Optimal Repair for Omega-Regular Properties

Vrunda Dave[1], Shankara Narayanan Krishna[1], Vishnu Murali[2](✉), and Ashutosh Trivedi[2]

[1] Indian Institute of Technology Bombay, Mumbai, India
[2] University of Colorado Boulder, Boulder, USA
vishnu.murali@colorado.edu

Abstract. This paper presents an optimization based framework to automate system repair against omega-regular properties. In the proposed formalization of *optimal repair*, the systems are represented as Kripke structures, the properties as ω-regular languages, and the repair space as *repair machines*—weighted omega-regular transducers equipped with Büchi conditions—that rewrite strings and associate a cost sequence to these rewritings. To translate the resulting cost-sequences to easily interpretable payoffs, we consider several aggregator functions to map cost sequences to numbers—including limit superior, supremum, discounted-sum, and average-sum—to define quantitative cost semantics. The problem of optimal repair, then, is to determine whether traces from a given system can be rewritten to satisfy an ω-regular property when the allowed cost is bounded by a given threshold. We also consider the dual challenge of *impair verification* that assumes that the rewritings are resolved adversarially under some given cost restriction, and asks to decide if all traces of the system satisfy the specification irrespective of the rewritings. With a negative result to the impair verification problem, we study the problem of designing a minimal mask of the Kripke structure such that the resulting traces satisfy the specifications despite the threshold-bounded impairment. We dub this problem as the *mask synthesis* problem. This paper presents automata-theoretic solutions to repair synthesis, impair verification, and mask synthesis problem for limit superior, supremum, discounted-sum, and average-sum cost semantics.

1 Introduction

Given a Kripke structure and an ω-regular specification, the model checking problem is to decide whether all traces of the system satisfy the specification. Vardi and Wolper [17] initiated the automata-theoretic approach to model-checking by reducing the ω-regular model checking problem to the language inclusion problem. If the system violates the specification, this approach returns a simple lasso-shaped counterexample demonstrating the violation. While these

---

This work was supported by the National Science Foundation (NSF) under Grant ECCS-2015403 and NSF CAREER award CCF-2146563.

A. Bouajjani et al. (Eds.): ATVA 2022, LNCS 13505, pp. 354–370, 2022.
https://doi.org/10.1007/978-3-031-19992-9_23

counterexamples often aid the designer in manually repairing the system, this repair process can be exhausting and error-prone. Moreover, different repair policies may incur different costs rendering the repair problem a non-trivial optimization problem. *This paper investigates a range of problems in synthesizing optimal repair policies against ω-regular specification.*

As a concrete motivation for various repair problems, we consider security issues (confidentiality and availability) in manufacturing. It is well documented [7] that acoustic side-channels leak valuable intellectual property information during the manufacturing process. Consider a 3D printer which can print either squares or triangles. Since the movement of the stepper motors of the printer vary based on the design, this difference in movement leads to the printer producing different sounds. Thus, an intruder may be able to discern the shape being printed by observing the audio output of the system as it acts as an acoustic side-channel. One can model such a system as a *Kripke structure*: a mockup of such systems is represented in Fig. 1a where the label corresponds to the state being idle ($\perp$), printing squares ($\square$), or printing triangles ($\triangle$).

Suppose that the system designer wishes to protect the information that a given printer prints only a fixed number of objects of one shape, or the sequence in which these shapes appear, from an eavesdropper. This specification, and a rich class of similar specifications on the observations, can be captured using ω-regular languages (see the Büchi automaton of Fig. 1b which requires that both shapes are printed infinitely often), and one can verify if the system satisfies such a specification using classical model checking. It is easy to see that our system does not satisfy this property for all traces. To repair this situation, we may wish to add spurious motor rotations to mimic the other shape, but adding such rotations comes with a cost (say energy or time overheads). The choices and cost available for repair can intuitively be expressed as a repair machine (a weighted nondeterministic transducer) given in Fig. 1c. For example, the label $\square|\square\triangle, 3$ represents the situation where the repair machine modifies the observation corresponding to a square shape by appending a spurious rotation mimicking a triangle shape with an extra cost of 3 units.

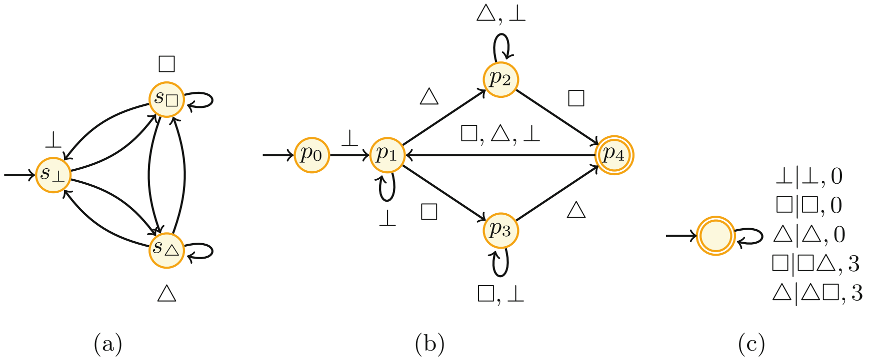


Fig. 1. (a) Krikpe structure representing the 3D printer system, (b) Büchi automaton B specifying the property, and (c) Repair machine

A key synthesis problem, then, is to compute a minimum cost repair strategy to add these spurious rotations such that the system after repair satisfies the specification. The cost of an ω-sequence can be aggregated using discounted-sum, average-sum, liminf, limsup, inf, or sup. We call this problem the *repair synthesis* where the goal is given an aggregator and cost threshold, design a strategy on the nondeterministic transducer such that every trace of the system can be written to satisfy the specification with cost bounded by the given threshold.

Example 1. Consider the repair machine T from Fig. 1c with the average-sum cost semantics and a threshold of 2. For every spurious motor rotation, T incurs a cost of 3 units of power. Note that a strategy of replacing every $\triangle$ with $\triangle\square$, maps $\bot\triangle^\omega$ to $\bot(\triangle\square)^\omega$ which is accepted by B. The mean cost of this rewrite is 3 and is above threshold. However, there exists a strategy that rewrites $\bot\triangle^\omega$ to $\bot(\triangle\triangle\triangle\square)^\omega$ that is accepted by B, with a mean cost equal to 1.

A related problem is that of *impair verification* that is connected to availability vulnerabilities. Consider an attack model in the aforementioned 3D manufacturing setting where an attacker with bounded capabilities controls the rewriting process (by introducing subtle undetectable changes in the manufacturing process) and intends to rewrite the traces in such a way that the resulting trace satisfy some undesirable behavior (to make the acoustic profile violate some regulatory norms) with a cost bounded below a threshold. Such undesirable rewritings may impair the capabilities of the system and render it unavailable for normal use. The impair verification problem is to verify whether the system is safe from such adversarial rewritings.

If the system is found to be vulnerable to impair and the system designer has no control over the rewriting process, a viable mitigation approach is to minimally restrict the behavior of the system to harden it against the adversarial rewriting. We formalize this problem as the *mask synthesis* problem.

Contributions. We consider repair machines to be specified as weighted ω-transducers and study various optimal repair problems for different aggregator functions. As we deal with reactive systems, we consider cost semantics that aggregate infinite sequence of costs to a scalar via aggregator functions discounted sum, average sum, limit superior, and supremum. We formalize and study the following problems related to optimal repair:

- **Repair Synthesis.** Given a system, an ω-regular specification, a repair machine, and a cost semantics, decide whether there exists a strategy to rewrite traces of the system to satisfy the specification within a given threshold.
- **Impair Verification.** Given a system, an ω-regular property capturing the undesirable behaviors, a repair machine, a cost semantics, decide whether there exist a trace of the system that satisfy the undesirable behavior under adversarial rewritings within a given threshold.
- **Mask Synthesis.** Given a system, an ω-regular property (undesirable behaviors), a repair machine, a cost semantics, find a minimal restriction of the system such that no remaining trace of the system satisfy the undesirable behavior under any adversarial rewritings within the threshold.

Our work is inspired by the idea of weighted transducers studied in [10] for finite strings. The notions of robust verification and kernel synthesis studied in [10] are templates for the impair verification and mask synthesis problems studied here, but the present setting requires extension of those results to the setting of ω-words: this is one of the secondary contributions of this paper.

Our results imply that the results presented in [10] carry over to the setting of ω-words for the discounted-sum and mean cost-semantics, the robust verification problem for both of these can be decided in P (cf. Theorems 5 and 6), while the robust kernel for discounted-sum cost-semantics is ω-regular if the language of the Kripke structure is a cut-point language (cf. Theorem 8). Furthermore, the notion of repair synthesis, to the best of our knowledge, is yet unexplored. We characterize the complexity of repair synthesis (Theorems 2–4) and impair verification problems (Theorems 5–7), and for the mask synthesis problem we discuss which aggregators allow ω-regular mask (Theorems 8–9).

Proofs of the theorems can be found in the technical report [9].

2 Preliminaries

Let Σ denote a finite alphabet. We write Σ^ω and Σ^* for the set of infinite and finite words over Σ. We denote an empty string by ϵ.

Kripke Structures. A *Kripke structure* is a tuple $K = (S, \hookrightarrow, S_0, AP, \mathcal{L})$ where S denotes a set of states, $\hookrightarrow \subseteq S \times S$ is the transition relation, $S_0 \subseteq S$ is the set of initial states, AP is the set of atomic propositions, and $\mathcal{L} : S \to 2^{AP}$ denotes the labeling function. An infinite sequence of states $\pi = s_0 s_1 \ldots \in S^\omega$ is said to be a path of the Kripke structure if $(s_i, s_{i+1}) \in \hookrightarrow$ for all $i \in \mathbb{N}$. Let $\Sigma = 2^{AP}$. The labeling function applied to a path $\pi = s_0 s_1 \ldots \in S^\omega$ defines traces $\mathcal{L}(\pi) = a_0 a_1 \ldots \in \Sigma^\omega$ of K where for each $i \geq 0$ we have that $a_i = \mathcal{L}(s_i)$. We use $\mathcal{T}_K$ to indicate the set of all traces of K.

Omega-Regular Specifications. A *non-deterministic Büchi automaton* (NBA) over Σ is a tuple $A = (Q, \Sigma, Q_0, Q_f, \delta)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $Q_f \subseteq Q$ is the set of final states, Σ is the finite input alphabet, and $\delta \subseteq Q \times \Sigma \times Q$ denotes the transition relation. We define the extended transition relation $\widehat{\delta} \subseteq Q \times \Sigma^* \times Q$ in the standard fashion, i.e. $(q, \epsilon, q) \in \widehat{\delta}$ for $q \in Q$ and $ax \in \Sigma\Sigma^*$ we have $(q, ax, q') \in \widehat{\delta}$ if there exists $q'' \in Q$ such that $(q, a, q'') \in \delta$ and $(q'', x, q') \in \widehat{\delta}$.

A run ρ over a word $w = w_0 w_1 \ldots \in \Sigma^\omega$ is an infinite sequence of states $q_0, q_1 \ldots$ such that $(q_i, w_i, q_{i+1}) \in \delta$. A run ρ is accepting iff some final state from Q_f occurs infinitely often in ρ. The language defined by the automaton A, denoted as $L(A)$, is the set of words w over Σ^ω such that there exists an accepting run of w by A.

Cost Aggregation Semantics. An aggregator function $\oplus : \mathbb{N}^\omega \to \mathbb{Q}_{\geq 0}$ maps infinite sequences of numbers to a scalar. Let $\tau = \tau_1 \tau_2 \cdots \in \mathbb{N}^\omega$ with each $\tau_i \in \mathbb{N}$. We consider the following aggregators:

- $\mathsf{DSum}_\lambda \stackrel{\text{def}}{=} \overline{\tau} \mapsto \lim_{n\to\infty} \sum_{i=1}^{n} \lambda^{i-1}\tau_i$, with discount factor $0 \le \lambda < 1$,
- $\mathsf{Mean} \stackrel{\text{def}}{=} \overline{\tau} \mapsto \limsup_{n\to\infty}(1/n) \cdot \sum_{i=1}^{n} \tau_i$,
- $\mathsf{Sup} \stackrel{\text{def}}{=} \overline{\tau} \mapsto \sup\{\tau_i \mid i \in \mathbb{N}\}$, and
- $\mathsf{LimSup} \stackrel{\text{def}}{=} \overline{\tau} \mapsto \limsup\{\tau_i \mid i \in \mathbb{N}\}$.

Quantitative Games. A *game arena* $\mathcal{G} = (G, V_{\text{Min}}, V_{\text{Max}})$ consists of a graph $G = (V, E, w)$ where V is a finite set of vertices, $E \subseteq V \times V$ is the set of edges, $w : E \to \mathbb{N}$ is the weight function. The sets V_{Max} and V_{Min} characterize a partition of the vertex set V such that player Min controls the edges from vertices in V_{Min}, while Max controls the vertices in V_{Max}.

A play of the game $\mathcal{G}$ is an infinite sequence of vertices $\pi = \langle v_0, v_1, \ldots\rangle$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. A finite play is a finite such sequence, that is, a sequence in V^*. We denote by $\mathsf{last}(\pi)$ the final vertex in the finite play π. We write $\mathsf{Play}_{\mathcal{G}}$ and $\mathsf{FPlay}_{\mathcal{G}}$ for the set of infinite and finite plays of the game arena $\mathcal{G}$, respectively. A strategy of player Min in $\mathcal{G}$ is a partial function $\sigma : \mathsf{FPlay} \to V$ defined over all plays $\pi \in \mathsf{FPlay}$ with $last(\pi) \in V_{\text{min}}$, such that we have $(\mathsf{last}(\pi), \sigma(\pi)) \in E$. A strategy χ of player Max is defined analogously. We say that a strategy σ is *positional* if $\mathsf{last}(\pi) = \mathsf{last}(\pi')$ implies $\sigma(\pi) = \sigma(\pi')$. Strategies that are not positional are called *history dependent*. Let Σ_{Min} and Σ_{Max} be the sets of all strategies of player Min and player Max, respectively. We write Π_{Min} and Π_{Max} for the set of positional strategies of player Min and player Max, respectively. For a game arena $\mathcal{G}$, vertex v of $\mathcal{G}$ and strategy pair $(\sigma, \chi) \in \Sigma_{\text{Min}} \times \Sigma_{\text{Max}}$, let $\mathsf{Play}^{\sigma,\chi}(v)$ be the infinite play starting from v in which player Min and Max play according to σ and χ, respectively.

The weight function $w : E \to \mathbb{N}$ can be naturally extended from edges to plays as $w : \mathsf{Play}_{\mathcal{G}} \to \mathbb{N}^\omega$ as $\pi \mapsto c_0 c_1 \ldots$ where $c_i = w(v_i, v_{i+1})$ for all $i \in \mathbb{N}$. Given an aggregator function $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, we define the payoff of player Min to player Max for a play π as $\oplus(w(\pi))$. Depending on the choice of the aggregator function $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, we refer to the game as $\oplus$-game. In a $\oplus$-game, the goal of player Min is to choose her actions in such a way so as to minimize the payoff, while the goal of player Max is to maximize the payoff. For every vertex $v \in V$, define the *upper value* $\overline{\mathsf{Val}}_\oplus(\mathcal{G}, v)$ as the minimum payoff player Min can ensure irrespective of player Max's strategy. Symmetrically, the *lower value* $\underline{\mathsf{Val}}_\oplus(\mathcal{G}, v)$ of a vertex $v \in V$ is the maximum payoff player Max can ensure irrespective of player Min's strategy.

$$\overline{\mathsf{Val}}_\oplus(\mathcal{G}, v) = \inf_{\sigma\in\Sigma_{\text{Min}}} \sup_{\chi\in\Sigma_{\text{Max}}} \oplus(w(\mathsf{Play}^{\sigma,\chi}(v)))$$
$$\underline{\mathsf{Val}}_\oplus(\mathcal{G}, v) = \sup_{\chi\in\Sigma_{\text{Max}}} \inf_{\sigma\in\Sigma_{\text{Min}}} \oplus(w(\mathsf{Play}^{\sigma,\chi}(v))).$$

The inequality $\underline{\mathsf{Val}}_\oplus(\mathcal{G}, v) \le \overline{\mathsf{Val}}_\oplus(\mathcal{G}, v)$ holds for all two-player zero-sum games. A game is *determined* when, for every vertex $v \in V$, the lower value and upper value are equal. In this case, we say that the value of the game $\mathsf{Val}_\oplus$ exists with

$\mathsf{Val}_\oplus(\mathcal{G}, v) = \underline{\mathsf{Val}}_\oplus(\mathcal{G}, v) = \overline{\mathsf{Val}}_\oplus(\mathcal{G}, v)$ for every $v \in V$. For strategies $\sigma \in \Sigma_{\mathrm{Min}}$ and $\chi \in \Sigma_{\mathrm{Max}}$ of players Min and Max, we define their values Val^σ and Val^χ as

$$\mathsf{Val}^\sigma_\oplus : v \mapsto \sup_{\chi \in \Sigma_{\mathrm{Max}}} \oplus(w(\mathsf{Play}^{\sigma,\chi}(v))) \text{ and}$$
$$\mathsf{Val}^\chi_\oplus : v \mapsto \inf_{\sigma \in \Sigma_{\mathrm{Min}}} \oplus(w(\mathsf{Play}^{\sigma,\chi}(v))).$$

A strategy σ_* of player Min is called *optimal* if $\mathsf{Val}^{\sigma_*}_\oplus = \mathsf{Val}_\oplus$. Likewise, a strategy χ_* of player Max is optimal if $\mathsf{Val}^{\chi_*}_\oplus = \mathsf{Val}_\oplus$. We say that a game is *positionally determined* if both players have positional optimal strategies.

Theorem 1 [4,19]. *For* $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$*,* $\oplus$*-games are determined in positional strategies. The complexity of solving is in* $\mathsf{NP} \cap \mathsf{co\text{-}NP}$ *for* DSum_λ*-games and* Mean*-games, and, is in P for* Sup*-games and* LimSup*-games.*

The goal of the player Min in a Büchi game [6] over a game arena $\mathcal{G}$ and a set $F \subseteq V$ is to choose her actions in such a way that some vertex $v_f \in F$ occurs infinitely often in the play, while the goal of the Max player is to prevent this. We note from [4] that LimSup-games generalize Büchi games. For Theorem 1 it follows that the winning region, i.e. the set of vertices where the player Min has a strategy to win can be computed in P.

3 Problem Definition

Just as weighted transducers extend finite state automata with outputs and costs on transitions, NBAs can be extended to *weighted non-deterministic Büchi transducers* by adding an output word and costs to transitions. We define a repair machine as a weighted non-deterministic Büchi transducer equipped with a cost aggregation. We introduce repair machines and their computational problems.

Definition 1. *A* repair machine *(RM)* T *is a tuple* $(Q, \Sigma, Q_0, Q_f, \Gamma, \delta, \oplus)$ *where* Q *is a finite set of states,* $Q_0 \subseteq Q$ *is the set of initial states,* $Q_f \subseteq Q$ *is the set of final states,* Γ *is the output alphabet,* $\delta \subseteq Q \times \Sigma \times Q \times \Gamma^* \times \mathbb{N}$ *is the transition relation, and* $\oplus$ *is the cost aggregator function.*

For a given aggregator function $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, we refer to a repair machine as DSum-RM, Mean-RM, Sup-RM, LimSup-RM.

A transition $(q, a, q', w, c) \in \delta$ indicates that, the transducer on reading the letter $a \in \Sigma$ in state q, transitions to state q', and outputs a word $w \in \Gamma^*$, incurring a cost c for rewriting a to w. We write $q \xrightarrow{a/w}_c q'$ if $(q, a, q', w, c) \in \delta$. A run ρ of T on $u = a_1 a_2 \cdots \in \Sigma^\omega$ is a sequence $\langle q_0, (a_0, w_0, c_0), q_1, (a_1, w_1, c_1), \ldots \rangle$ where for every $i \geq 0$ we have that $q_0 \in Q_0$ and $q_i \xrightarrow{a_i/w_i}_{c_i} q_{i+1}$. Let $\mathsf{Runs}(T, u)$ be the set of runs of T on u. We write $\mathcal{O}(\rho)$ and $\mathcal{C}(\rho)$ for the projection on the outputs and cost sequences, i.e. $\mathcal{O}(\rho) = w_0 w_1 \ldots$ and $\mathcal{C}(\rho) = c_0 c_1 \ldots$, of a run ρ of T. We say that a run of T is accepting if states from Q_f are visited infinitely often. We write $\mathsf{dom}(T)$ for the set of all words which have an accepting run.

We define three different semantics for T. The function $[\![T]\!](u)$ returns the set of all pairs of outputs and cost sequences over the word $u \in \Sigma^\omega$; the function $[\![T]\!]^{\oplus}_{*}(u, v)$ returns the optimal rewriting cost w.r.t the aggregator function $\oplus$ over T for a rewriting of u to v; and $[\![T]\!]^{\oplus}_{\tau}(u)$ returns the set of all rewritings of a word u with cost bounded by a threshold $\tau \in \mathbb{R}$.

$$[\![T]\!](u) = \{(\mathcal{O}(\rho), \mathcal{C}(\rho)) \;:\; u \in \mathsf{dom}(T) \text{ and } \rho \in \mathsf{Runs}(T, u)\},$$
$$[\![T]\!]^{\oplus}_{*}(u, v) = \inf\{\oplus(\mathcal{C}(\rho)) \;:\; \rho \in \mathsf{Runs}(T, u) \text{ and } \mathcal{O}(\rho) = v\},$$
$$[\![T]\!]^{\oplus}_{\tau}(u) = \left\{\mathcal{O}(\rho) \;:\; \rho \in \mathsf{Runs}(T, u) \text{ and } [\![T]\!]^{\oplus}_{*}(u, \mathcal{O}(\rho)) \le \tau)\right\}.$$

Problems of Optimal Repair. Given the Kripke structure K representing the system, the ω-regular specification specified by the language $L \subseteq \Gamma^\omega$, a RM T, a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, and a threshold $\tau \in \mathbb{Q}_{\ge 0}$, the *repair synthesis* problem asks if there exists a strategy of rewriting every trace $t \in \mathcal{T}_K$ to some word $w \in L$ using T such that cost is at most τ.

We restrict the repair policies where Player Min is restricted to rewrite a letter of the trace based on history and not to rely on a lookahead. We give a game semantics to the repair synthesis problem as a turn-based two player game between players Min and Max that proceeds as follows. The game begins with player Max selecting the initial state $s_0 \in S_0$ of the Kripke structure and ends her turn. Player Min, starts from the initial state q_0 of the RM and then selects a valid rewriting w_i of $\mathcal{L}(s_0)$ such that $(q_0, \mathcal{L}(s_0), q'_i, w_i, c) \in \delta$ is a valid transition for some $c \in \mathbb{N}$ and changes the state of the RM to q'_i, she then ends her turn. The game continues in this fashion, where player Max selects the next state s'_i of the Kripke structure and Player Min selects a valid rewriting and thus the next state of the repair machine. This turn based game proceeds indefinitely and results in Player Max selecting a trace $t \in \mathcal{T}_K$ and player Min selecting a word $w \in \mathbb{N}^\omega$. Player Min wins the game if $w \in [\![T]\!]^{\oplus}_{\tau}(t)$, and $w \in L$, otherwise player Max wins the game. The existence of a winning strategy for Player Min implies the existence of a repair strategy.

Definition 2 (Repair Synthesis). *Given a Kripke structure K representing the system, an ω-regular specification L, a repair machine T, a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, and a threshold τ decide whether there exists a strategy to rewrite every trace $t \in \mathcal{T}_K$ to some word $w \in L$ with a cost of at most τ, and if so synthesise this strategy.*

We also consider the dual challenge of *impair verification* where the system is subjected to adversarial rewritings. This setting has applications in, among others, availability vulnerability detection. We consider an attack model where the rewritings given by the repair machine are resolved adversarially but are restricted to be within a given cost. The verification problem is to decide if there exists traces of the system that satisfy an ω-regular property capturing the undesirable behaviors for some such rewritings. The game semantics for the impair verification problem are similar to that of repair synthesis, however in the case of impair verification the player Max not only controls the selection of the next state s'_i, but also decides the rewriting by selecting the word w'_i as well.

Definition 3 (Impair Verification). *Given a structure K representing the system, an ω-regular language L capturing the undesirable behavior given as an NBA A, repair machine T, a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, and a threshold $\tau \in \mathbb{Q}_{\geq 0}$, the* impair verification *problem fails if there exists a trace $t \in \mathcal{T}_K$ that can be rewritten to some word $w \in L$ with a cost of at most τ under an adversarial strategy.*

When one may not be able to pass the impair verification problem, it may be desirable to design a way to minimally mask the Kripke structure such that the resulting system satisfies the specifications despite the threshold-bounded impairment. In such a case, we wish to find the maximal subset N' of traces which, even under adversarial rewrites, satisfy the ω-regular specification L.

Definition 4 (Mask Synthesis). *Given a Kripke structure K representing the system, an ω-regular language L capturing the undesirable behavior given as an NBA A, repair machine T, a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, and $\tau \in \mathbb{Q}_{\geq 0}$, the problem of* mask synthesis *is to find a maximal subset $N' \subseteq \mathcal{T}_K$ such that all traces $t \in N'$ pass the impair verification.*

The next three sections present our results on these three problems.

4 Repair Synthesis

To solve the problem of repair synthesis, we reduce it to a related problem of *threshold synthesis*. Threshold synthesis asks for a partition of the rational numbers $\mathbb{Q}_{\geq 0}$ into sets $\mathbb{G}$ (good) and $\mathbb{B}$ (bad) sets such that the repair synthesis problem can be solved for all good thresholds $\tau \in \mathbb{G}$. Given a system K, the specification $L \subseteq \Gamma^\omega$ represented by an NBA B, a repair machine T, and a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, we focus on the threshold synthesis problem: find a partition of $\mathbb{Q}_{\geq 0}$ into two sets $\mathbb{G}$ and $\mathbb{B}$ such that the policy synthesis can be solved for all $\tau \in \mathbb{G}$. We note that in the case of policy synthesis, the sets $\mathbb{G}$ and $\mathbb{B}$ are upward and downward closed respectively. If player Min has a winning strategy for some $\tau \in \mathbb{Q}_{\geq 0}$ then she may use the same strategy for all $\tau' \geq \tau$. Let the infimum value τ for which player Min wins be denoted as τ^*, then $\mathbb{G} = [\tau^*, \infty)$ and $\mathbb{B} = [0, \tau^*)$. We call this value τ^* the optimal threshold.

4.1 Solving the Büchi Games

Our approach to compute the optimal threshold is to first restrict the choice of player Min to those where she has a strategy to win with respect to the Büchi objective, irrespective of the choices of Player Max on the Kripke structure. If Player Min has no valid strategy to rewrite a trace of the system to satisfy the Büchi objective, then the optimal threshold $\tau^* = \infty$. We thus consider the case when $\tau^* \neq \infty$ by playing a Büchi game on a game arena and then pruning it.

To construct the game arena, we first construct the synchronized product $K \times T \times B$ of K, T, and B. Intuitively, $K \times T \times B$ accepts those traces of the system, which have some rewriting that is in L.

Definition 5. *The synchronized product* $K \times T \times B$ *of the Kripke Structure* $K = (S, \hookrightarrow, S_0, \mathcal{L})$*, the repair machine* $T = (Q, \Sigma, Q_0, Q_f, \Gamma, \Delta, C)$ *and the NBA* $B = (P, \Gamma, P_0, P_f, \delta)$ *is a weighted (directed) graph* $G^\times = (V, E, W, V_I, V_F)$*, where:*

- $V = S \times Q \times P \times \{1, 2\}$ *is the set of vertices consisting of states of the system* K*, repair machine* T*, and NBA* B*, and a counter that tracks the visitation of accepting states of* T *and* B *(like the degeneralization construction for the generalized Büchi automata)*
- $E \subseteq V \times V$ *is such that* $((s, q, p, i), (s', q', p', i')) \in E$ *if* $(s, s') \in \hookrightarrow$ *is a transition in* K*, for some* $w \in \Gamma^*$ *and* $c \in \mathbb{N}$ *transition* $(q, \mathcal{L}(s), q', w, c) \in \Delta$ *is in* T*, and* $(p, w, p') \in \widehat{\delta}$ *is a transition in* B*, and one of the following holds:*
  - $i = i' = 1$ *and* $q' \notin Q_f$
  - $i = i' = 2$ *and* $p \notin P_f$
  - $i = 1$ *and* $i' = 2$ *and* $q' \in Q_f$
  - $i = 2$ *and* $i' = 1$ *and* $p \in P_f$
- $W : E \to \mathbb{N}$ *is the weight function such that*

$$W((q, s, p, i), (q', s', p', i')) = \min\{c \,:\, (q, \mathcal{L}(s), q', w, c) \in \Delta\};$$

- $V_I \subseteq V = Q_0 \times S_0 \times P_0 \times \{1\}$ *is the set of initial vertices; and*
- $V_F \subseteq V = Q \times S \times P_f \times \{2\}$ *is the set of final vertices.*

To distinguish the choice of player Max and Min, we define a game structure $\mathcal{G}^\times$ on the product graph $G^\times$ by introducing intermediate states by appending another layer to the track counter. The formal construction is shown next.

The game graph $\mathcal{G}^\times = ((\overline{V}, \overline{E}, \overline{W}, V_I, V_F), \overline{V}_{\text{Min}}, \overline{V}_{\text{Max}})$ for product $G^\times = (V, E, W, V_I, V_F)$ is such that:

- $\overline{V} = S \times Q \times P \times \{1, 2, 3\}$;
- $\overline{E}$ is such that for $e = ((s, q, p, i)(s', q', p', i')) \in E$ we have two edges to separate the choice of the RM and the NBA from the Kripke structure:
  - $e_1 = ((s, q, p, i), (s, q', p', 3)) \in \overline{E}$ and
  - $e_2 = ((s, q', p', 3)(s', q', p', i')) \in \overline{E}$;

  with the weights $\overline{W}(e_1) = W(e)$ and $\overline{W}(e_2) = 0$;
- $\overline{V}_{\text{Min}} = S \times Q \times P \times \{1, 2\}$; and
- $\overline{V}_{\text{Max}} = S \times Q \times P \times \{3\}$.

Note that the first choice is made by player Max in choosing the starting state of the Kripke structure, and in the subsequent transitions player Min reads those states and makes a choice over the rewrites. For this reason, the choice of player Max appear to be lagging by one.

We play the Büchi-game on $\mathcal{G}^\times$ with the set of accepting states as V_F. We then prune the arena to contain only those states that are in the winning region of player Min with respect to the Büchi objective, that is, the set of states where player Min has a strategy to enforce visiting Büchi states irrespective of the strategy chosen by the player Max. We denote this pruned game arena as $\mathcal{G}$.

4.2 Optimal Threshold for DSum-RM

We reduce the problem of finding the optimal threshold τ^* for a DSum-RM to the problem of finding the value of a DSum-game on the game arena $\mathcal{G}$. As such we reduce the choices of selecting a trace by player Max and that of selecting a rewriting by player Min in the context of repair synthesis to choices made by the players in a DSum-game over an arena $\mathcal{G}$. In particular, we have the following.

Theorem 2. *The optimal threshold τ^* for the* DSum*-RM can be computed in NP $\cap$ co-NP via solving a* DSum*-game on $\mathcal{G}$.*

Proof (Sketch). We solve the DSum_γ game on $\mathcal{G}$ with $\gamma = \sqrt{\lambda}$, the value of this game corresponds to the optimal threshold τ^*, as each edge of the synchronized product is captured by a pair of edges in $\mathcal{G}$. For any $\varepsilon > 0$, Player Min has a strategy of following this DSum strategy, and then following the strategy of the Büchi-game such that the cost of this rewriting is $\tau^* + \varepsilon$.

4.3 Optimal Threshold for Mean-RM

Similar to the case of the DSum-RM, in the case of the Mean-RM, we reduce the problem of finding the optimal threshold τ^* to the problem of finding the value of a Mean-game on a game arena $\mathcal{G}$. However we note that unlike the case of the DSum-RMs we also need to ensure that the mean cost cycle is co-accessible from the accepting vertices. In particular we have the following result.

Theorem 3. *The optimal threshold τ^* for the* Mean*-RM can be computed in NP $\cap$ co-NP via solving a* Mean *game on $\mathcal{G}$.*

Proof (Sketch). The proof of this theorem is similar to that of Theorem 2. Here, we first find a least cost mean cycle that is co-accessible by Player Min from the winning strategy of the Büchi-game on $\mathcal{G}$ (either a cycle following some Mean-game or the Büchi cycle itself). To do so we determine vertex that is co-accessible along the Mean-game over $\mathcal{G}$ as well as the Büchi-game. Player Min then alternates between two strategies in rounds, the first, where she follows the strategy of the Mean-game and the second to where she follows the strategy of the Büchi-game. At any round i, she follows the strategy of the Mean-game until she cycles on the co-accessible vertex 2^i many times and then follows the strategy of the Büchi-game once to return to this vertex. As the least cost-cycle has twice the number of edges of the synchronized product we divide the value of the Mean-game by two to determine the optimal threshold. We note that the above strategy relies on infinite memory, however Player Min can restrict the number of rounds for any $\varepsilon > 0$, and so she has a finite memory policy to guarantee repair for any threshold of $\tau^* + \varepsilon$.

4.4 Optimal Thresholds for Sup-RMs and LimSup-RMs

In the case of the Sup aggregator function we first order the edges of $G^\times$ in the descending order of their weights and remove them in stages from the largest to the smallest. If, at any stage, the removal of edge e, leads to a failure of satisfying the Büchi condition, we infer that e is necessary to satisfy the Büchi condition for some state in G. We claim that the weight of the edge e is τ^*.

Similar to the Sup aggregator function, we start removing edges of $G^\times$ in the descending order of their weights only if they are present in an accepting cycle in the case of theLimSup aggregator function. Then, if at any stage, the removal of edge e, leads to a failure of satisfying the Büchi condition, we infer that the $\tau^* = W(e)$ and conclude that we can safely remove edges with a higher weight.

Theorem 4. *Computing optimal threshold* τ^* *for* Sup *and* LimSup-*RMs is in P.*

Proof (Sketch). Note that the removal of any edge e from the synchronized product that causes the Büchi-objective to no longer be satisfied guarantees that all the rewrite strategies for at least one trace do not satisfy the Büchi objective. Hence the removal prevents the satisfaction of either the acceptance of RM T or the NBA B, and in either case, leads to a trace of the Kripke structure that cannot be rewritten to some word that is accepted by the NBA B.

5 Impair Verification

Given the Kripke structure K representing the system, the ω-regular language L capturing undesirable behavior, represented as an NBA B, a repair machine T, and a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, we reduce the impair verification problem to the threshold verification problem. The threshold verification problem is to find a partition of $\mathbb{Q}_{\geq 0}$ into two sets $\mathbb{G}$ and $\mathbb{B}$, such that none of the traces to system can be rewritten to a word that is in the language of B for all $v \in \mathbb{G}$. Let τ^* denote the infimum value for which a trace $t \in \mathcal{T}_K$ can be rewritten to some word $w \in \Gamma^\omega$ such that $w \in L$. Then, the threshold verification problem is solved for any $\tau < \tau^*$, as $[\![T]\!]^\oplus_\tau(t) \not\subseteq L$ for every trace $t \in \mathcal{T}_K$. Thus the set $\mathbb{G} = (0, \tau^*)$ and the set $\mathbb{B} = [\tau^*, \infty)$ and problem reduces to finding the optimal threshold τ^*.

In order to find the optimal threshold τ^*, we construct the synchronised product $G^\times = K \times T \times B$ as detailed in Definition 5. We prune $G^\times$ to keep only those states from where player Max has a winning strategy against the Büchi objective. The construction is similar to Büchi games, except that the opponent has no choice. In the following, we refer to this pruned graph as G.

5.1 Optimal Threshold for DSum-RM

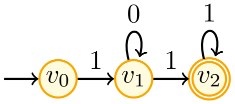
In the case of a DSum-RM, we show that the optimal threshold τ^* is the minimum infinite discounted cost path in G. While it may not be possible to achieve this cost, for any $\varepsilon > 0$ we show the existence of a finite memory strategy of player Max that guarantees that some rewriting with threshold of $\tau^* + \varepsilon$ is in L.

We claim that the optimal threshold τ^* is the minimum discounted cost in G. To find this value, we associate a variable $\mathcal{V}_s$, to each vertex $v \in V$, characterizing minimum discounted cost among all paths starting from the state s. The minimum discounted values can then be characterized as [15]:

$$\mathcal{V}_v = \min_{(v,v')\in E} \{W(v,v') + \lambda \cdot \mathcal{V}_{v'}\}$$

This equation can be computed by solving the following LP.

$$\max \sum_{v\in V} \mathcal{V}_v \text{ subject to:} \qquad \mathcal{V}_v \leq W(v,v') + \lambda \mathcal{V}_v \text{ for all } (v,v') \in E.$$

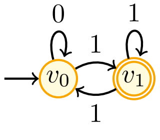


A positional discount-optimal strategy can be computed from the solutions of these equations simply by picking a successor vertex minimizing the right side of the optimality equations. Observe, however, that the resulting path may not satisfy the Büchi condition. Consider the graph shown in the inset (right). In order to satisfy the Büchi objective, a run must visit the state v_2, while to minimize the discounted cost the strategy is to cycle in the state v_1 getting a discounted sum of 1. While it is possible to achieve an ε-optimal discounted cost and satisfy the Büchi objective by looping on v_1 for an arbitrary number of steps before moving to the state v_2, no strategy satisfying the Büchi objective can achieve a DSum cost of 1.

Theorem 5. *The optimal threshold τ^* for* DSum*-RMs can be computed in P.*

5.2 Optimal Threshold for Mean-RM

In the case of the Mean aggregator function, we note that only those edges that are visited infinitely often have an effect on the cost. We say that a cycle is accepting if there exists some vertex $v \in V_F$ that occurs in the cycle. We let C_1 denote the least average cost cycle that can be reached and is reachable from some accepting cycle C_2. We use d_1 and d_2 to denote the total cost of these cycles and n_1 and n_2 to be the number of edges in each of them respectively. We then show that τ^* is the mean value of cycle C_1.



We observe that a strategy to determine this optimal threshold requires infinite memory. However for any $\varepsilon > 0$, there exists a finite memory strategy that is ε close to τ^*. Consider the graph shown in the inset (right) and the following strategy adopted by Player Max. Player Max cycles between v_0 and v_1 in rounds. At any given round i, Player Max cycles on v_0 for 2^i times, and then moves and cycles once in v_1 and returns to v_0. Observe this strategy ensures that the Büchi objective is satisfied while also ensuring the Mean cost to be 0 but requires infinite memory to keep track of the rounds. However, Player Max can achieve a ε-optimal mean cost by limiting the number of rounds.

Theorem 6. *The threshold τ^* for* Mean*-RMs can be computed in P.*

5.3 Optimal Thresholds for Sup-RMs and LimSup-RMs

For the Sup aggregator function, let S be the set of values c_i such that c_i is the supremum of the cost of some lasso that starts from some $v_i \in V_I$ and cycles in a loop containing some $v_f \in V_F$. Let k be the least element in S. We claim $\tau^* = k$. Similar to the Sup aggregator function, we consider the set S to contain the values c_i such that c_i is the supremum of the costs of the edges in the cycles that visit some $v_f \in V_F$ in the case of the LimSup aggregator function. We then take the least of these to be the optimal threshold for the LimSup-RMs.

Theorem 7. *The threshold τ^* for* Sup *and* LimSup-*RMs can be computed in P.*

6 Mask Synthesis

Given a Kripke structure K representing the system, an ω-regular language L capturing the undesirable behavior given as an NBA B, repair machine T, a cost semantics $\oplus \in \{\mathsf{DSum}_\lambda, \mathsf{Mean}, \mathsf{Sup}, \mathsf{LimSup}\}$, and $\tau \in \mathbb{Q}_{\geq 0}$, the problem of *mask synthesis* is to find a maximal subset $N' \subseteq \mathcal{T}_K$ such that all traces $t \in N'$ pass the impair verification.

It is well known that every Kripke structure admits an ω-regular language N such that a word $u \in N$ if and only if $u \in \mathcal{T}_K$. Let the ω-regular language of K be N. To solve the mask synthesis problem, we restrict the domain of the repair machine T to N by constructing a repair machine T' using product construction and give our results on the repair machine T'.

6.1 Mask Synthesis for *DSum*-RMs

We show that the maximal subset N' for isolated cut-point languages [3] is ω-regular. Given a threshold $\tau \in \mathbb{Q}$, the maximal subset N', is the set of all words $u \in \mathsf{dom}(T')$, such that for every word $w' \in [\![T]\!]_\tau^{\mathsf{DSum}}(u)$ we also have $w \notin L$. A threshold τ is ε-isolated for RM T', if for $\varepsilon > 0$ and all accepting runs r of T',

$$[\![T']\!]_*^{\mathsf{DSum}}(r, w) \in [0, v-\varepsilon] \cup [v+\varepsilon, \infty).$$

It is isolated if it is isolated for some ε. To prove that N' is ω-regular for such thresholds, we first note that isolated-cut point languages are ω-regular in the context of weighted automata [11]. We follow a similar strategy to [10], and slowly unroll our synchronous product. We note that since the repair machine is over ω strings, there must exist some n such that

$$\mathsf{DSum}(w_0 w_1 \ldots) \leq \mathsf{DSum}(w_0 \ldots w_n) + B_n,$$

where $B_n = V\frac{\lambda^n}{1-\lambda}$, where V is the largest cost that is not ∞. Therefore if $\mathsf{DSum}(w_0, w_1, \ldots) \leq v - \varepsilon + B_n$ we can conclude that $\mathsf{DSum}(w_0, \ldots, w_n) \leq v - \varepsilon$.

Lemma 1. *Let T' be a* DSum *repair machine and $\tau \in \mathbb{Q}$. If τ is ε-isolated for some ε, then there is $n^* \in \mathbb{N}$ such that any partial run r of length at least n^* satisfies one of the following properties:*

1. $\mathsf{DSum}(r) \leq \tau - \varepsilon$ *and* $\mathsf{DSum}(rr') \leq \tau - \varepsilon$ *for every infinite continuation* r' *of* r.
2. $\mathsf{DSum}(r) \geq \tau + \frac{\varepsilon}{2}$ *and* $\mathsf{DSum}(rr') \geq \tau + \varepsilon$ *for every infinite continuation* r' *of* r.

Here, for finite r, $\mathsf{DSum}(r)$ *is defined in the usual fashion except that the summation will be upto the length of* r.

Theorem 8. *Let* T' *be a* DSum *repair machine,* $v \in \mathbb{Q}$*, and* L *an* ω*-regular language given by an NBA. For all* n*, we can construct an NBA* A_n *such that* $L(A_n) \subseteq L(A_{n+1})$ *and* $L(A_n) \subseteq \overline{N'} \cap \mathsf{dom}(T')$*. Moreover, if* τ *is* ε*-isolated, there exists* n^* *such that* $L(A_{n^*}) = \overline{N'} \cap \mathsf{dom}(T')$.

For the construction of A_n in Theorem 8, a notion of bad and dangerous runs are defined. Intuitively, The bad runs are all those runs which are accepting with cost $\leq \tau$, such that the output word is not in L. The dangerous runs are the finite partial runs which can be extended to bad runs. The idea for construction of A_n is to identify all the finite partial runs r of length n which can later be extended to bad runs. This way we can construct a sequence of Büchi automata that better under approximate the automata for the non-robust words in the domain. Thanks to Lemma 1, we can assure that there exists a fixed point at n^* such that A_{n^*} recognizes all the non-robust words from T'.

6.2 Mask Synthesis for Mean-RMs

The mask synthesis problem for Mean-RMs is already undecidable for finite words [10, Theorem 17] and this result carries over to the case of ω-words.

6.3 Mask Synthesis for Sup-RMs and LimSup-RMs

For the Sup-RMs, we can construct an NBA recognizing all output words with a cost greater than τ and show that the maximal subset N' is ω-regular. The results for Sup-RMs can be extended carefully to only account the costs occurring in accepting loops and be used for the LimSup-RMs as well.

Theorem 9. *Let* T' *be a* Sup*-RM,* $\tau \in \mathbb{Q}$ *and* L *be a* ω*-regular language. The language of* N' *is* ω*-regular and we can effectively construct an NBA for it.*

7 Related Work

Our work is closest to the idea of weighted transducers as studied in [10] for finite strings. We extend the known results of [10] in the context of robust verification and kernel synthesis from finite strings to infinite strings.

D'Antoni, Samanta, and Singh [8] presented QLOSE, a program repair approach with quantitative objectives. The QLOSE approach permits rewriting syntactical expressions with arbitrary expressions while keeping the control structure of the program intact. In comparison, our approach permits modification of the control structure albeit with a finite set of expressions (encoded as a finite alphabet) considered for rewriting. Consequently, our setting remains decidable as opposed to repair with Qlose that is, in general, undecidable, and for tractability it restricts the correctness criterion to being correct over a given set of input-output examples. Similarly Samanta, Olivo, and Emerson [16], considered cost-aware program repair for Turing-complete programs through the use of predicate abstraction. However, their cost function is dependent only on the program location as opposed to more general ω-traces as proposed in our work.

Jobstmann, Griesmayer, and Bloem [13], and von Essen and Jobstman [18], studied program repair as a two-player game with qualitative ω-regular objectives. Our work, in contrast, allows quantitative notions of repair costs.

Cerny and Henzinger [2] championed for the need of partial program synthesis, which can be thought of as a repair, though its aim is to complete the given partial program, with respect to the specification. Although not directly related to repair, the framework of model measuring [12] presents a notion of distance between models; it studies the problem that given a model M and specification find the maximal distance such that all models within that distance from M satisfy the specification. Bansal, Chaudhuri, and Vardi [1] study comparator automata that read two infinite sequences of weights and relate their aggregate values to compare such quantitative systems. Kupferman and Tamir [14] consider the problem of cheating, where they use weighted automata and a penalty function to determine if the environment is cheating. The penalty function considered is again a map from a pair of letters to a value and so the environment is only permitted letter-to-letter rewritings. In contrast, our models permits more general letter-to-string rewritings constrained with ω-regular objectives.

Chatterjee et al. [5] consider the problem of solving both quantitative and qualitative objectives and define the notion of implication games where the objective is to solve both. While we provide direct proofs, Theorems 2 and 3 can also be recovered from results on implication games.

8 Conclusion

This paper presented a generalization of fundamental problems on weighted transducers and robustness threshold synthesis for ω-words. We proposed and solved the problem of minimal cost repair formulated as two player games on weighted transducers. We note that this problem is similar to multi-objectives optimization where the goal of the players is to satisfy an ω-regular property while optimizing a quantitative payoff. We also considered a related problem of impair verification that is related to availability problem where an attacker intends to rewrite the observations of the system to make it satisfy some undesirable behavior. We believe that the repair problem may find application in designing mitigation policies against side-channel vulnerability where some confidential property of the system is leaking in the output trace, and the goal is to find a minimum-cost repair to make the system opaque.

References

1. Bansal, S., Chaudhuri, S., Vardi, M.Y.: Comparator automata in quantitative verification. In: Baier, C., Dal Lago, U. (eds.) FoSSaCS 2018. LNCS, vol. 10803, pp. 420–437. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89366-2_23
2. Cerný, P., Henzinger, T.A.: From boolean to quantitative synthesis. In: International Conference on Embedded Software, EMSOFT 2011, pp. 149–154 (2011)
3. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 385–400. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87531-4_28
4. Chatterjee, K., Doyen, L., Henzinger, T.A.: A Survey of stochastic games with limsup and liminf objectives. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 1–15. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02930-1_1
5. Chatterjee, K., Henzinger, T.A., Otop, J., Velner, Y.: Quantitative fair simulation games. Inf. Comput. **254**, 143–166 (2017)
6. Chatterjee, K., Henzinger, T.A., Piterman, N.: Algorithms for Büchi games. arXiv preprint. arXiv:0805.2620 (2008)
7. Chhetri, S.R., Canedo, A., Faruque, M.A.A.: Confidentiality breach through acoustic side-channel in cyber-physical additive manufacturing systems. ACM Trans. Cyber-Phys. Syst. **2**(1), 1–25 (2017)
8. D'Antoni, L., Samanta, R., Singh, R.: Qlose: program repair with quantitative objectives. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9780, pp. 383–401. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_21
9. Dave, V., Krishna, S., Murali, V., Trivedi, A.: Optimal repair for omega-regular properties (2022). arxiv.org/abs/2207.13416
10. Filiot, E., Mazzocchi, N., Raskin, J., Sankaranarayanan, S., Trivedi, A.: Weighted transducers for robustness verification. In: International Conference on Concurrency Theory, CONCUR 2020, pp. 17:1–17:21 (2020)
11. Henzinger, T.A., Doyen, L., Chatterjee, K.: Expressiveness and closure properties for quantitative languages. In: Logic in Computer Science, Symposium on, pp. 199–208 (2009)
12. Henzinger, T.A., Otop, J.: From model checking to model measuring. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 273–287. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40184-8_20
13. Jobstmann, B., Griesmayer, A., Bloem, R.: Program repair as a game. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 226–238. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_23
14. Kupferman, O., Tamir, T.: Coping with selfish on-going behaviors. Inf. Comput. **210**, 1–12 (2012)
15. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. John Wiley & Sons Inc., USA (1994)
16. Samanta, R., Olivo, O., Emerson, E.A.: Cost-aware automatic program repair. In: Müller-Olm, M., Seidl, H. (eds.) SAS 2014. LNCS, vol. 8723, pp. 268–284. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10936-7_17

17. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Proceedings of the First Symposium on Logic in Computer Science, pp. 322–331. IEEE Computer Society (1986)
18. von Essen, C., Jobstmann, B.: Program repair without regret. Formal Methods Syst. Des. **47**(1), 26–50 (2015). https://doi.org/10.1007/s10703-015-0223-6
19. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoret. Comput. Sci. **158**(1), 343–359 (1996)