Coping with Selfish On-going Behaviors

Orna Kupferman* Tami Tamir†

Abstract

A rational and selfish environment may have an incentive to cheat the system it interacts with. Cheating the system amounts to reporting a stream of inputs that is different from the one corresponding to the real behavior of the environment. The system may cope with cheating by charging penalties to cheats it detects. In this paper, we formalize this setting by means of weighted automata and their resilience to selfish environments. Automata have proven to be a successful formalism for modeling the on-going interaction between a system and its environment. In particular, weighted finite automata (WFAs), which assign a cost to each input word, are useful in modeling an interaction that has a quantitative outcome. Consider a WFA $\mathcal{A}$ over the alphabet Σ. At each moment in time, the environment may cheat $\mathcal{A}$ by reporting a letter different from the one it actually generates. A penalty function $\eta : \Sigma \times \Sigma \to \mathbb{R}^{\geq 0}$ maps each possible false-report to a penalty, charged whenever the false-report is detected. A detection-probability function $p : \Sigma \times \Sigma \to [0,1]$ gives the probability of detecting each false-report. We say that $\mathcal{A}$ is (η, p)-resilient to cheating if $\langle \eta, p \rangle$ ensures that the minimal expected cost of an input word is achieved with no cheating. Thus, a rational environment has no incentive to cheat $\mathcal{A}$.

We study the basic problems arising in the analysis of this setting. In particular, we consider the problem of deciding whether a given WFA $\mathcal{A}$ is (η, p)-resilient with respect to a given penalty function η and a detection-probability function p; and the problem of achieving resilience with minimum resources, namely, given $\mathcal{A}$ and η, finding the minimal (with respect to $\sum_{\sigma,\sigma'} \eta(\sigma,\sigma') \cdot p(\sigma,\sigma')$) detection-probability function p, such that $\mathcal{A}$ is (η, p)-resilient. While for general WFAs both problems are shown to be PSPACE-hard, we present polynomial-time algorithms for deterministic WFAs.

1 Introduction

The environment of modern systems often consists of other systems, having objectives of their own. For example, an e-commerce applications interacts with sellers and buyers. A seller may provide a non-reliable description of the goods he is selling. Furthermore, sellers may provide false feedback and twisted rating of their competitors. Buyers may commit to some transaction but not accomplish it, or may provide a bid that is lower than the real value they are willing to pay, hoping to win even with it. As another example, the environment of various service-providing systems are clients that wish to minimize their payment. Clients' payments may be based on their self-reports, which are usually screened but may be false. In the same way, biased users may affect the quality of recommendation systems for various products or services.

The above examples demonstrate the fact that environments have two types of behaviors: the *truthful* behavior – the one they would produce if they follow their protocol, and the *reported* behavior – the one they actually output, hoping it would lead to a better outcome for them. While the design of systems cannot assume that the environment would take its truthful behavior, we can assume that environments are *rational*, in the sense they always take a behavior that maximizes their outcome.

*School of Engineering and Computer Science, Hebrew University, Jerusalem, Israel. E-mail: orna@cs.huji.ac.il

†School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. E-mail:tami@idc.ac.il

Mechanism design is a field in game theory and economics studying the design of games for rational players. A game is *incentive compatible* if no player has an incentive to deviate from his truthful behavior [NR99, NRTV07]. The outcome of traditional games depend on the final position of the game. In contrast, the systems we want to reason about maintain an *on-going interaction* with their environment [HP85], and reasoning about their behavior refer not to their final state (in fact, much of the research in the area considers non-terminating systems, with no final state) but rather to the *language* of computations that they generate. In [FKL10], the authors study *rational synthesis*, where the synthesized systems are guaranteed to satisfy their specifications when they interact with rational environments (rather than with hostile environments that do not have objectives other than to fail the system [PR89]). In this paper, we suggest and study a possible model for reasoning about incentive capacity in the context of on-going behaviors and quantitative properties, or formal power series. Reporting of trustworthy information is an essential component also in service-providing systems.

Automata have proven to be a successful formalism for modeling on-going behaviors. Consider a system with a set P of atomic propositions. Each assignment to the atomic propositions corresponds to a letter σ in the alphabet 2^P. Accordingly, a computation of the system, which is a sequence of such assignments, is a word over the alphabet 2^P, and a specification for the system is a language over this alphabet, describing the desired properties of the system. By translating specifications to automata, it is possible to reduce questions about systems and their specifications to questions about automata [VW94]. For example, a system S satisfies a specification ψ if the language that contains exactly all the computations generated by S is contained in the language of an automaton that accepts exactly all words satisfying ψ.

A boolean language maps words to true or false. A *qualitative language* maps words to values from a richer domain [CCH+05, Hen07]. A *Weighted automaton* $\mathscr{A}$ on finite words (WFAs, for short) [Eil74, SS78, Moh97, DKe09] defines a quantitative language $L : \Sigma^* \to \mathbb{R}^{\geq 0} \cup \{\infty\}$. Technically, each transition of $\mathscr{A}$ has a traversal cost, each state has an acceptance cost, and the cost of a run is the sum of the costs of the transitions taken along the run plus the acceptance cost of its last state. The cost of a word is then the minimum cost over all runs on it (note that the cost may be infinite).

A rational and selfish environment may have an incentive to cheat the WFA and report a word different from the one generated by its truthful behavior. The WFA may cope with cheating by charging penalties to cheats it detects. Formally, at each moment in time, the environment may cheat the WFA by reporting a letter different from the one its truthful behavior generates. A *detection-probability function* $p : \Sigma \times \Sigma \to [0,1]$ gives the probability of detecting each false-report. A *penalty function* $\eta : \Sigma \times \Sigma \to \mathbb{R}^{\geq 0}$ gives the penalty charged whenever a particular false-report is detected. Thus, when the environment reports that a letter σ is σ', then the WFA detects the cheating with probability $p(\sigma, \sigma')$, in which case the environment is charged $\eta(\sigma, \sigma')$. The expected cost of a word w is then the minimum (over all words w' of the same length as w) cost of w' plus the expected cost of reporting w to be w'. We say that a WFA $\mathscr{A}$ is (η, p)-*resilient to cheating* if $\langle \eta, p \rangle$ ensures that, for all words, the above minimal expected cost is achieved in a cheat-free run. Thus, a dominant strategy for the environment is one that does not cheat.

We study the basic problems arising in the analysis of this setting. First, we observe that, by linearity of expectation, a detection probability function p and a penalty function η can be combined to a single *expected-fee* function $\theta = \eta \circ p$; that is, for all $\sigma, \sigma' \in \Sigma$, we have $\theta(\sigma, \sigma') = \eta(\sigma, \sigma') \cdot p(\sigma, \sigma')$. Accordingly, we can study θ-resilience, which simplifies the probabilistic reasoning. Second, we make use of

the fact it is possible to construct, given a WFA $\mathcal{A}$ and an expected-fee function θ, a WFA $\mathit{Cheat}(\mathcal{A},\theta)$ that takes cheating into account and in which the cost of a word is its minimal possible cost (achieved by a best cheating strategy). We show that θ-resilience to cheating is a semantic property. Thus, given a weighted language $L : \Sigma^* \to \mathbb{R}^{\geq 0} \cup \{\infty\}$, and a penalty function θ, then either all WFAs for $\mathcal{A}$ are θ-resilient to cheating, or none of them is. It follows that the natural problem of translating a given WFA $\mathcal{A}$ that need not be θ-resilient to cheating to an equivalent WFA that is θ-resilient to cheating is not interesting, as equivalent WFAs have the same resilience.

With these observations and constructions, we turn to study the practical problems of the setting. From the environment's point of view, we consider the problem of finding, given $\mathcal{A}$, θ, and a word $w \in \Sigma^*$, a word w' such that the environment can minimize the cost of w in $\mathcal{A}$ by reporting it to be w'. We show that the problem can be reduced to the problem of finding a shortest path in a graph, which can be solved in polynomial time [Dij59].

We then turn to study problems from the designer's point of view. We start with the problem of deciding whether a given WFA $\mathcal{A}$ is θ-resilient to cheating with respect to a given expected fee function θ. We show that the problem is PSPACE-hard, but present a polynomial-time solution for the case $\mathcal{A}$ is deterministic. Our solution is based on dynamic programming, taking into account words of increasing lengths. In particular, we show that cycles along which cheating is beneficial (and can therefore lead to an unbounded incentive to cheat) can be detected after quadratically many iterations.

A system with no limits on penalties and with unbounded resources can prevent cheating by fixing a high expected-fee function. In practice, penalties may be limited by an external authority, and increasing the probability of detecting cheats requires resources. Consider a WFA $\mathcal{A}$ and two expected-fee functions θ_1 and θ_2 such that $\theta_1 \leq \theta_2$ (that is $\theta_1(\sigma,\sigma') \leq \theta_2(\sigma,\sigma')$ for all $\sigma,\sigma' \in \Sigma$). If $\mathcal{A}$ is θ_1-resilient to cheating, then $\mathcal{A}$ is clearly also θ_2-resilient to cheating, yet θ_1 achieves resilience more efficiently. In particular, θ_1 can be obtained from θ_2 by reducing the probability of cheat detection, hence saving on resources required for cheat detection. Recall that $\theta = \eta \circ p$, for a penalty function η and a detection probability function p. Assuming that the penalty function η is determined by an external authority, and that system's resources are allocated to increase the detection probability, we consider the following problem of *minimal resources resilience*: Given a WFA $\mathcal{A}$ and a penalty function η, find a probability detection function p such that $\mathcal{A}$ is $(\eta \circ p)$-resilient, and the detection budget, given by $\sum_{\sigma,\sigma'} \eta(\sigma,\sigma')p(\sigma,\sigma')$, is minimal. Note that the probabilities in our objective function are weighted by η. This reflects the fact that detecting a cheat with a high penalty tends to require high resources. Indeed, in practice, the higher is the responsibility of a guard, the higher is his salary. We study the minimal resources resilience problem and show that it is PSPACE-hard. As in resilience testing, the problem is easier in the deterministic case, for which we present a polynomial-time solution, based on describing the problem as a linear program. Essentially, the constraints of the linear program are induced by the restrictions used in the testing algorithm, with the expected-fee values being variables. The same method can be used in order to solve additional minimal-budget problems, with any desired linear objective function over the detection-probability function or the penalty function.

We also consider two variants of the setting. In the *rising-penalty* variant, the expected penalty for cheating increases with the number of cheats. This variant reflects the realistic response of systems to user's false report: allocating more resources to cheat detection, or formally, increasing the detection probability with each detected cheat. In the *bounded cheating* variant the number of times the environ-

ment can cheat or the total budget it can invest in penalties is bounded.

2 Preliminaries

In this section we give a formal description of the model we consider, and present several observations and constructions that will be used throughout the paper.

2.1 Weighted Finite Automaton

Given an alphabet Σ, a weighted language is a function $\mathscr{L} : \Sigma^* \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ mapping each word in Σ^* to a positive (possibly ∞) cost. A *weighted finite automaton* (WFA, for short) is $\mathscr{A} = \langle \Sigma, Q, \Delta, c, Q_0, \tau \rangle$, 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 $\tau : Q \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ is a final cost function. A transition $d = \langle q, \sigma, p \rangle \in \Delta$ (also written $\Delta(q, \sigma, p)$) can be taken when reading the input letter $\sigma \in \Sigma$, and it causes $\mathscr{A}$ to move from state q to state p with *cost* $c(d)$. The transition relation Δ induces a transition function $\delta : Q \times \Sigma \to 2^Q$, where for a state $q \in Q$ and a letter $\sigma \in \Sigma$, we have $\delta(q, \sigma) := \{p : \Delta(q, \sigma, p)\}$. 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 $\delta(q, \varepsilon) = q$, and $\delta(q, u \cdot \sigma) := \delta(\delta(q, u), \sigma)$, for every $u \in \Sigma^*$ and $\sigma \in \Sigma$.

Note that a WFA $\mathscr{A}$ may be nondeterministic in the sense that it may have many initial states, and the transition function may lead to several successor states. If $|Q_0| = 1$ and for every state $q \in Q$ and letter $\sigma \in \Sigma$ we have $|\delta(q, \sigma)| \leq 1$, then $\mathscr{A}$ is a *deterministic* WFA (for short, DWFA).

For a word $w = w_1 \dots w_n \in \Sigma^*$, a run of $\mathscr{A}$ on w is a sequence $r = r_0, r_1, \dots, r_n \in Q^{n+1}$, where $r_0 \in Q_0$ and for every $1 \leq i \leq n$, we have $\Delta(r_{i-1}, w_i, r_i)$. The cost of a run is the sum of the costs of the transitions that constitute the run, along with the final cost. [1] Formally, let $r = r_0, r_1, \dots, r_n$ be a run of $\mathscr{A}$ on w, and let $d = d_1 \dots d_n \in \Delta^*$ be the corresponding sequence of transitions. The cost of r is $cost(\mathscr{A}, r) = \sum_{i=1}^{n} c(d_i) + \tau(r_n)$. For two indices $1 \leq j_1 < j_2 \leq n$, we use $cost(\mathscr{A}, r, j_1, j_2)$ to denote the cost of the sub-run leading from $q_{j_1 - 1}$ to q_{j_2}. Thus, $cost(\mathscr{A}, r, j_1, j_2) = \sum_{i=j_1}^{j_2} c(d_i)$ The cost of w in $\mathscr{A}$, denoted $cost(\mathscr{A}, w)$, is the minimal cost over all runs of $\mathscr{A}$ on w. Thus, $cost(\mathscr{A}, w) = \min\{cost(\mathscr{A}, r) : r \text{ is an accepting run of } \mathscr{A} \text{ on } w\}$. Note that while WFAs do not have a set of acceptance states, runs that reach states q for which $\tau(q) = \infty$ have cost ∞, thus the function τ can be viewed as a refinement of the partition of the state space to accepting and rejecting states. The weighted language of $\mathscr{A}$, denoted $L(\mathscr{A})$, maps each word $w \in \Sigma^*$ to $cost(\mathscr{A}, w)$.

We assume that all states $q \in Q$ are reachable in $\mathscr{A}$. We assume that all states, except maybe the initial states are not empty, in the sense they map at least one word to a finite cost. Thus, for all $q \in Q$ there is $w \in \Sigma^*$ such that the cost of w in $\mathscr{A}$ with initial state q is in $\mathbb{R}$. Finally, given two WFAs $\mathscr{A}$ and $\mathscr{A}'$, we say that $\mathscr{A}$ is cheaper than $\mathscr{A}'$, denoted $\mathscr{A} \preceq \mathscr{A}'$, if for every word $w \in \Sigma^*$, we have that $cost(\mathscr{A}, w) \leq cost(\mathscr{A}', w)$.

[1] 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 a word is the semiring sum over all runs on it. For our purposes, 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.

2.2 Input Cheating and Resilience of Automata

Recall that a WFA induces a weighted language that maps each word to a cost in $\mathbb{R}^{\geq 0} \cup \{\infty\}$. Words may cheat the automaton hoping to be mapped to a lower cost: When the automaton runs on a word $w = w_1 \dots w_n \in \Sigma^*$, then in each position $1 \leq i \leq n$, the word can cheat the automaton and report that the letter w_i is a different letter $w'_i \in \Sigma$. Cheating has a price, and the setting includes a *penalty function* $\eta : \Sigma \times \Sigma \to \mathbb{R}^{\geq 0}$, satisfying $\eta(\sigma, \sigma) = 0$, and a *detection-probability function* $p : \Sigma \times \Sigma \to [0,1]$ indicating the probability of catching each specific cheat. Formally, whenever σ is reported to be σ', the automaton detects the cheating with probability $p(\sigma, \sigma')$, in which case it charges $\eta(\sigma, \sigma')$. The expected penalty for reporting σ to be σ' is therefore $\eta(\sigma, \sigma') \cdot p(\sigma, \sigma')$.

For two words $w = w_1, w_2, \dots, w_n$ and $w' = w'_1, w'_2, \dots, w'_n$, the expected cost of reporting w to be w' is $\sum_{i=1}^{n} \eta(w_i, w'_i) \cdot p(w_i, w'_i)$. Given a WFA $\mathcal{A}$, a penalty function η, a detection-probability function p, and two words w, w' such that $|w| = |w'|$, the expected cost of w in $\mathcal{A}$ when w is reported to be w', denoted $\mathit{expected\_faked\_cost}(\mathcal{A}, \eta, p, w, w')$, is $\mathit{cost}(\mathcal{A}, w') + \sum_{i=1}^{n} \eta(w_i, w'_i) \cdot p(w_i, w'_i)$. Finally, $\mathit{expected\_best\_cost}(\mathcal{A}, \eta, p, w)$ is the lowest expected cost with which w can be read by $\mathcal{A}$ (with or without cheating). Thus, $\mathit{expected\_best\_cost}(\mathcal{A}, \eta, p, w) = \min_{w' : |w'| = |w|} \mathit{expected\_faked\_cost}(\mathcal{A}, \eta, p, w, w')$. We refer to the word w' with which the minimum is achieved as the *cheating pattern* for w.

We say that $\mathcal{A}$ is (η, p)-*resilient to cheating* if it is not worthwhile to cheat $\mathcal{A}$ given the penalty function η and the detection-probability function p. Formally, $\mathcal{A}$ is (η, p)-resilient to cheating if for every input word w, it holds that $\mathit{cost}(\mathcal{A}, w) = \mathit{expected\_best\_cost}(\mathcal{A}, \eta, p, w)$.

Studying resilience of automata, it is convenient to consider a non-probabilistic setting in which cheats are always detected. We use $\hat{1}$ denote the detection-probability function satisfying $\hat{1}(\sigma, \sigma') = 1$ for all $\sigma, \sigma' \in \Sigma$. As argued in Theorem 2.1 below, the probabilistic setting can be easily reduced to the non-probabilistic one. The theorem follows easily from the linearity of expectation.

Theorem 2.1. *Consider a WFA $\mathcal{A}$, penalty function η, and detection-probability function p. Let $\theta = \eta \circ p$. Thus, $\theta : \Sigma \times \Sigma \to \mathbb{R}^{\geq 0}$ is such that for all $\sigma, \sigma' \in \Sigma$, we have that $\theta(\sigma, \sigma') = \eta(\sigma, \sigma') \cdot p(\sigma, \sigma')$. Then, for every $w \in \Sigma^*$, we have $\mathit{expected\_best\_cost}(\mathcal{A}, \eta, p, w) = \mathit{expected\_best\_cost}(\mathcal{A}, \theta, \hat{1}, w)$*

Thus, by considering the penalty function $\theta = \eta \circ p$, we can reduce a probabilistic setting with η and p to a non-probabilistic one. The cost of a word in $\mathcal{A}$ is still an expected one, but for simplicity of notations, we use the terms $\mathit{faked\_cost}(\mathcal{A}, \theta, w, w')$ and $\mathit{best\_cost}(\mathcal{A}, \theta, w)$, which are analogue to $\mathit{expected\_faked\_cost}(\mathcal{A}, \eta, p, w, w')$ and $\mathit{expected\_best\_cost}(\mathcal{A}, \eta, p, w)$, and refer to θ-resilience to cheating, rather than (η, p)-resilience.

Example 2.2. Consider the DWFA $\mathcal{A}$ in Figure 1. Every state q_i in the figure is labeled by its final cost. For example, $\tau(q_4) = 4$, and $\tau(q_3) = x$, for some $x \in \mathbb{R}$. Every transition is labeled by the letter and cost associated with it. For example, $\Delta(q_2, b, q_5)$ and $c(q_2, b, q_5) = 1$. Assume that the penalty function is uniform and for all $\sigma, \sigma' \in \{a, b, c\}$ with $\sigma \neq \sigma'$, we have $\theta(\sigma, \sigma') = 2$.

The DWFA $\mathcal{A}$ demonstrates two of the phenomenon that makes the analysis of cheating challenging. First, testing an WFA for θ-resilience (even a DWFA, and even with a uniform θ) may not be local. In our example, if we take $x = 0$, then it is easy to see that for every three states q, q', and q'', and two letters σ and σ', it holds that $c(q, \sigma, q') + \tau(q') \leq c(q, \sigma', q'') + \tau(q'') + \theta(\sigma, \sigma')$; that is, for all words of length 1 it is not beneficial to cheat, independent of the initial state. Clearly, this is a necessary condition for

$\mathcal{A}$ to be θ-resilient: if there are q, q', q'', σ, and σ' that violate the condition, then the word $w \cdot \sigma$ for which $\delta(q_0, w) = q$, has $\mathit{faked\_cost}(\mathcal{A}, \theta, w \cdot \sigma, w \cdot \sigma') < \mathit{cost}(\mathcal{A}, w \cdot \sigma)$, thus $\mathit{best\_cost}(\mathcal{A}, \theta, w \cdot \sigma) < \mathit{cost}(\mathcal{A}, w \cdot \sigma)$ and $w \cdot \sigma$ has an incentive to cheat and pretend to be $w \cdot \sigma'$. This condition, however, is not sufficient. For example, $\mathit{cost}(\mathcal{A}', aa) = 8$ while $\mathit{faked\_cost}(\mathcal{A}, \theta, aa, bb) = 2 + 2\theta(a, b) = 6$. That is, *aa* has an incentive to cheat and pretend to be *bb*.

Second, $\mathcal{A}$ demonstrates that cheating may be beneficial only for words that are unboundedly long. To see this, note that $\mathit{cost}(\mathcal{A}, bc^k) = k + 1$ and $\mathit{cost}(\mathcal{A}, c^{k+1}) = x + 1$. Since cheating in the first letter costs 2, we have that $\mathit{best\_cost}(\mathcal{A}, \theta, bc^k) = \min(k+1, x+3)$ and $\mathit{best\_cost}(\mathcal{A}, \theta, c^{k+1}) = \min(k+3, x+1)$. Thus, the larger x is, the longer are the shortest input words that have an incentive to cheat.

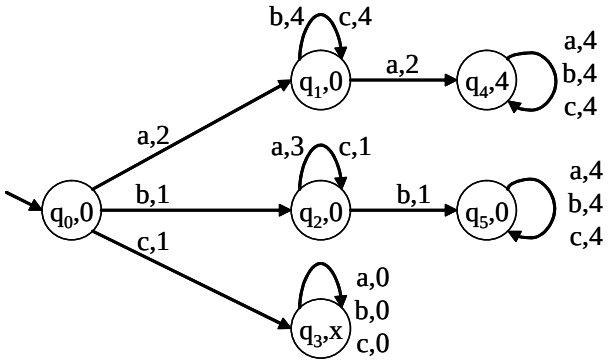

Figure 1: The DWFA $\mathcal{A}$.

A basic challenge in the setting of rational environments is to design systems in which the environment has no incentive to cheat. In our setting, one could ask whether a given WFA $\mathcal{A}$ that is not θ-resilient to cheating can be modified to an equivalent WFA $\mathcal{A}'$ that is θ-resilient to cheating. Theorem 2.3 below states that this is impossible.

Theorem 2.3. *Resilience to cheating is a semantic property. That is, given a weighted language $L : \Sigma^* \to \mathbb{R}^{\geq 0} \cup \{\infty\}$ and a penalty function θ, either all WFAs for L are θ-resilient to cheating, or none of them is θ-resilient to cheating.*

Proof: Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two WFAs for L. Thus, for every $w \in \Sigma^*$, we have that $\mathit{cost}(\mathcal{A}_1, w) = \mathit{cost}(\mathcal{A}_2, w) = L(w)$. We show that if $\mathcal{A}_1$ is not θ-resilient to cheating, then so is $\mathcal{A}_2$. Assume that $\mathcal{A}_1$ is not θ-resilient to cheating, and let w and w' be such that $|w'| = |w|$ and $\mathit{faked\_cost}(\mathcal{A}_1, \theta, w, w') < \mathit{cost}(\mathcal{A}_1, w)$. Recall that $\mathit{faked\_cost}(\mathcal{A}_1, \theta, w, w') = \mathit{cost}(\mathcal{A}_1, w') + \theta(w, w')$. By the equivalence of $\mathcal{A}_1$ and $\mathcal{A}_2$, we have that $\mathit{cost}(\mathcal{A}_1, w) = \mathit{cost}(\mathcal{A}_2, w)$ and $\mathit{cost}(\mathcal{A}_1, w') = \mathit{cost}(\mathcal{A}_2, w')$. Hence, since $\theta(w, w')$ is independent of the WFA, we also have $\mathit{faked\_cost}(\mathcal{A}_2, \theta, w, w') < \mathit{cost}(\mathcal{A}_2, w)$, and we are done. □

Note that Theorem 2.3 applies for both nondeterministic and deterministic WFAs. Thus, nondeterminism cannot help a WFA to cope with cheats. Note also that Theorem 2.3 considers a given penalty function θ and does not include the possibility of achieving resilience by modifying the penalty function, possibly using the same budget. We will get back to this problem in Section 4.

2.3 The Cheating-Allowed Automaton

Reasoning about a WFA $\mathcal{A}$ and its resilience to cheating, one has to take into account the infinitely many possible cheating patterns that $\mathcal{A}$ should be resilient too. In this section we show that these patterns can be modeled by a single WFA obtained from $\mathcal{A}$ by adding transitions that mimics cheating.

Theorem 2.4. *Consider a WFA $\mathcal{A}$ and a penalty function $\theta : \Sigma \times \Sigma \to \mathbb{R}^{\geq 0}$. There is a WFA $\mathcal{A}'$, with the same state space as $\mathcal{A}$, such that $cost(\mathcal{A}', w) = best\_cost(\mathcal{A}, \theta, w)$.*

Proof: Let $\mathcal{A} = \langle \Sigma, Q, \Delta, c, q_0, \tau \rangle$. We define $\mathcal{A}' = \langle \Sigma, Q, \Delta', c', q_0, \tau \rangle$, where the transition relation Δ' and the cost function c' are defined as follows. For every two states $q, q' \in Q$, if there is $\sigma' \in \Sigma$ such that $\Delta(q, \sigma', q')$, then $\Delta'(q, \sigma, q')$ for every $\sigma \in \Sigma$, and $c'(q, \sigma, q') = \min_{\sigma' : \Delta(q, \sigma', q')} \{c(q, \sigma', q') + \theta(\sigma, \sigma')\}$. That is, if the set Σ' of letters with which $\mathcal{A}$ can move from q to q' is not empty, then $\mathcal{A}'$ can move from q to q' with all letters – by reporting them to be some letter in Σ'. The cost of this transition for a letter σ is calculated by taking the most beneficial replacement from Σ': the one that minimizes the sum of the cost of the transition and the cost of cheating.

It is not hard to see the correspondence between the nondeterminism of $\mathcal{A}'$ and the choices of cheating patterns. Formally, for every word w, a cheating pattern w' for w induces a run of $\mathcal{A}'$ on w whose cost is $faked\_cost(\mathcal{A}, \theta, w, w')$. Likewise, every run of $\mathcal{A}'$ on w induces a word w' that can serve as a cheating pattern for w. Hence, since the cost of w in $\mathcal{A}'$ is the minimal cost of some run of $\mathcal{A}'$ on w, we have that $best\_cost(\mathcal{A}, \theta, w) = cost(\mathcal{A}', w)$, and we are done. □

Given a WFA $\mathcal{A}$ and a penalty function θ, we refer to the WFA $\mathcal{A}'$ constructed in Theorem 2.4 as $Cheat(\mathcal{A}, \theta)$. For example, the WFA in Figure 2 is $Cheat(\mathcal{A}, \theta)$, for the WFA $\mathcal{A}$ described in Figure 1 and $\theta(\sigma, \sigma') = 2$ for all $\sigma, \sigma' \in \Sigma$ with $\sigma \neq \sigma'$.

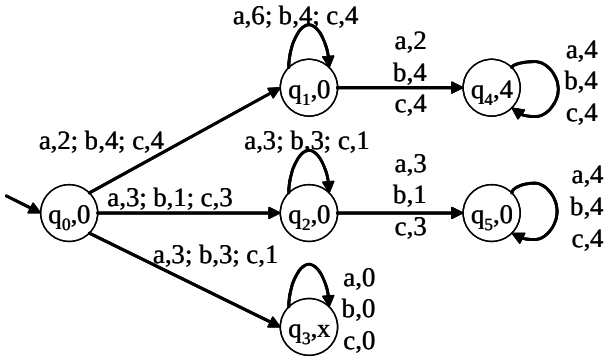

Figure 2: The WFA $\mathcal{A}' = Cheat(\mathcal{A}, \theta)$, with uniform $\theta = 2$.

Corollary 2.5. *For every WFA $\mathcal{A}$ and penalty function θ, we have that $\mathcal{A}$ is θ-resilient to cheating iff $\mathcal{A} \preceq Cheat(\mathcal{A}, \theta)$, that is, for every word $w \in \Sigma^*$, we have that $cost(\mathcal{A}, w) \leq cost(Cheat(\mathcal{A}, \theta), w)$.*

Theorem 2.6. *Given a WFA $\mathcal{A}$, a penalty function θ, and a word $w \in \Sigma^*$, the problem of finding $best\_cost(\mathcal{A}, \theta, w)$ and a cheating pattern for it, can be solved in polynomial time.*

Proof: Given a WFA $\mathcal{A}$ and a word $w \in \Sigma^*$, it is possible to find $cost(\mathcal{A}, w)$ as follows (note that we refer here to cost without cheating). If $\mathcal{A}$ is deterministic, we traverse the single run of $\mathcal{A}$ on w and find its cost. If $\mathcal{A}$ is nondeterministic, we first restrict $\mathcal{A}$ to runs along which w is read, and then find the cheapest such run. Formally, we define the product $\mathcal{A}_w$ of $\mathcal{A}$ with an un-weighted automaton with $|w|+1$ states whose language is $\{w\}$. The WFA $\mathcal{A}_w$ describes exactly all the run of $\mathcal{A}$ on w and it has no cycles. We apply to $\mathcal{A}_w$ a shortest-path algorithm [Dij59] and find the shortest path from an initial state to a final state.

Now, given $\mathcal{A}$ and θ, let $\mathcal{A}'$ be $Cheat(\mathcal{A}, \theta)$. Then, for every word w, we have that $best\_cost(\mathcal{A}, \theta, w) = cost(\mathcal{A}', w)$, which can be calculated as described above. Also, the run r' of $\mathcal{A}'$ on w for which $cost(\mathcal{A}', w) = cost(r', w)$ reveals the cheating pattern. □

Limited Cheating and Rising Penalty Variants: In the above described setting, an input word can cheat as many times as it wants. Also, the penalties are fixed throughout the interaction. It is easy to modify the construction of $Cheat(\mathcal{A}, \theta)$ and, consequently, our results below, to account for variant models. For example, by taking several copies of $Cheat(\mathcal{A}, \theta)$, it is possible to give a constant bound on the number of allowed cheats (the states maintain the number of cheats detected so far) or constant bound on the budget a word can use for cheating (the states maintain the total cheating costs detected so far). By taking several copies of $Cheat(\mathcal{A}, \theta)$ and modifying the costs in the different copies, it is possible to let $\mathcal{A}$ increase the penalties when cheats are detected (this corresponds to increasing either the detection-probability function or the penalties themselves; as indeed happens in practice when cheats are detected).

3 Resilience Testing

In this section we study the problem of deciding, given a WFA $\mathcal{A}$ and a penalty function θ, whether $\mathcal{A}$ is θ-resilient to cheating. Recall that $\mathcal{A}$ is θ-resilient to cheating if $cost(\mathcal{A}, w) = best\_cost(\mathcal{A}, \theta, w)$. We show that the problem is PSPACE-hard for WFA but can be solved in polynomial time for DWFA.

3.1 Hardness Proof for WFA

Theorem 3.1. *Consider a WFA $\mathcal{A}$ and a penalty function θ. The problem of deciding whether $\mathcal{A}$ is θ-resilient is PSPACE-hard.*

Proof: We do a reduction from the universality problem for NFAs, proven to be PSPACE-hard in [RS59]. Given an NFA $\mathcal{U}$, we construct a WFA $\mathcal{A}_\mathcal{U}$ such that $\mathcal{A}_\mathcal{U}$ is 0-resilient (that is, $\theta(\sigma, \sigma') = 0$ for all $\sigma, \sigma' \in \Sigma$) iff $\mathcal{U}$ is universal. Note that an automaton is 0-resilient iff no input word has an incentive to cheat even if cheating is free. The idea behind the construction is that words not in $L(\mathcal{U})$ would induce words that have an incentive to cheat $\mathcal{A}_\mathcal{U}$. Thus, $\mathcal{U}$ is universal iff no word has an incentive to cheat $\mathcal{A}_\mathcal{U}$, so even the 0 penalties suffice to ensure resilience. Formally, let $\mathcal{U} = \langle \Sigma, Q, \Delta, Q_0, F \rangle$, where $F \subseteq Q$ is a set of final states, and let a be some letter in Σ. We assume that $|\Sigma| > 1$. We define $\mathcal{A}_\mathcal{U}$ to go with the letter a to a copy of $\mathcal{U}$ and to go with all letters $\Sigma \setminus \{a\}$ to an accepting sink (see Figure 3). Thus, $\mathcal{A}_\mathcal{U} = \langle \Sigma, Q \cup \{q_0, q_{qcc}\}, \Delta', \{q_0\}, c, \tau \rangle$, where $\Delta' = \Delta \cup (\{q_0\} \times \{a\} \times Q_0) \cup (\{q_0\} \times (\Sigma \setminus \{a\}) \times \{q_{acc}\}) \cup (\{q_{acc}\} \times \Sigma \times \{q_{acc}\})$.

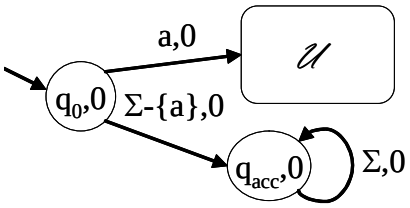

Figure 3: The WFA $\mathscr{A}_{\mathscr{U}}$.

Also, for all $\langle q,\sigma,q'\rangle \in \Delta'$, we have $c(\langle q,\sigma,q'\rangle) = 0$ and for all $q \in Q \cup \{q_0, q_{qcc}\}$ we have $\tau(q) = 0$. It is easy to see that $\mathscr{A}_{\mathscr{U}}$ accepts (with cost 0) all words of the form $a \cdot w$, for $w \in L(\mathscr{U})$, or of the form $\sigma \cdot w$, for $\sigma \neq a$ and $w \in \Sigma^*$. Accordingly, if $\mathscr{U}$ is universal, then $\mathscr{A}_{\mathscr{U}}$ accepts all words in Σ^* with cost 0, and is therefore 0-resilient. Also, if $\mathscr{U}$ is not universal, then there is $w \notin L(\mathscr{U})$ such that $cost(\mathscr{A}_{\mathscr{U}}, a \cdot w) = \infty$, while $faked\_cost(\mathscr{A}_{\mathscr{U}}, a \cdot w, b \cdot w) = \theta(a,b)$, for any $b \in \Sigma \setminus \{a\}$. Hence, $\mathscr{A}_{\mathscr{U}}$ is not 0-resilient, and we are done. □

Many fundamental problems about WFAs are still open. Unlike standard (non-weighted) automata, not all weighted automata can be determinized [Moh97]. In fact, even the problem of deciding whether a given WFA has an equivalent DWFA is open, and so are problems that use determinization in their solution, like deciding whether $\mathscr{A} \preceq \mathscr{A}'$ for two WFAs $\mathscr{A}$ and $\mathscr{A}'$ [Kro94, CDH08]. We note that the problem of deciding whether $\mathscr{A} \preceq \mathscr{A}'$ is open even when $\mathscr{A}$ is a DWFA – it is the nondeterminism in $\mathscr{A}'$ that makes the problem challenging. Thus, even for the case $\mathscr{A}$ is deterministic, we cannot reduce the problem of deciding whether $\mathscr{A} \preceq Cheat(\mathscr{A},\theta)$ to a problem whose solution is known. As we describe below, we are still able to present a polynomial solution to the problem.

3.2 A Polynomial Algorithm for DWFA

We turn to consider the case where $\mathscr{A}$ is deterministic. We show that in this case, the problem of deciding whether $\mathscr{A}$ is θ-resilient, for a given penalty function θ, can be solved in polynomial time. Let $\mathscr{A} = \langle \Sigma, Q, \Delta, c, q_0, \tau\rangle$ be a DWFA. Let $n = |Q|$. For a given penalty function θ, let $\mathscr{A}' = \langle \Sigma, Q, \Delta', c', q_0, \tau\rangle$ be $Cheat(\mathscr{A},\theta)$. We describe an algorithm for deciding whether $\mathscr{A} \preceq \mathscr{A}'$. By Corollary 2.5, the latter holds iff $\mathscr{A}$ is θ-resilient to cheating.

Our algorithm is similar to the algorithm for deciding whether a given DWFA is equivalent to a WFA in which it is embodied [AKL09]. We define a sequence of functions $h_0, h_1, \ldots : Q \times Q \to \mathbb{R} \cup \{\infty, -\infty\}$, as follows. [2] Intuitively, $h_i(q,q')$ indicates how much a word of length at most i can gain if instead of a run of $\mathscr{A}$ that leads to q it takes a run of $\mathscr{A}'$ that leads to q'. This difference does not include the final costs of q, and q'. Note that there may not be words of length at most i along which q and q' are reachable, in which case $h_i(q,q')$ would be $-\infty$. Also, it may be that for all words w of length at most i, the cheapest run in $\mathscr{A}'$ that reads w and leads to q' costs more than the run of $\mathscr{A}$ that reads w and leads to q, in which case $h_i(q,q')$ is negative.

[2] In the definition of h_i we use addition and subtraction on the elements of $\mathbb{R} \cup \{\infty, -\infty\}$. For every finite $x \in \mathbb{R}$, we have $\infty - x = \infty$, and $x - \infty = -\infty$. Also $\infty - \infty = 0$.

It is easy to see that if for some $i \in \mathbb{N}$ and $q, q' \in Q$, we have that $h_i(q,q') > \tau(q') - \tau(q)$, then there is a word of length at most i for which $cost(\mathscr{A}, w) > cost(\mathscr{A}', w)$, thus $\mathscr{A} \not\preceq \mathscr{A}'$. We show that h_i can be calculated efficiently, and that even though the sequence of functions may not reach a fixed-point, it is possible to determine whether $\mathscr{A} \preceq \mathscr{A}'$ after calculating h_i for $i = 0, \ldots, O(n^2)$. Intuitively, it follows from the fact that not reaching a fixed-point after $O(n^2)$ iterations points to cycles along which the gain of $\mathscr{A}'$ with respect to $\mathscr{A}$ is unbounded.

We initialize $h_0(q_0, q_0) = 0$ and $h_0(q,q') = -\infty$ for all other pairs. Indeed, (q_0, q_0) is the only pair of states to which an empty word might reach on $\mathscr{A}$ and $\mathscr{A}'$.

The calculation of h_{i+1}, for $i \geq 0$, uses a function $g_{i+1} : Q \times Q \times \Sigma \to \mathbb{R} \cup \{\infty, -\infty\}$. Intuitively, $g_{i+1}(q,q',\sigma)$ indicates how much a word of length at most $i+1$ that ends with the letter σ can gain if instead of a run of $\mathscr{A}$ that leads to q it takes a run of $\mathscr{A}'$ that leads to q'. Then,

$$g_{i+1}(q,q',\sigma) = \max_{p,p' : \Delta(p,\sigma,q) \wedge \Delta'(p',\sigma,q')} (h_i(p,p') + c(p,\sigma,q) - c'(p',\sigma,q')). \tag{1}$$

Thus, the calculation of $g_{i+1}(q,q',\sigma)$ considers all pairs $\langle p, p' \rangle \in Q$ from which q and q' can be reached, respectively, when a is read. Since $g_{i+1}(q,q',\sigma)$ is the gain obtained by running in $\mathscr{A}'$ instead of in $\mathscr{A}$, we add to $h_i(p,p')$ the cost of the transition $\langle p, \sigma, q \rangle$ in $\mathscr{A}$ and subtract the cost of the transition $\langle p', \sigma, q' \rangle$ in $\mathscr{A}'$. Now, for $i \geq 0$, we have

$$h_{i+1}(q,q') = \max\{h_i(q,q'), \max_{\sigma \in \Sigma} g_{i+1}(q,q',\sigma)\}. \tag{2}$$

For $i \geq 0$ and $q, q' \in Q$, we say that a word w *witnesses* $h_i(q,q')$ if $|w| \leq i$ and there is a run of $\mathscr{A}'$ on w that leads to q' and traversing its transitions costs $h_i(q,q')$ less than traversing the transitions of the run of $\mathscr{A}$ on w, which leads to q. Note that since the functions h_i ignore the final costs, the above refers to the cost of traversing the transitions along the runs, rather than the cost of the runs. Clearly, if $h_i(q,q')$ is finite, then it has at least one witness.

We can now present the algorithm for deciding whether $\mathscr{A} \preceq A'$:

1. For $i = 0, \ldots, n^2$: Calculate h_i; if for some $q, q' \in Q$, we have $h_i(q,q') > \tau(q') - \tau(q)$, then return $(\mathscr{A} \not\preceq \mathscr{A}')$.
2. For $i = n^2 + 1, \ldots, 2n^2$: Calculate h_i; if for some $q, q' \in Q$, we have $h_{i-1}(q,q') < h_i(q,q')$, then return $(\mathscr{A} \not\preceq \mathscr{A}')$.
3. Return$(\mathscr{A} \preceq \mathscr{A}')$.

We prove the correctness of the algorithm.

Assume first that the algorithm returns that $\mathscr{A} \not\preceq \mathscr{A}'$. We distinguish between two cases. If the algorithm declares that $\mathscr{A} \not\preceq \mathscr{A}'$ in Step 1, then the word w that witnesses $h_i(q,q')$ satisfies $cost(\mathscr{A}', w) < cost(\mathscr{A}, w)$. If the algorithm declares that $\mathscr{A} \not\preceq \mathscr{A}'$ in Step 2, let $n^2 < i \leq 2n^2$ and $q, q' \in Q_{i+1}$ be such that $h_i(q,q') < h_{i+1}(q,q')$, and let w be the word of length $i+1$ that witnesses $h_{i+1}(q,q')$. Let $r = q_0, \ldots, q_{i+1}$ be the single run of $\mathscr{A}$ on w, and let $r' = q'_0, \ldots, q'_{i+1}$ be a run of $\mathscr{A}'$ on w such that $cost(\mathscr{A}', w) = cost(\mathscr{A}', r')$. Thus, r' is the run of $\mathscr{A}'$ along which $cost(\mathscr{A}', w)$ is obtained. Note that $q = q_{i+1}$ and $q' = q'_{i+1}$. Since $i+1 > n^2$, there must be two indices $0 \leq j_1 < j_2 \leq i+1$ such that $q_{j_1} = q_{j_2}$ and $q'_{j_1} = q'_{j_2}$. Let $\gamma = cost(\mathscr{A}, r, j_1+1, j_2)$ and $\gamma' = cost(\mathscr{A}', r', j_1+1, j_2)$.

Consider the word $w' = w_1 \cdots w_{j_1} \cdot w_{j_2+1} \cdots w_{i+1}$. Thus, w' is obtained from w by removing the sub-word $w_{j_1+1} \cdots w_{j_2}$ along which $\mathscr{A}$ and $\mathscr{A}'$ cycle. The single run of $\mathscr{A}$ on w' is $v = q_0, \ldots, q_{j_1}, q_{j_2+1}, \ldots, q_{i+1}$. Also, $v' = q'_0, \ldots, q'_{j_1}, q'_{j_2+1}, \ldots, q'_{i+1}$ is a legal run of $\mathscr{A}'$ on w'. Note that $cost(\mathscr{A}, v) = cost(\mathscr{A}, r) - \gamma$ and $cost(\mathscr{A}', v') = cost(\mathscr{A}', r') - \gamma'$. Since $h_i(q, q') < h_{i+1}(q, q')$, both r and v end in q_{j_1}, both r' and v' end in q'_{j_1}, and w' is of length at most i (and may therefore serve as a witness to $h_i(q, q')$), it must be that $cost(\mathscr{A}, v) - cost(\mathscr{A}', v') < cost(\mathscr{A}, r) - cost(\mathscr{A}', r')$. Hence, $\gamma - \gamma' > 0$.

For $j \geq 1$, let $w_j = w_1 \cdots w_{j_1} \cdot (w_{j_1+1} \cdots w_{j_2})^j$. Thus, w_j is obtained from w by pumping the sub-word $w_{j_1+1} \cdots w_{j_2}$ for j times. Let $\alpha = cost(\mathscr{A}, r, 1, j_1)$ and let α' be the cost of the cheapest run of $\mathscr{A}'$ that reads $w_1 \cdots w_{j_1}$ and leads from q_0 to q'_{j_1}. Recall that $\gamma - \gamma' > 0$, thus $\gamma > \gamma'$. Hence, since α, α', $\tau(q_{j_1})$, and $\tau(q'_{j_1})$ are all finite, there must be $j \geq 0$ for which $\alpha + j \cdot \gamma + \tau(q_{j_1}) > \alpha' + j \cdot \gamma' + \tau(q'_{j_1})$. Since $cost(\mathscr{A}, w_j) = \alpha + j \cdot \gamma + \tau(q_{j_1})$ and $cost(\mathscr{A}', w_j) \leq \alpha + j \cdot \gamma + \tau(q_{j_1})$, it follows that there is $j \geq 0$ for which $cost(\mathscr{A}, w_j) > cost(\mathscr{A}', w_j)$, thus $\mathscr{A} \not\preceq \mathscr{A}'$, and we are done.

Assume now that $\mathscr{A} \not\preceq \mathscr{A}'$. Let $w = w_1 \cdots w_l$ be the shortest word for which $cost(\mathscr{A}, w) > cost(\mathscr{A}', w)$. Let $r = q_0, \ldots, q_l$ be the single run of $\mathscr{A}$ on w, and let $r' = q'_0, \ldots, q'_l$ be a run of $\mathscr{A}'$ on w such that $cost(\mathscr{A}', w) = cost(\mathscr{A}', r')$. Thus, r' is the run along which $cost(\mathscr{A}', w)$ is achieved.

We distinguish between two cases. First, if $l \leq n^2$, then, by the definition of the functions h_i, we have $h_l(q_l, q'_l) > \tau(q'_l) - \tau(q_l)$, thus the algorithm detects that $\mathscr{A} \not\preceq \mathscr{A}'$ in Step 1.

Second, if $l > n^2$, then there must be two indices $0 \leq j_1 < j_2 \leq n^2$ such that $q_{j_1} = q_{j_2}$ and $q'_{j_1} = q'_{j_2}$. Let $\gamma = cost(\mathscr{A}, r, j_1 + 1, j_2)$ and $\gamma' = cost(\mathscr{A}', r', j_1 + 1, j_2)$. Since w is the shortest word for which $cost(\mathscr{A}, w) > cost(\mathscr{A}', w)$, it must be that $\gamma > \gamma'$. Indeed, otherwise, the word $w' = w_1 \cdots w_{j_1} \cdot w_{j_2+1} \cdots w_l$, which is shorter than w, also satisfies $cost(\mathscr{A}, w') > cost(\mathscr{A}', w')$.

Let $y \in \Sigma^*$ be a word of length at most n^2 that witnesses $h_{n^2}(q_{j_1}, q'_{j_1})$. Thus, $|y| = t$, for $t \leq n^2$, and there are runs $s = s_0, \ldots, s_t$ and $s' = s'_0, \ldots, s'_t$ of $\mathscr{A}$ and $\mathscr{A}'$, respectively, on y, such that $s_t = q_{j_1}$, $s'_t = q'_{j_1}$, and $cost(\mathscr{A}, s, 1, t) - cost(\mathscr{A}', s', 1, t) = h_{n^2}(q_{j_1}, q'_{j_1})$.

Let $j = j_2 - j_1$. Consider the word $w' = y \cdot w_{j_1+1} \cdots w_{j_2}$. The word w' is of length $t + j$. The single run of $\mathscr{A}$ on w' is $v = s_0, s_1, \ldots, s_t, q_{j_1+1}, \ldots, q_{j_2}$. Also, $v' = s'_0, s'_1, \ldots, s'_t, q'_{j_1+1}, \ldots, q'_{j_2}$ is a legal run of $\mathscr{A}'$ on w'. Note that $cost(\mathscr{A}, v, 1, t + j) = cost(\mathscr{A}, s, 1, t) + \gamma$ and $cost(\mathscr{A}', v', 1, t + j) = cost(\mathscr{A}', s', 1, t) + \gamma'$. Also, since w' may serve as a witness to $h_{t+j}(q_{j_1}, q'_{j_2})$, it must be that $h_{t+j}(q_{j_1}, q'_{j_2}) \geq cost(\mathscr{A}, v, 1, t + j) - cost(\mathscr{A}', v', 1, t + j)$. Since y witnesses $h_t(q_{j_1}, q'_{j_1})$ and $\gamma - \gamma' > 0$, it follows that $h_{t+j}(q_{j_1}, q'_{j_1}) > h_t(q_{j_1}, q'_{j_1})$. Since $h_t(q_{j_1}, q'_{j_1}) = h_{n^2}(q_{j_1}, q'_{j_1})$, we conclude that $h_{t+j}(q_{j_1}, q'_{j_1}) > h_{n^2}(q_{j_1}, q'_{j_1})$.

We claim that $n^2 < t + j \leq 2n^2$. Since $t, j \leq n^2$, then clearly $t + j \leq 2n^2$. To see that $n^2 < t + j$, assume by way of contradiction that $t + j \leq n^2$. Then, the word w' is of length at most n^2, and it can serve as witness to $h_{n^2}(q_{j_1}, q'_{j_1})$. Since $h_{t+j}(q_{j_1}, q'_{j_1}) > h_{n^2}(q_{j_1}, q'_{j_1})$, this contradicts the fact that y witnesses $h_{n^2}(q_{j_1}, q'_{j_1})$. Now, since $h_{t+j}(q_{j_1}, q'_{j_1}) > h_{n^2}(q_{j_1}, q'_{j_1})$, we conclude that there is an iteration $n^2 \leq i \leq 2n^2$ such that $h_i(q_{j_1}, q'_{j_1}) < h_{i+1}(q_{j_1}, q'_{j_1})$, and the algorithm declares that $\mathscr{A} \not\preceq \mathscr{A}'$ is Step 2.

The function h_0 can be calculated in polynomial time, and so is the function h_{i+1}, given h_i. Hence, since we need only a polynomial number of iterations, we can conclude with the following.

Theorem 3.2. *Consider a DWFA $\mathscr{A}$ and a penalty function θ. The problem of deciding whether $\mathscr{A}$ is θ-resilient can be solved in polynomial time.*

4 Achieving Resilience with Minimum Resources

A system with no limit on penalties and with unbounded resources can prevent cheating by fixing a high penalty function. In practice, penalties may be limited by an external authority, and increasing the probability of detecting cheats requires resources. In this section we study the problem of minimizing the resources required in order to guarantee resilience.

We assume that the penalty function η is determined by an external authority and that $\mathcal{A}$ is $(\eta, \hat{1})$-resilient. Thus, the environment has no incentive to cheat if cheating is always detected.[3] Given a WFA $\mathcal{A}$, and a penalty function η, our goal is to find a detection-probability function p, such that $\mathcal{A}$ is (η, p)-resilient to cheating and the budget $B = \sum_{\sigma,\sigma' \in \Sigma} \eta(\sigma,\sigma') \cdot p(\sigma,\sigma')$ is minimal. The rationale behind our goal is that the system can control the probability of catching cheats. In practice, detection probability can be increased by investing in "guards", each responsible for a specific possible cheat. The budget we have is the total payment for the guards. The payment to the guard responsible for detecting σ being reported as σ' is independent of the actual number of times σ is being reported as σ'. On the other hand, the payment is proportional to the penalty $\eta(\sigma,\sigma')$ charged whenever the guard detects the cheat. Indeed, in practice, detecting a cheat with a high penalty tends to require high resources: knowing that his success leads to a high revenue, a guard would require high salary. We say that $\mathcal{A}$ *can achieve resilience with budget B* if there are η and p such that the budget of η and p is B, and $\mathcal{A}$ is (η, p)-resilient to cheating.

As explained in Section 2.2, we can consider an equivalent non-probabilistic setting in which all cheats are always detected and are charged according to the penalty function $\theta = \eta \circ p$. In the rest of this section we therefore consider the problem of deciding, given a WFA $\mathcal{A}$ and a budget $B \in \mathbb{R}^{\geq 0}$, whether $\mathcal{A}$ can achieve resilience with budget B, as well as the optimization problem of finding the minimal budget with which $\mathcal{A}$ can achieve resilience. A solution for the above problems induces the expected-fee function θ. Having θ in hand, we use the given penalty function η to fix $p(\sigma,\sigma') = \frac{\theta(\sigma,\sigma')}{\eta(\sigma,\sigma')}$. In order to guaranteed that our solution is feasible, that is, the probability function is over the rage $[0,1]$, our algorithm only considers solutions in which for all $\sigma,\sigma' \in \Sigma$ we have $\eta(\sigma,\sigma') \geq \theta(\sigma,\sigma')$.

4.1 Hardness Proof for WFA

We first show that, as in the resilience testing problem, the nondeterministic setting is much more difficult.

Theorem 4.1. *Consider a WFA $\mathcal{A}$. Given a budget B, the problem of deciding whether there is a penalty function θ with budget B such that $\mathcal{A}$ is θ-resilient to cheating is PSPACE-hard.*

Proof: As in the proof of Theorem 3.1, we do a reduction from the universality problem for NFAs. Given an NFA $\mathcal{U}$, we construct a WFA $\mathcal{A}_\mathcal{U}$ such that there is a penalty function θ with budget 0 with which $\mathcal{A}_\mathcal{U}$ is θ-resilient to cheating iff $\mathcal{U}$ is universal.

The construction is similar to the one described in the proof of Theorem 3.1, except that now the transition from q_0 to q_{acc} is labeled by both all the letters in $\Sigma \setminus \{a\}$, with cost 0, and the letter a, with cost 1. It is easy to see that the cost in $\mathcal{A}_\mathcal{U}$ of words of the form $a \cdot w$ is 0 for $w \in L(\mathcal{A})$ and is 1 for

[3] Note that this is a reasonable assumption as otherwise, the authority providing the penalty function encourages cheating.

$w \notin L(\mathcal{A})$. Also, for $\sigma \neq a$, the cost of words of the form $\sigma \cdot w$ is 0, regardless of the membership of w in $L(\mathcal{A})$. Accordingly, if $\mathcal{U}$ is universal, then $\mathcal{A}_{\mathcal{U}}$ accepts all words in Σ^* with cost 0, and is therefore 0-resilient, in which a budget 0 suffices to ensure resilience. Also, if $\mathcal{U}$ is not universal, then there is $w \notin L(\mathcal{U})$ such that $cost(\mathcal{A}_{\mathcal{U}}, a \cdot w) = 1$, while $faked\_cost(\mathcal{A}_{\mathcal{U}}, a \cdot w, b \cdot w) = \theta(a,b)$, for any $b \in \Sigma \setminus \{a\}$. Hence, in order to ensure θ-resilience, a penalty function θ must satisfy $\theta(a,b) \geq 1$, thus the budget required to θ is at least $|\Sigma| - 1$, and we are done. □

4.2 A Polynomial Algorithm for DWFA

We turn to consider deterministic WFAs. Note that if we define an order $\leq$ between penalty functions, where $\theta_1 \leq \theta_2$ iff $\theta_1(\sigma,\sigma') \leq \theta_2(\sigma,\sigma')$ for all $\sigma,\sigma' \in \Sigma$, then the penalty functions that ensure resilience are not linearly ordered. This last observation hints that the problem of finding a minimal sufficient penalty with respect to which $\mathcal{A}$ is resilient cannot be solved in a straightforward way, as it cannot be based on a search in a linearly ordered domain. Still, as we show below, when $\mathcal{A}$ is a deterministic DFA, it is possible to describe the resilience requirements as a set of linear inequality constraints. Since the optimization objective can be also described as a linear function, it is possible to determine the minimal sufficient penalty function using linear programming (LP). LP is a mathematical tool suitable for determining an optimal solution for a linear objective function defined over a set of variables, while obeying a set of requirements represented as linear equations [Chv83].

We describe the problem as a linear programming optimization problem with a polynomial number of variables and constraints. Given a WFA $\mathcal{A}$ and a penalty function η, the algorithm returns a new penalty function θ such that:

1. $\sum_{\sigma,\sigma' \in \Sigma} \theta(\sigma,\sigma')$ is minimal.
2. For all $\sigma,\sigma' \in \Sigma$, we have $0 \leq \frac{\theta(\sigma,\sigma')}{\eta(\sigma,\sigma')} \leq 1$.
3. $\mathcal{A}$ is θ-resilient.

Note that the second property assures that $\theta = \eta \circ p$, for some probability function p satisfying $p(\sigma,\sigma') \in [0,1]$.

The first property defines the objective function of the LP. The LP constraints assure the second and third properties. Specifically, for the third property, the LP constraints assure that the algorithm described in Section 3.2, for testing whether $\mathcal{A}$ is θ-resilient, would return $\mathcal{A} \preceq Cheat(\mathcal{A},\theta)$. Accordingly, the variables we use are the following:

- For all $\sigma,\sigma' \in \Sigma$, the variable $\theta_{\sigma,\sigma'}$ maintains the penalty function $\theta(\sigma,\sigma')$.
- For $0 \leq i \leq 2n^2$ and $q,q' \in Q$, the variable $h_{i,q,q'}$ maintains $h_i(q,q')$.
- For $0 \leq i \leq 2n^2$, $q,q' \in Q$, and $\sigma \in \Sigma$, the variable $g_{i,q,q',\sigma}$ maintains $g_i(q,q',\sigma)$.

The objective function is $\min \sum_{\sigma,\sigma'} \theta_{\sigma,\sigma'}$. Since the penalty function is non-negative, we have $|\Sigma|^2$ constraints $\theta_{\sigma,\sigma'} \geq 0$ for all $\sigma,\sigma' \in \Sigma$. In addition, $\theta_{\sigma,\sigma} = 0$ for all $\sigma \in \Sigma$. Also, in order to guarantee that the detection-probability function is feasible, we have, for all $\sigma,\sigma' \in \Sigma$, the constraint $\theta_{\sigma,\sigma'} \leq \eta_{\sigma,\sigma'}$.

The additional constraints follow the structure of the algorithm presented in Section 3.2. For $k = 1,\ldots,n^2$, the k-th set of constraints assures that no word of length at most k should benefit from cheating. For $k = n^2+1,\ldots,2n^2$, the k-th set of constraints assures that no cycle that can lead to unlimited gain exists. Each such set consists of a polynomial number of constraints and introduces a polynomial number of variables. Specifically, variables of type $h_{i,q,q'}$ bound the gain of words of length at most i, and variables of type $g_{i,q,q',\sigma}$ bound this gain for words of length at most i ending with σ. While the variables $h_{i,q,q'}, g_{i,q,q',\sigma}$ are defined for every $0 \leq i \leq 2n^2$, $q,q' \in Q$, and $\sigma \in \Sigma$, in practice, many of these variables are not constrained, as it might be that no word of length at most i can reach state q in $\mathscr{A}$ and q' in $\mathscr{A}'$.

We first describe the constraints considering words of length 1, and then the constraints for general k. Note that the first set of constraints can be viewed as a special case of the general set, however, since we know that q_0 is the only possible state preceding states reachable by a single letter, the presentation of this set is simpler. We also note that in order to clarify the intuition behind each constraint, the constraints are not necessarily presented in the canonical form of an LP (that is, with all variables in the left hand side and all constants in the right hand side).

In order to assure that words of length 1 will not cheat, we have a variable $h_{1,q,q'}$ for all $q,q' \in Q$, and a variable $g_{1,q,q',\sigma}$ for all $q,q' \in Q, \sigma \in \Sigma$. To reflect Equation (1) in the algorithm described in Section 3.2, we have, for all $\sigma' \in \Sigma$ such that $\Delta(q_0,\sigma,q)$ and $\Delta(q_0,\sigma',q')$, the constraint $g_{1,q,q',\sigma} \geq c(q_0,\sigma,q) - c(q_0,\sigma',q') - \theta(\sigma,\sigma')$. To reflect Equation (2), we have, for all $q,q' \in Q$ and $\sigma \in \Sigma$ for which $g_{1,q,q',\sigma}$ is bounded, the constraint $h_{1,q,q'} \geq g_{1,q,q',\sigma}$. Since $h_0(q_0,q_0) = 0$ and the sequence of functions $h_0, h_1, \ldots$ is non-decreasing, we also have, for the state q_0, the constraint $h_1(q_0,q_0) \geq 0$. Finally, to reflect the comparison done in Step 1 of the resilience-testing algorithm, for all $q,q' \in Q$ we have the constraint $h_{1,q,q'} \leq \tau(q') - \tau(q)$.

For example, the first set of constraints defined for the DWFA $\mathscr{A}$ in Figure 4 is as follows.

$$
\begin{array}{ll}
g_{1,q_1,q_2,a} \geq 2-3-\theta_{a,b} & h_1(q_1,q_2) \geq g_{1,q_1,q_2,a} \\
g_{1,q_1,q_2,a} \geq 2-8-\theta_{a,c} & h_1(q_1,q_2) \leq 5-10 \\
g_{1,q_2,q_1,b} \geq 3-2-\theta_{b,a} & h_1(q_2,q_1) \geq g_{1,q_2,q_1,b} \\
g_{1,q_2,q_1,c} \geq 8-2-\theta_{c,a} & h_1(q_2,q_1) \geq g_{1,q_2,q_1,c} \\
g_{1,q_2,q_2,b} \geq 3-8-\theta_{b,c} & h_1(q_2,q_1) \leq 10-5 \\
g_{1,q_2,q_2,c} \geq 8-3-\theta_{c,b} & h_1(q_2,q_2) \geq g_{1,q_2,q_2,c} \\
h_1(q_0,q_0) \geq 0 & h_1(q_2,q_2) \geq g_{1,q_2,q_2,b} \\
 & h_1(q_2,q_2) \leq 5-5
\end{array}
$$

In order to assure that words of length i do not cheat, we have a variable $h_{i,q,q'}$ for all $q,q' \in Q$, and a variable $g_{i,q,q',\sigma}$ for all $q,q' \in Q$ and $\sigma \in \Sigma$. To reflect Equation (1), we have, for all $p,p' \in Q$ and $\sigma' \in \Sigma$ such that $\Delta(p,\sigma,q)$ and $\Delta(p',\sigma',q')$, the constraint

$$g_{i,q,q',\sigma} \geq h_{i-1,p,p'} + c(p,\sigma,q) - c(p',\sigma',q') - \theta(\sigma,\sigma').$$

To reflect Equation (2), we have, for all $q,q' \in Q$ and $\sigma \in \Sigma$ for which $g_{i,q,q',\sigma}$ is bounded, the constraint $h_{i,q,q'} \geq g_{i,q,q',\sigma}$. Also, for all $q,q' \in Q$ we have the constraints $h_{i,q,q'} \geq h_{i-1,q,q'}$. Finally, for all $q,q' \in Q$

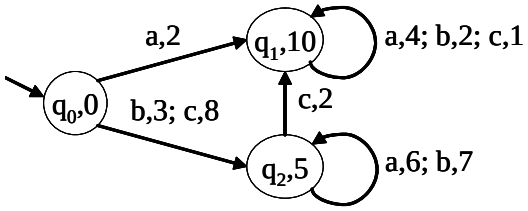

Figure 4: The DWFA $\mathcal{A}$.

we have the constraint $h_{i,q,q'} \leq \tau(q') - \tau(q)$. This last type of constraints, considering the final costs, corresponds to the comparison done in Step 1 of the resilience-testing algorithm.

For example, for the DWFA presented in Figure 4, the following are the constraints relevant to words of length 2 that without cheating must get to q_2 but consider getting to q_1. Since words of length 1 can only reach q_1 or q_2 and $\Delta(q_1, c, q_2) = \Delta(q_2, c, q_2) = \emptyset$, there are no constraints involving the variable $g_{2,q_2,q_1,c}$.

$$
\begin{array}{ll}
g_{2,q_2,q_1,a} \geq h_{1,q_2,q_2} + 6 - 2 - \theta(a,c) & g_{2,q_2,q_1,b} \geq h_{1,q_2,q_1} + 7 - 4 - \theta(b,a) \\
g_{2,q_2,q_1,a} \geq h_{1,q_2,q_1} + 6 - 4 - \theta(a,a) & g_{2,q_2,q_1,b} \geq h_{1,q_2,q_1} + 7 - 2 - \theta(b,b) \\
g_{2,q_2,q_1,a} \geq h_{1,q_2,q_1} + 6 - 2 - \theta(a,b) & g_{2,q_2,q_1,b} \geq h_{1,q_2,q_1} + 7 - 1 - \theta(b,c) \\
g_{2,q_2,q_1,a} \geq h_{1,q_2,q_1} + 6 - 1 - \theta(a,c) & g_{2,q_2,q_2,b} \geq h_{1,q_2,q_1} + 7 - 2 - \theta(b,c) \\
h_{2,q_2,q_1} \geq g_{2,q_2,q_1,a} & h_{2,q_2,q_1} \geq h_{1,q_2,q_1} \\
h_{2,q_2,q_1} \geq g_{2,q_2,q_1,b} & h_{2,q_2,q_1} \leq 10 - 5
\end{array}
$$

For $k = n^2 + 1 \ldots 2k^2$, the set of variables and the set of constraints are very similar to these sets for $k \leq n^2$. The only difference is the last type of constraints for every $q, q' \in Q$. Instead of $h_{i,q,q'} \leq \tau(q') - \tau(q)$, we have $h_{i,q,q'} \leq h_{i-1,q,q'}$. These constraints corresponds to the detection of gain increasing cycles, done in step 2 of the resilience testing algorithm.

The correctness of the following claim follows from the construction of the constraints.

Claim 4.2. *The set of penalty functions in all feasible solutions to the LP is identical to the set of penalty functions for which the resilience algorithm provides a positive answer.*

In particular, the feasible solution for which $\sum_{\sigma,\sigma'} \theta_{\sigma,\sigma'}$ is minimized, corresponds to a penalty function with minimal total budget. The total number of constraints and variables in our LP is polynomial in $|Q|$ and $|\Sigma|$. Therefore, it is possible to find an optimal solution for it [Kha79, Chv83] in polynomial time. This implies a polynomial algorithm for the minimum cost resilience problem of a DWFA.

Acknowledgment We thank Pnina and Yosef Bernholtz for many helpful discussions.

References

[AKL09] B. Aminof, O. Kupferman, and R. Lampert. Reasoning about online algorithms with weighted automata. In *Proc. 20th ACM-SIAM Symp. on Discrete Algorithms*, pages 835–844, 2009.

[CCH+05] A. Chakrabarti, K. Chatterjee, T.A. Henzinger, O. Kupferman, and R. Majumdar. Verifying quantitative properties using bound functions. In *Proc. 13th Conf. on Correct Hardware Design and Verification Methods*, volume 3725 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.

[CDH08] K. Chatterjee, L. Doyen, and T. Henzinger. Quantative languages. In *Proc. 17th Annual Conf. of the European Association for Computer Science Logic*, pages 385–400, 2008.

[Chv83] V. Chvatal. *Linear Programming*. W.H. Freeman and Company, 1983.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DKe09] M. Droste, W. Kuich, and H. Vogler (eds.). *Handbook of Weighted Automata*. Springer, 2009.

[Eil74] S. Eilenberg. *Automata, Languages and Machines*. Academic Press, San Diego, 1974.

[FKL10] D. Fisman, O. Kupferman, and Y. Lustig. Rational synthesis. In *Proc. 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer, 2010.

[Hen07] T.A. Henzinger. Quantitative generalizations of languages. In *Development in Language Theory*, pages 20–22, 2007.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. In K. Apt, editor, *Logics and Models of Concurrent Systems*, volume F-13 of *NATO Advanced Summer Institutes*, pages 477–498. Springer, 1985.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[Kro94] D. Krob. The equality problem for rational series with multiplicities in the tropical emiring is undecidable. *Journal of Algebra and Computation*, 4:405–425, 1994.

[Moh97] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[NR99] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 129–140, 1999.

[NRTV07] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[SS78] A. Salomaa and M. Soittola. *Automata: Theoretic Aspects of Formal Power Series*. Springer-Verlag New York, Inc., 1978.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.