\mathcal{A}'$. We have to show that for every $a \in \Sigma$, if $u' = \delta'(q',a)$, then $u' \in \rho(q',a)$. Observe that by the definition of $\rho(q',a)$ it is enough to show that for every $u \in \delta(q',a)$ and every $i \geq 0$, we have that $\tilde{f}_i(u,u') = f_i(u,u')$. Assume by way of contradiction that there is a $u \in \delta(q',a)$ such that $\tilde{f}_i(u,u') \neq f_i(u,u')$. By Proposition A.15, we have that $f_i(u,u') = \infty$. Applying Lemma A.17 to $f_i(u,u')$, we get that there is an index $k \geq 0$ such that $\tilde{f}_k(u,u') + c(q',a,u') - c(q,a,u) > 0$. It follows that $u' \notin \tilde{\rho}_k(q',a)$. But this contradicts the fact that $\mathcal{A}'$ is good for $\tilde{f}_k$, and our claim is proved. □

The next lemma shows that at a fixed-point index j, the function f_j quantifies exactly the cost difference, over all words in Σ^*, between $\mathcal{A}$ and any deterministic automaton that is good for f_j .

Lemma A.19 *If $\mathcal{A}'$ is good for f_j, where j is a fixed-point index of f, then for every $q, q' \in Q$, such that q' is reachable in $\mathcal{A}'$, we have that $f_j(q,q') = \textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*)$.*

Proof: Consider first the case where $f_j(q,q') = -\infty$. Since j is a fixed-point index, it follows that for every $i \geq j$ we also have $f_i(q,q') = -\infty$. Hence, by Lemma A.16, we have that $L(\mathcal{A}^q) \cap \Sigma^* = \emptyset$. It follows (by definition) that $\textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = -\infty$, and the lemma holds. Consider now the case where $f_j(q,q') \in \mathbb{R}$. Given $i \geq j$, since j is a fixed-point index of f, then $f_j(q,q') = f_i(q,q')$. By proposition A.15, we have that $f_j(q,q') = f_i(q,q') = \tilde{f}_i(q,q') \in \mathbb{R}$. By Lemma A.18, $\mathcal{A}'$ is good for $\tilde{f}_i$. Hence, by Lemmas A.10 and A.11, we have that $\textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^{\leq i}) = \tilde{f}_i(q,q')$. Since this is true for every $i \geq j$ then $\textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \tilde{f}_i(q,q')$. Since $\tilde{f}_i(q,q') = f_i(q,q')$ then also in this case the lemma holds. It is left to consider the case where $f_j(q,q') = \infty$. By Lemma A.18, $\mathcal{A}'$ is good for $\tilde{f}$. Thus, by Lemma A.17, the sequence $\tilde{f}_0(q,q'), \tilde{f}_1(q,q'), \ldots$ is unbounded. By Lemma A.11, it follows that $\textit{costdiff}(\mathcal{A}'^{q'}, \mathcal{A}^q, \Sigma^*) = \infty$. □

We can now prove the following theorem which together with Proposition A.14 implies Theorem 4.5.

Theorem A.20 *A WFA $\mathcal{A} = \langle \Sigma, Q, \Delta, c, Q_0, F \rangle$ is DBP iff for a fixed-point index j of f, there is a state $q'_0 \in Q_0$ such that for every $q_0 \in Q_0$ we have that $f_j(q_0, q'_0) \leq 0$.*

Proof: For the first direction, assume that $\mathcal{A}$ is DBP, and take some $\mathcal{A}' = \langle \Sigma, Q, \Delta', c, q'_0, F \rangle$ in $\det_0(\mathcal{A})$. By Lemma A.12, $\mathcal{A}'$ is good for $\tilde{f}$, and thus, by Lemma A.18, it is also good for f_j. By Lemma A.19, for every $q_0 \in Q_0$, we have that $f_i(q_0, q'_0) = \textit{costdiff}(\mathcal{A}'^{q'_0}, \mathcal{A}^{q_0}, \Sigma^*)$. Since $\mathcal{A}'$ is equivalent to $\mathcal{A}$ it must be that for every $q_0 \in Q_0$ we have that $\textit{costdiff}(\mathcal{A}'^{q'_0}, \mathcal{A}^{q_0}, \Sigma^*) \leq 0$.

For the other direction, assume that there is a state $q'_0 \in Q_0$, such that for every $q_0 \in Q_0$ we have that $f_j(q_0, q'_0) \leq 0$. Take some $\mathcal{A}'$ that is good for $\tilde{f}$ (we will later show that there is such an $\mathcal{A}'$). By Lemma A.18, $\mathcal{A}'$ is also good for f_j. Thus, by Lemma A.19, for every $q_0 \in Q_0$ we have that $\textit{costdiff}(\mathcal{A}'^{q'_0}, \mathcal{A}^{q_0}, \Sigma^*) = f_j(q_0, q'_0)$. Since by our assumption, for every $q_0 \in Q_0$ we have that $f_j(q_0, q'_0) \leq 0$, then $\textit{costdiff}(\mathcal{A}'^{q'_0}, \mathcal{A}^{q_0}, \Sigma^*) \leq 0$, and $\mathcal{A}'$ must be equivalent to $\mathcal{A}$, and thus $\mathcal{A}$ is DBP. It remains to show that indeed there is an automaton $\mathcal{A}' = \langle \Sigma, Q, \Delta', c, q'_0, F \rangle$ that is good for $\tilde{f}$.

To build $\mathcal{A}'$ we start without any transitions and iteratively add transitions as follows: For every state q that is reachable from q'_0, and every $a \in \Sigma$ such that $\delta(q, a) \neq \emptyset$ but $\delta'(q, a) = \emptyset$, we arbitrarily chose some $u \in \tilde{\rho}(q, a)$ and add the transition $\langle q, a, u \rangle$. It is not hard to see that if we never run into a situation where $\tilde{\rho}(q, a) = \emptyset$ then we end up with an automaton that is good for $\tilde{f}$. The fact that throughout the construction we always have $\tilde{\rho}(q, a) \neq \emptyset$ is proved by an induction on the distance of q from q'_0. For the induction base ($q = q'_0$), we have to show that for every $i \geq 0$, if $\delta(q'_0, a) \neq \emptyset$ then $\tilde{\rho}_i(q'_0, a) \neq \emptyset$. The case $i = 0$ is true since by definition $\tilde{\rho}_0(q'_0, a) = \delta(q'_0, a)$. Given $i > 0$, recall that by our assumption, for every $q_0 \in Q_0$ we have that $f_j(q_0, q'_0) \leq 0$. Thus, in particular, $f_j(q'_0, q'_0) \leq 0$, which implies (since j is a fixed-point index, and f is monotonically increasing) that $f_i(q'_0, q'_0) \leq f_j(q'_0, q'_0) \leq 0$. It follows, by Proposition A.15, that $\tilde{f}_i(q'_0, q'_0) \leq 0$, and thus also $\tilde{f}_i(q'_0, q'_0, a) \leq 0$. Observe that if $\delta(q'_0, a) \neq \emptyset$, then by the definitions of $\tilde{f}_i(q'_0, q'_0, a)$ and $\tilde{\rho}_i(q'_0, a)$, either $\tilde{\rho}_i(q'_0, a) \neq \emptyset$, or for all states $u \in \delta(q'_0, a)$ we have that $L(\mathcal{A}^{q'_0}) \cap \Sigma^{\leq i-1} = \emptyset$. In the latter case, by Lemma A.2 and the definition of $\tilde{\rho}_i(q'_0, a)$, it must be that $\delta(q'_0, a) = \tilde{\rho}_i(q'_0, a)$, which completes the proof of the induction base. The induction step follows directly from Proposition A.4. □

Observe that in the proof of Theorem A.20, we actually show that if $\mathcal{A}$ is DBP, then $det_0(\mathcal{A})$ is exactly the set of automata that are good for a fixed-point of f.