Weighted Transducers for Robustness Verification

Emmanuel Filiot
Université libre de Bruxelles
efiliot@ulb.ac.be

Nicolas Mazzocchi
Université libre de Bruxelles, BELGIUM

Jean-François Raskin
Université libre de Bruxelles, BELGIUM

Sriram Sankaranarayanan
University of Colorado Boulder, USA

Ashutosh Trivedi
University of Colorado Boulder, USA

Abstract

Automata theory provides us with fundamental notions such as languages, membership, emptiness and inclusion that in turn allow us to specify and verify properties of reactive systems in a useful manner. However, these notions all yield "yes"/"no" answers that sometimes fall short of being satisfactory answers when the models being analyzed are imperfect, and the observations made are prone to errors. To address this issue, a common engineering approach is not just to verify that a system satisfies a property, but whether it does so *robustly*. We present notions of robustness that place a metric on words, thus providing a natural notion of distance between words. Such a metric naturally leads to a topological neighborhood of words and languages, leading to quantitative and robust versions of the membership, emptiness and inclusion problems. More generally, we consider weighted transducers to model the cost of errors. Such a transducer models neighborhoods of words by providing the cost of rewriting a word into another. The main contribution of this work is to study robustness verification problems in the context of weighted transducers. We provide algorithms for solving the robust and quantitative versions of the membership and inclusion problems while providing useful motivating case studies including approximate pattern matching problems to detect clinically relevant events in a large type-1 diabetes dataset.

2012 ACM Subject Classification Computer systems organization → Dependable and fault-tolerant systems and networks; Theory of computation → Formal languages and automata theory

Keywords and phrases Weighted transducers, Quantitative verification, Fault-tolerance

Funding *Emmanuel Filiot*: Associate researcher at F.R.S.-FNRS
Nicolas Mazzocchi: PhD student funded by a FRIA fellowship from the F.R.S.-FNRS.

Acknowledgements This work is partially supported by the PDR project Subgame perfection in graph games (F.R.S.-FNRS), the MIS project F451019F (F.R.S.-FNRS), the ARC project Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond (Fédération Wallonie-Bruxelles), the EOS project Verifying Learning Artificial Intelligence Systems (F.R.S.-FNRS and FWO), the COST Action 16228 GAMENET (European Cooperation in Science and Technology), and the US National Science Foundation (NSF) under award numbers CPS 1836900 and CCF 1815983.

1 Introduction

Automata theoretic verification commonly uses an automaton S to specify the behaviors of a system being analyzed and another automaton P to specify the property of interest. These automata are assumed to be finite state machines accepting finite or infinite words. The key step is to verify whether the language inclusion $L(S) \subseteq L(P)$ holds. Failing this inclusion, a counterexample σ is generated such that $\sigma \in L(S)$ whereas $\sigma \notin L(P)$. Another important area lies in runtime verification, wherein given a sequence of observations represented by $\sigma \in \Sigma^*$, we wish to check whether these observations satisfy the specification: $\sigma \in L(P)$. The verification community has considered numerous extensions to these basic ideas such as richer models of the system S that allow for succinct specifications (e.g., hierarchical state machines, state-charts), or go beyond finite state machines and include features such as real-time (timed automata) [4], physical quantities (hybrid automata) [3], and matching calls/returns [8, 23, 6]. The complexity of the language inclusion and membership problems in these settings are also well understood [11].

However, inclusion and membership problems lead to yes/no Boolean answers. The no answer for an inclusion problem is witnessed by a counterexample trace. However, the yes answer provides nothing further. A quantitative approach to these questions was proposed independently by Fainekos et al. [16], Donze et al. [14] and Rizk et al. [21] for the satisfaction of metric/signal temporal logic formula φ for a trace σ generated by continuous and hybrid systems. Therein, the authors use the euclidean metric over real-valued traces that defines a metric distance $d(\sigma,\sigma')$ between traces σ,σ' in order to check whether traces that are in the epsilon neighborhood of a given trace σ also satisfy the formula: $(\forall \sigma')\ d(\sigma,\sigma') < \epsilon \Rightarrow \sigma' \models \varphi$. Recent work, notably by Hasuo et al [24, 1] and Deshmukh et al [12] generalizes these notions to time domain as well as the signal data domain. Efficient algorithms for computing the robustness of a trace with respect to metric (signal) temporal formulas are known, and furthermore, the theory led to numerous approaches to finding falsifications of complex Simulink/Stateflow models, mining robust requirements and other monitoring problems [7].

Robustness Using Weighted Transducers. In this paper, we specify distances between finite words over Σ^*, using the notion of *cost functions*. A cost function assigns a non-negative rational cost to each pair of words $(w_1,w_2) \in \Sigma^* \times \Sigma^*$, modelling the cost of rewriting w_1 into w_2. By bounding the costs of rewritings, it models how words can be transformed. As a result, a neighborhood can be defined for each word, assuming that the cost of "rewriting" a word w back to itself is 0. This, in turn, allows to reason about robustness of languages. In order to model cost functions, we use *weighted transducers* with non-negative weights [15] along with an *aggregator* that combines the cost of each individual rewriting of the transducer into an overall cost between the input and output words. We now provide motivating examples for the cost functions that can be specified by such a model. A formal definition is provided in Section 2.

Motivating Example. Consider the transducer T of Figure 1. This transducer is over alphabet Σ : $\{a_1,a_2,b\}$. It allows to rewrite the letter a_1 into a_1 (at cost 0), and the letter a_2 into either a_2 (at cost 0) or b (at cost 1). Additionally, these rewritings are possible only at state q_1. This allows us to have a model wherein errors appear in bursts rather than individually: I.e, an error at a location increases the likelihood of one at the subsequent location. Thus, the transducer models all possible words w' that a given input word w can be rewritten into. As an example, the word $w : a_1a_2a_2a_2$ into $w' : a_1bba_2$ through transitions that rewrite the first two occurrences of a_2 into b. At the same time, the transducer forbids certain rewritings. For instance, the word w above cannot be rewritten into the word $w'' : bba_2a_1$ since the rewrite from an a_1 into a b or an a_2 into an a_1 is clearly disallowed by the transducer T in Figure 1.

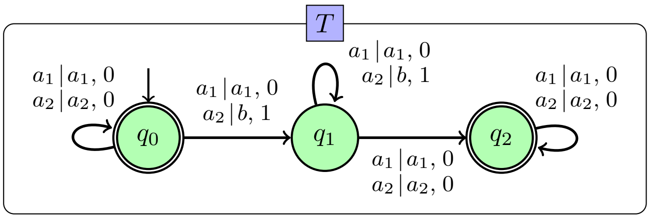

Figure 1 A weighted-transducer over $\Sigma = \{a_1,a_2,b\}$

While the transducer T specifies the cost for individual rewritings through its transitions, we define the cost of rewriting the entire word w into another w' by additionally specifying an *aggregator* function. For simplicity, we assume that there is exactly one run of the transducer that rewrites w into w'. The case of nondeterministic transducers is defined in Section 2.

1. *Discounted Sum (*`DSum`*):* Given a discount factor $\lambda \in \mathbb{Q} \cap (0,1)$, the cost of rewriting a word w into another word w' is defined as $\sum_{i=1}^{n} \lambda^{(i-1)} \tau_i$, wherein n is the size of a run through the transducer and τ_i is the cost associated with the i^{th} transition.
2. *Average (*`Mean`*):* This aggregator computes the mean cost: $\frac{1}{n}\sum_{i=1}^{n} \tau_i$ for $n > 0$.
3. *Sum (*`Sum`*):* This aggregator computes the sum: $\sum_{i=1}^{n} \tau_i$ for $n > 0$.

Returning to our example, the `Sum`-cost of rewriting $a_1a_2a_2a_2$ into a_1bba_2 is 2, for the `DSum`-cost with discount factor 1/2, it is 3/4, and for `Mean`-cost it is 1/2.

Our approach handles a more general nondeterministic transducer model that can allow for insertions of new letters, deletion of letters, transpositions and arbitrary substitutions of one letter by a finite word. Cost functions defined by such transducers may not satisfy the axioms of a metric, however many commonly encountered type of metrics between words such as the Cantor distance and the Levenstein (or edit) distance can be modeled as weighted transducers [13]. For example, edit distance is naturally modelled by a sum-transducer. Cantor distance maps any pair of word (w_1,w_2) of same length to 2^{-i} where i the first position where w_1 and w_2 differ, and to 0 if $w_1 = w_2$. This metric can be modelled by a discounted-sum transducer with discount factor 1/2.

Robustness problems. Given a cost function $c : (\Sigma^* \times \Sigma^*) \to \mathbb{Q}_{\geq 0}$ defined by a weighted-transducer with an aggregator function, we can define "neighborhoods" of languages for a given distance $\nu \geq 0$. For a regular language $N \subseteq \Sigma^*$ and a threshold $\nu \in \mathbb{Q}_{\geq 0}$, let us define its ν-neighborhood N_ν : $\{w' \in \Sigma^* \mid (\exists w \in N)\ c(w,w') \leq \nu\}$. Given a property $L \subseteq \Sigma^*$, we consider the following robustness problems:

- **Robust inclusion**: Given N,ν and L, check whether $N_\nu \subseteq L$.
- **Threshold synthesis**: Given N,L, find the largest threshold ν such that $N_\nu \subseteq L$.
- **Robust kernel synthesis**: given N,ν,L, find the largest $M \subseteq N$ s.t. $M_\nu \subseteq L$.

▶ **Example 1.** Consider the transducer of Figure 1 using the the `Sum` aggregator. We take L as the set of words which does not have *bbb* as a subword. Now, any word of the form $(a_1a_2)^*$ are ν-robust for any threshold ν since the letter a_1 is not rewritten by the transducer T. Such questions are tackled using the robust inclusion problem. On the other hand, let us choose a word $w \in a_2a_2a_2(a_1^*)$. It is ν-robust for all the thresholds $\nu \leq 2$ but not for $\nu \geq 3$. This is determined using the threshold synthesis problem. For all $\nu \geq 3$, the set of ν-robust words in $N = \Sigma^*$ is $(a_1 + a_1a_2 + a_1a_2a_2)^*$, and for $\nu \leq 2$, any word in Σ^* is ν-robust. Such questions are solved using the robust kernel synthesis problem.

Contributions. We show that the robust inclusion problem is solvable in PTIME when N and L are regular languages (given as NFA and DFA respectively) and the weighted-transducer defining the cost function is also given as input (Corollary 12). To obtain this result, we show that we can effectively compute in PTIME the largest threshold ν as defined before, thus solving the threshold synthesis problem (Theorem 11). This result holds for the three measures `Sum`, `DSum` and `Mean`. For `Sum`, we show that the robust kernel is effectively regular (Lemma 14) and testing its non-emptiness is PSPACE-complete (Theorem 15). For `Mean`, we show that the robust kernel is not regular in general (Lemma 16), and its non-emptiness is undecidable (Theorem 17). For `DSum`, we leave those questions partially open. We conjecture that the robust kernel is non-regular in general and provide a sufficient condition under which it is regular (Theorem 22).

Next, we present an implementation of the algorithms to synthesize robustness thresholds and report some experiments with our implementation, illustrating its application to analyzing manual control strategies under the presence of human error and approximate pattern analysis in type-1 diabetes data. Here we analyze a publicly available dataset of blood glucose values for people with type-1 diabetes. In both cases, we use a weighted transducer to model some of the specifics of human error and glucose sensor noise patterns. For the type-1 diabetes application, we use a robust pattern matching to detect behaviors that are clinically significant while accounting for the peculiarities of the glucose sensor.

Our work bears some similarities with earlier work by Henzinger et al [17, 22]. In these papers, notions of robustness for string to string transformations are studied and the notion of continuity of these transformations is defined. This is different from our setting, in which we use weighted transducers to define notions of distances, and these transducers are not necessarily continuous. Our notion of robustness is with respect to the rewriting of the words of one language and not about the transducers. The transducers themselves serve to define neighborhoods of strings.

2 Preliminaries and Problem Statements

Let Σ be an alphabet. We denote the empty word by the symbol $\varepsilon \notin \Sigma$ and we write Σ^* for the set of finite words over Σ. Let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. As usual, we write $\mathbb{Q}$ for the set of rationals, $\mathbb{N} = \{0,1,\ldots\}$ for naturals, and $\mathbb{N}^*$ for the words over the infinite alphabet $\mathbb{N}$.

A finite automaton over Σ is a tuple $A = (Q, Q_I, Q_F, \Delta)$ where Q is the finite set of states, $Q_I \subseteq Q$ is the set of initial states, $Q_F \subseteq Q$ is the set of final states and $\Delta \subseteq Q \times \Sigma \times Q$ is the set of transitions. A *run* r of A over a word $u = a_1 \ldots a_n \in \Sigma^*$ of length $n > 0$ is a sequence of transitions $t_1 \ldots t_n \in \Delta^*$ such that there exist $q_0, q_1, \ldots, q_n$ and for all $1 \leq i \leq n$, $t_i = (q_{i-1}, a_i, q_i)$. The run r is *simple* if no state repeats along r, i.e. $i \neq j$ implies that $q_i \neq q_j$ and, it is a cycle if $q_0 = q_n$. We say that r is a *simple cycle* if its a cycle and $t_2 \ldots t_n$ is simple. Also, r is *accepting* if it starts from an initial state $q_0 \in Q_I$ and ends into a final state $q_n \in Q_F$. We denote by $\mathrm{AccRun}_A(u)$ the set of accepting runs of A on the word u. The language defined by A is the set of words $L(A) = \{u \mid \mathrm{AccRun}_A(u) \neq \varnothing\}$. The automaton A is called *deterministic* (DFA for short) if Q_I is a singleton and Δ is a function from $Q \times \Sigma$ to Q. We define the representation size of an automaton $A = (Q, Q_I, Q_F, \Delta)$ as $|A| = |Q| + |\Delta|$.

Weighted transducers extend finite automata with string outputs and weights on transitions [15]. Any accepting run over some input word rewrites each input symbol into a (possibly empty) word, with some cost in $\mathbb{N}$. Transducers can also have ϵ-input transitions with non-empty outputs, such that output symbols can be produced even though nothing is read on the input (e.g. allowing for symbol insertions). The output of a run is the concatenation of all output words occurring on its transitions. Its cost is defined by an *aggregator function* $\mathsf{C} : \mathbb{N}^* \to \mathbb{Q}_{\geq 0}$, which associates a rational number to a sequence of non-negative integers.

We consider three different aggregator functions, given later. Since there are possibly several accepting runs over the same input, and generating the same output, we take the minimal cost of them to compute the value of a pair of input and output words.

▶ **Definition 2** (C-transducers). *Let $\mathsf{C} : \mathbb{N}^* \to \mathbb{Q}_{\geq 0}$ be an aggregator function. A C-transducer T is a tuple $(A, \mathtt{W})$ where $A = (Q, Q_I, Q_F, \Delta)$ is an NFA over $(\Sigma_\varepsilon \times \Sigma^*) \setminus \{(\varepsilon, \varepsilon)\}$ and the function $\mathtt{W} : \Delta \to \mathbb{N}$ associates weights to each transition.*

Given a transition $t = (q, a, v, q') \in Q \times \Sigma_\varepsilon \times \Sigma^* \times Q$, we write $\mathtt{Orig}(t) = q$, $\mathtt{In}(t) = a$, $\mathtt{Out}(t) = v$, and $\mathtt{Dest}(t) = q'$. We say that a transition $t \in \Delta$ can be triggered by T if it is in state $\mathtt{Orig}(t)$

and reads $\mathtt{In}(t)$ on its input (note that it is always possible to read $\mathtt{In}(t)=\varepsilon$). It, then, moves to $\mathtt{Dest}(t)$ and rewrites its input into $\mathtt{Out}(t)$. A run $r=t_1...t_n$ of T is a run of A. We write $\mathtt{In}(r)=\mathtt{In}(t_1)...\mathtt{In}(t_n)$ and $\mathtt{Out}(r)=\mathtt{Out}(t_1)...\mathtt{Out}(t_n)$ and say that r is a run of T on the pair of words $(\mathtt{In}(r),\mathtt{Out}(r))$. Let $(u_1,u_2)=(\mathtt{In}(r),\mathtt{Out}(r))$. If moreover r is accepting, we say that (u_1,u_2) is accepted by T, and denote by $\mathrm{AccRun}_T(u_1,u_2)$ the set of accepting runs over (u_1,u_2). We also say that u_1 is accepted by T if (u_1,u_2) is accepted by T for some $u_2\in\Sigma^*$. We denote the *weight sequence* of r by $\mathtt{W}(r)=\mathtt{W}(t_1)...\mathtt{W}(t_n)$ and its corresponding (aggregated) *cost* is $\mathsf{C}(r)=\mathsf{C}(\mathtt{W}(r))$.

A transducer T defines a relation from Σ^* to itself, called a *translation*, denoted R_T and defined by: $R_T=\{(u_1,u_2)\mid \mathrm{AccRun}_T(u_1,u_2)\neq\varnothing\}$. The *domain* of T, denoted $\mathrm{dom}(T)$ is the set of words u_1 for which there exists u_2 such that $(u_1,u_2)\in R_T$. The cost of a pair of words (u_1,u_2) is given by:

$$\mathsf{C}_T(u_1,u_2)=\begin{cases}+\infty & \text{if } (u_1,u_2)\notin R_T\\ \min\{\mathsf{C}(r)\mid r\in \mathrm{AccRun}_T(u_1,u_2)\} & \text{otherwise.}\end{cases}$$

Note that since runs consume at least one symbol of the input or one of the output, there are finitely many runs on a pair (u_1,u_2), hence the min is well-defined. Finally, given $\nu\in\mathbb{Q}$ and an input word $u_1\in\mathrm{dom}(T)$, we define the threshold output language $T_{\leq\nu}(u_1)$ of u_1 as: $T_{\leq\nu}(u_1)=\{u_2\mid \mathsf{C}_T(u_1,u_2)\leq\nu\}$. This notation extends naturally to languages $N\subseteq\Sigma^*$ by setting: $T_{\leq\nu}(N)=\bigcup_{u_1\in N\cap\mathrm{dom}(T)}T_{\leq\nu}(u_1)$.

▶ **Assumption 3.** *We restrict our attention to* C*-transducers* T *that satisfy the condition that for all* $u\in\mathrm{dom}(T)$*,* $\mathsf{C}_T(u,u)=0$ *(in particular* $(u,u)\in R_T$*). In other words, it is always possible to rewrite* u *into itself at zero cost.*

This assumption requires that each point must belong to any of its neighborhoods, which naturally comes from the indiscernibility axiom of distance. However, we do not require the triangle inequality axiom, that the edit distance does not satisfy.

Cost functions. We consider three aggregator functions, namely the sum, the mean and the discounted-sum. Let $\lambda\in\mathbb{Q}\cap(0,1)$ be a discount factor. Given a sequence of weights $\overline{\tau}=\tau_1...\tau_n$, those three functions are defined by:

$$\mathtt{Sum}(\overline{\tau})=\sum_{i=1}^{n}\tau_i \qquad \mathtt{Mean}(\overline{\tau})=\begin{cases}0 & \text{if } \overline{\tau}=\varepsilon\\ \frac{\mathtt{Sum}(\overline{\tau})}{n} & \text{otherwise}\end{cases} \qquad \mathtt{DSum}(\overline{\tau})=\sum_{i=1}^{n}\lambda^{(i-1)}\tau_i$$

Weighted-automata. When a C-transducer outputs only empty words, then its output component can be removed and we get what is called a C-*automaton*, which defines a function from words to costs. For $\mathsf{C}=\mathtt{Sum}$, this definition of $\mathtt{Sum}$-automaton coincides with the classical notion weighted automata over the semiring $(\mathbb{N}\cup\{+\infty\},\min,+)$ from [15].

Robustness problems. We study the following three fundamental problems related to robustness for three different aggregator functions $\mathsf{C}\in\{\mathtt{Sum},\mathtt{Mean},\mathtt{DSum}\}$. Given a threshold $\nu\in\mathbb{Q}$, a C-transducer T and a regular language L, a word $u\in\mathrm{dom}(T)$ is said to be ν-*robust* (or just robust if ν is clear from the context) if $T_{\leq\nu}(u)\subseteq L$. In other words, all its rewritings of cost ν at most are in L. A language $N\subseteq\Sigma^*$ is said to be ν-robust if $N\cap\mathrm{dom}(T)$ contains only ν-robust words. Finally, the ν-*robust kernel* of T is the set $\mathtt{Rob}_T(\nu,L)$ of ν-robust words: $\mathtt{Rob}_T(\nu,L)=\{u\in\mathrm{dom}(T)\mid T_{\leq\nu}(u)\subseteq L\}$. We prove that as the error threshold grows, so does the robust kernel.

▶ **Proposition 4.** *Given $\nu,\nu' \in \mathbb{Q}_{>0}$, a* C*-transducer T and a regular language L, we have that $\nu' \leq \nu \Longrightarrow \mathtt{Rob}_T(\nu',L) \subseteq \mathtt{Rob}_T(\nu,L)$.*

Proof. By definition $T_{\leq\nu}(u_1) = \{u_2 \mid \mathsf{C}_T(u_1,u_2) \leq \nu\}$. For all $u_1 \in \mathrm{dom}(T)$ we have that $u_1 \in \mathtt{Rob}_T(\nu,L)$ iff for all u_2 both $u_2 \in L$ and $\mathsf{C}_T(u_1,u_2) \leq \nu$ hold. Clearly $u_1 \in \mathtt{Rob}_T(\nu,L)$ implies $u_1 \in \mathtt{Rob}_T(\nu',L)$ for any $\nu' \leq \nu$. ◀

We are in a position to formally define the three key problems studied in this paper. For these definitions, we let $\mathsf{C} \in \{\mathtt{Sum},\mathtt{Mean},\mathtt{DSum}\}$.

▶ **Problem 5** (Robust Inclusion)**.** *Given a* C*-transducer T, a regular language $N \subseteq \Sigma^*$ as an* NFA*, a threshold $\nu \in \mathbb{Q}_{\geq 0}$ and a language $L \subseteq \Sigma^*$ as a* DFA*, the* robust inclusion problem *is to decide whether $N \subseteq \mathtt{Rob}_T(\nu,L)$, i.e. whether $T_{\leq\nu}(N) \subseteq L$.*

Note that we consider our specification language L deterministically presented, for tractability.

▶ **Problem 6** (Threshold Synthesis)**.** *Given a* C*-transducer T, a regular language $N \subseteq \Sigma^*$ as an* NFA*, and a regular language $L \subseteq \Sigma^*$ as a* DFA*, the* threshold synthesis problem *is to output a partition of the set of thresholds $\mathbb{Q}_{\geq 0} = G \uplus B$ into sets G and B of* good *and bad thresholds, i.e.*

$$G = \{\nu \in \mathbb{Q}_{\geq 0} \mid N \subseteq \mathtt{Rob}_T(\nu,L)\} \text{ and } B = \{\nu \in \mathbb{Q}_{\geq 0} \mid N \not\subseteq \mathtt{Rob}_T(\nu,L)\}.$$

As direct consequence of Proposition 4, the sets G and B are intervals of values, that is for all $\nu_1,\nu_2 \in \mathbb{Q}_{\geq 0}$, if $\nu_1 < \nu_2$ and $\nu_2 \in G$, then $\nu_1 \in G$, and if $\nu_1 \in B$ then $\nu_2 \in B$.

▶ **Problem 7** (Robust Kernel Non-emptiness)**.** *Given a* C*-transducer T, a regular language $L \subseteq \Sigma^*$ as a* DFA*, a threshold $\nu \in \mathbb{Q}_{\geq 0}$, the* robust kernel non-emptiness *problem is to decide if* there exists $u \in \mathtt{Rob}_T(\nu,L)$.

For the cases where we provide algorithms for solving the non-emptiness of the robust kernel, we also succeed in synthesizing the robust kernel as an automaton.

3 Robust Verification

Given an instance of the threshold synthesis problem, we show how to compute the interval of good thresholds G and the interval of bad thresholds B in PTIME for all the three measures we consider. As a corollary, we show that the robust inclusion problem for $\mathtt{Sum}$,$\mathtt{Mean}$,$\mathtt{DSum}$ measures is in PTIME.

In the following, we assume that $N = \mathrm{dom}(T)$. This is w.l.o.g. as transducers are closed (in polynomial time) under regular domain restriction (using a product construction of T with the automaton for N). With this assumption, the set of good thresholds G becomes $G = \{\nu \in \mathbb{Q}_{\geq 0} \mid \mathrm{dom}(T) \subseteq \mathtt{Rob}_T(\nu,L)\}$ and dually for the set of bad thresholds B. We let $\nu_{T,L}$ be the infimum of the set of bad thresholds, i.e. $\nu_{T,L} = \inf B = \inf\{\nu \in \mathbb{Q}_{\geq 0} \mid \mathrm{dom}(T) \not\subseteq \mathtt{Rob}_T(\nu,L)\}$. As illustrated by the following example, computing $\nu_{T,L}$ allows us to compute $G = [0,v_{T,L}]$ and $B = [v_{T,L},+\infty)$.

▶ **Example 8.** Let $\Sigma = \{a,b,c\}$ and $\mathsf{C} \in \{\mathtt{Mean},\mathtt{DSum}\}$. Consider the best threshold problem for T the C-transducer of Figure 2, $N = \mathrm{dom}(T) = a^*$ and $L = a^* + b^*$. Note that the translations accepted by OK and ID belong to L. On the contrary, translations accepted by KO do not belong to L and so they are not robust w.r.t. L for any threshold. For $\mathtt{Mean}$ measure, the cost of a translation into c^* is exactly 1 while the one into b^* range over $[0,1)$. Hence $\nu^{\mathtt{Mean}}_{T,L} = 1$ and the set partition of good and bad thresholds is $G^{\mathtt{Mean}} = [0,1)$ and $B^{\mathtt{Mean}} = [1,+\infty)$. In the case of

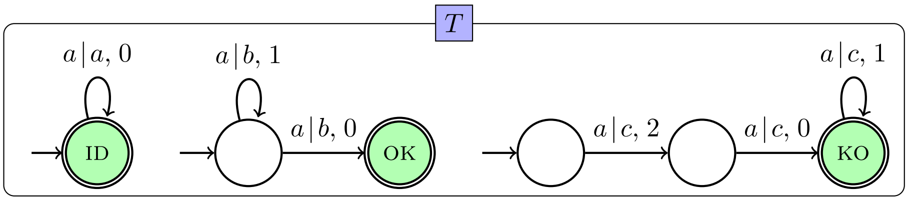

Figure 2 Transducer T for which the infimums $\nu_{T,L}^{\texttt{Mean}} = 1$ and $\nu_{T,L}^{\texttt{DSum}} = 2$ are bad thresholds for T interpreted as `Mean`- and `DSum`-transducer with discount factor $\frac{1}{2}$ respectively, and for $L = a^* + b^*$.

`DSum` with discount factor 0.5, the cost of a translation into c^* range over [2,2.5) while the one into b^* range over [0,2). So $\nu_{T,L}^{\texttt{DSum}} = 2$ and the thresholds are partitioned by $G^{\texttt{DSum}} = [0,2)$ and $B^{\texttt{DSum}} = [2,+\infty)$.

Then, we associate with every transducer T and property L given by some DFA A (assumed to be complete), a graph called the *weighted-graph associated with T and A*, and denoted by $G_{T,A}$. Intuitively, $G_{T,A}$ is obtained by first taking the synchronised product of T and A (where A is simulated on the outputs of T) and then by projecting this product on the inputs.

Formally, given $T = (Q, Q_I, Q_F, \Delta, \texttt{W})$ and $A = (P, p_I, P_F, \delta)$, the synchronised product $G_{T,A} = (V, E, \texttt{W}' : E \to \mathbb{N})$ is such that:

- $V = Q \times P$
- E is the set of edges $e = (q,p) \to (q',p')$ such that there exists $a \in \Sigma_\varepsilon$ and a transition $t = (q,a,u,q') \in \Delta$ such that $p' = \delta(p,u)$ where δ has been extended to words in the expected way. We say that e is compatible with t.
- For all $e \in E$, $\texttt{W}'(e) = \min\{\texttt{W}(t) \mid e \text{ is compatible with some } t \in \Delta\}$.

Additionally, we note $V_I = Q_I \times \{p_I\}$ the set of *initial vertices* and $V_F = Q_F \times (P \setminus P_F)$ the set of *final vertices* of this graph. Given a path π in this graph as a sequence of edges $e_1 \ldots e_n$, we let $\mathsf{C}(\pi) = \mathsf{C}(\texttt{W}'(e_1) \ldots \texttt{W}'(e_n))$.

The following lemma establishes some connection between $\nu_{T,L}$ and the paths of $G_{T,A}$.

▶ **Lemma 9.** *The infimum cost of paths from a vertex in V_I to a vertex in V_F is equal to $\nu_{T,L}$, i.e. $\nu_{T,L} = \inf\{\mathsf{C}(\pi) \mid \exists s_0 \in V_I\ \exists s_f \in V_F\ s_0 \xrightarrow{\pi}_{G_{T,A}} s_f\}$.*

Proof. We first show that any path π from V_I to V_F satisfies $\mathsf{C}(\pi) \geq \nu_{T,L}$. Take such a path. By construction of $G_{T,A}$, there exists an input word $u_1 \in \mathrm{dom}(T)$, some output word $u_2 \notin L$ and an accepting run r of T on (u_1,u_2) of value $\mathsf{C}(r) = \mathsf{C}(\pi)$. Since the value $\mathsf{C}_T(u_1,u_2)$ is the minimal value of all accepting runs of T over (u_1,u_2), we have $\mathsf{C}(r) \geq \mathsf{C}_T(u_1,u_2)$ and u_1 is not robust for threshold $\mathsf{C}_T(u_1,u_2)$, *a fortiori* for threshold $\mathsf{C}(r)$, from which we get $\mathsf{C}(r) = \mathsf{C}(\pi) \geq \nu_{T,L}$. This shows that $\nu_{T,L} \leq \inf\{\mathsf{C}(\pi) \mid \exists s_0 \in V_I \exists s_f \in V_F\ s_0 \xrightarrow{\pi}_{G_{T,A}} s_f\}$.

Suppose that $\nu_{T,L}$ is strictly smaller than this infinimum (that we denote m) and take some rational number ν such that $\nu_{T,L} < \nu < m$. Since $\nu_{T,L} < \nu$, it is a bad threshold which means that there exists $u_1 \in \mathrm{dom}(T)$ such that $u_1 \notin \texttt{Rob}_T(\nu, L)$. Hence there exists $u_2 \notin L$ such that $\mathsf{C}_T(u_1,u_2) \leq \nu$, and by definition of $G_{T,A}$, there exists a path π from V_I to V_F of value $\mathsf{C}(\pi) \leq \nu$. This contradicts the fact that $\nu < m$ by definition of m. Hence, $\nu_{T,L} = m$, concluding the proof. ◀

The next lemma establishes that the infimum of values of paths between two sets of states in a weighted graph can be computed in PTime and it is also decidable in PTime if the infimum is realized by a path, for all the three measures considered in this paper. As a direct corollary of this lemma we obtain the main theorem of the section. The full proof can be found in Appendix.

▶ **Lemma 10.** *For a weighted graph $G=(V,E,\mathtt{W}:E\to\mathbb{Q}_{\geq 0})$, a set of sources $V_I\subseteq V$ and a set of targets $V_F\subseteq V$, the infimum of the weights of paths from V_I to V_F can be computed in* PTIME *for all* $\mathsf{C}\in\{\mathtt{Sum},\mathtt{DSum},\mathtt{Mean}\}$. *Moreover, we can decide in* PTIME *if this infimum is realizable by a path.*

Sketch of proof. First, if no state of V_F are reachable from some state of V_I, we have $\nu_{T,L}=+\infty$. Otherwise we use different procedures, depending on the aggregator C.

For `Sum`, the infimum can be computed in PTIME using Dijkstra algorithm and it is always feasible. For `Mean`, we first note that the infimum is the `Mean` value of either a simple path or the value of a reachable cycle that can be iterated before moving to some target. In the latter case, the infimum is not feasible but can be approximated as close as possible by iterating the cycle. So, the infimum is feasible iff it is the `Mean` value of a simple path. The minimal `Mean` values amongst simple paths and cycles can be computed in PTIME with dynamic programming thanks to [18]. For `DSum`, Theorem 1 of [5] provides a PTIME algorithm that computes for all $v\in V$, the infimum of `DSum` values x_v of paths reaching the target V_F from v. ◀

▶ **Theorem 11.** *For a given* C*-transducer T, a language $N\subseteq\Sigma^*$ given as an NFA and $L\subseteq\Sigma^*$ given as a DFA, the set partition of good and bad thresholds (G,B) for* $\mathsf{C}\in\{\mathtt{Sum},\mathtt{DSum},\mathtt{Mean}\}$ *can be computed in* PTIME.

Proof. First, we restrict the domain of T to N by taking the product of T and the automaton for N (simulated over the input of T). Then, according to Lemma 10, we can compute in PTIME the value $\nu_{T,L}$. This value is the infimum of B. If this infimum is feasible then the interval B is left closed and equal to $[\nu_{T,L},+\infty)$ while $G=[0,\nu_{T,L})$, and on the contrary, if this infimum is not feasible, then B is left open and equal to $(\nu_{T,L},+\infty)$, while $G=[0,\nu_{T,L}]$. Note that when $\nu_{T,L}=0$ and is feasible, then $G=[0,0)=\varnothing$. ◀

As a direct consequence, the robust inclusion problem for a threshold ν can be solved by checking if $\nu\in G$, and so we have the following corollary.

▶ **Corollary 12.** *Let* $\mathsf{C}\in\{\mathtt{Sum},\mathtt{DSum},\mathtt{Mean}\}$. *Given T a* C*-transducer, $N\subseteq\Sigma^*$ given as an NFA, $L\subseteq\Sigma^*$ given as a DFA and $\nu\in\mathbb{Q}$. The language inclusion $N\subseteq\mathtt{Rob}_T(\nu,L)$ can be decided in* PTIME.

4 Robust Kernel Synthesis

In this section, we show that the robust kernel is regular for `Sum`-transducers, and checking its emptiness is PSPACE-complete. For `Mean`, we show that it is not necessarily regular, and checking its emptiness is undecidable. For `DSum`, we conjecture that the robust kernel is non-regular and give sufficient condition under which it is regular and computable, implying decidability of its emptiness.

4.1 Sum measure

To show robust kernel regularity, we rely on the construction of Theorem 2 of [2] in the context of weighted automata over the semiring $(\mathbb{N}\cup\{+\infty\},\min,+)$. The following lemma, use the same automata construction and provides an upper bound on the number of states required to denote a threshold language with a DFA.

▶ **Lemma 13.** *Let U be an n state* `Sum`*-automaton and $\nu\in\mathbb{N}$. The threshold language $L_\nu(U)=\{w\mid U(w)\geq\nu\}$, where $U(w)$ is defined as $+\infty$ if there is no accepting run on w,* *otherwise as the minimal sum of the weights along accepting runs on w, is regular. Moreover $L_\nu(U)$ is recognized by a DFA with $O\big((\nu+2)^n\big)$ states.*

Proof. First, let assume that U has universal domain (i.e. any word has some accepting run), otherwise we complete it by assigning value ν to each word of its complement.

Then, $U(w) \geq \nu$ iff all the accepting runs on w have value at least ν. We design a DFA D that accepts exactly those words. Since the weights of U are non-negative, D just has to monitor the sum of all runs up to ν, by counting in its states. If Q is the set of states of U, the set of states of D is $2^{Q\times\{0,\dots,\nu-1,\nu^+\}}$, where ν^+ intuitively means any value $\geq\nu$. We extend natural addition to $X=\{0,\dots,\nu-1,\nu^+\}$ by letting $a+b=\nu^+$ iff $a=\nu^+$, or $b=\nu^+$, or $a+b\geq\nu$. Then, D is obtained by subset construction: there is a transition $P\xrightarrow{\sigma}P'$ in D iff $P'=\{(q',i+j)\mid(q,i)\in P\wedge q\xrightarrow{\sigma|j}_U q'\}$. A state P is accepting if $P\cap\big((Q\setminus F)\times\{0,\dots,\nu-1\}\big)=\varnothing$, where F are the accepting states of U.

Though simple, the latter construction does not give the claimed complexity, as the number of states of D is $2^{n\nu}$. But the following simple observation allows us to get a better state complexity. Consider an input word of the form uv. If after reading u, D reaches some state P such that for some state q, there exists $(q,i),(q,j)\in P$ such that $i<j$, then if there is an accepting run of U from q on v, with sum s, there is an accepting run on uv with sum $i+s$ and one with sum $j+s$. Therefore if $i+s\geq\nu$, then $j+s\geq\nu$ and the pair (q,j) is useless in P. So, we can keep only the minimal elements in the states of D, where minimality is defined with respect to the partial order $(q,i)\preceq(p,j)$ if $q=p$ and $i\leq j$. Let us call D_{opt} the resulting "optimised" DFA. Its states can be therefore seen as functions from Q to $\{0,\dots,\nu-1,\nu^+\}$, so that we get the claimed state-complexity. ◀

▶ **Lemma 14** (Robust language regularity). *Let T be a* $\texttt{Sum}$*-transducer, $\nu\in\mathbb{N}$ and L be regular language. The language of* robust words $\texttt{Rob}_T(\nu,L)$ *is a regular language. Moreover, if L is given by a DFA with n_L states and T has n_T states, then* $\texttt{Rob}_T(\nu,L)$ *is recognisable by a DFA with $O\big((\nu+2)^{n_T\times n_L}\big)$ states.*

Proof of this lemma is provided in the appendix.

▶ **Theorem 15.** *Let T be a* $\texttt{Sum}$*-transducer, $\nu\in\mathbb{N}$ given in binary and L a regular language given as a DFA. Then, it is* PSPACE*-complete to decide whether there exists a robust word $w\in\texttt{Rob}_T(\nu,L)$. The hardness holds even if ν is a fixed constant, T is letter-to-letter*[1] *and io-unambiguous*[2]*, and its weights are fixed constants in $\{0,1\}$.*

Proof. From Lemma 14, $\texttt{Rob}_T(\nu,L)$ is recognisable by a DFA with $O\big((\nu+2)^{n_T\times n_L}\big)$ states, where n_T is the number of states of T and n_A the number of states of the DFA defining L. Checking emptiness of this automaton can be done in PSPACE (apply the standard NLOGSPACE emptiness checking algorithm on an exponential automaton that needs not be constructed explicitly, but whose transitions can be computed on-demand).

To show PSPACE-hardness, we reduce the problem from [19] of checking the non-emptiness of the intersection of n regular languages given by n DFA $A_1,\dots,A_n$, over some alphabet Γ. In particular, we construct T, ν and a DFA A such that $\bigcap_i L(A_i)\neq\varnothing$ iff there exists a robust word with respect to T,ν and L.

We define the alphabet as $\Sigma=\Gamma\cup\{\#_1,\dots,\#_n,\dashv\}$ where we assume that $\#_1,\dots,\#_n,\dashv\notin\Gamma$, and construct a transducer T which reads a word $w\dashv$ of length $k=|w|+1$ with $w\in\Gamma^*$, and

[1] A transducer is letter-to-letter if $\Delta\subseteq Q\times\Sigma\times\Sigma\times Q$.
[2] For all word pairs (w_1,w_2), there exists at most one run of T on w_1 outputting w_2.

rewrites it into either itself, or $(\#_i)^k$ for all $i \in \{1,...,n\}$. The identity rewriting has total weight 0 while the rewriting into $\#_i^k$ has total weight 1 if $w \in L(A_i)$, and 0 otherwise. The transducer T is constructed as the disjoint union of $n+1$ transducers $T_1,...,T_n,T_\dashv$. For all $i \in \{1,...,n\}$, T_i simulates A_i on the input and outputs $\#_i$ whenever it reads an input letter different from $\dashv$, with weight 0. When reading $\dashv$ from an accepting state of A_i, it outputs $\dashv$ with weight 1, and if it reads $\dashv$ from a non-accepting state, it outputs $\dashv$ with weight 0. Finally, $T_\dashv$ just realizes the identify function with weight 0. Note that T has polynomial size in $A_1,...,A_n$ and it is letter-to-letter and (input,output)-deterministic.

Now we prove that a word $w\dashv$ is robust iff $w \in \bigcap_i L(A_i)$. Assume that there exists a robust word $w\dashv$ for the property $L = (\Gamma \cup \{\dashv\})^*$ and threshold $\nu = 0$. Equivalently, it means that for all rewritings $\alpha \in \Sigma^*$, if $\texttt{Sum}_T(w\dashv,\alpha) \leq 0$ then $\alpha \in L$. It is equivalent to say that all its rewritings α satisfies either $\texttt{Sum}_T(w\dashv,\alpha) \geq 1$ or $\alpha \in L$. By definition of T, it is equivalent to say that all rewritings α are such that either $\alpha \in (\#_i)^* \cdot \dashv$ for some i and $w \in L(A_i)$, or $\alpha = w\dashv$. Since T necessarily rewrites $w\dashv$ into $w\dashv$, as well as into $(\#_1)^k,...,(\#_n)^k$, where $k = |w|+1$, the latter assumption is equivalent to saying that $w \in L(A_i)$ for all $i \in \{1,...,n\}$, concluding the proof. ◀

4.2 Mean measure

Let us first establish non-regularity of the robust kernel.

▶ **Lemma 16.** *Given a regular language L, a* $\texttt{Mean}$*-transducer T and $\nu \in \mathbb{Q}_{\geq 0}$, the language* $\texttt{Rob}_T(\nu,L)$ *is not necessarily regular, but recursive.*

Proof. Consider the language $L = \{w \mid \exists i \in \mathbb{N} : w(i) = a\}$ on the alphabet $\Sigma = \{a,b\}$, i.e. the set of words on Σ that contain at least one a. Now, consider a (one state) transducer T that can non-deterministically copy letters or change the current letter from a to b with weight one. Now, if we fix ν to be equal to $\frac{1}{2}$, then all the translations of w by T of cost less than $\frac{1}{2}$ are included in L, i.e. each translation of w will contain at least one letter a, if and only if, the number of a's in w is larger than the number of b's in w, i.e. $\texttt{Rob}_T(\frac{1}{2},L) = \{w \mid w_{\sharp a} > w_{\sharp b}\}$, which is not regular. Note that in general $\texttt{Rob}_T(\nu,L)$ is recursive because the membership problem to it, is decidable by Corollary 12 (applied on a singleton language). ◀

We now show that testing the non-emptiness of the robust kernel is undecidable.

▶ **Theorem 17.** *Let L be a regular language, T be a* $\texttt{Mean}$*-transducer and $\nu \in \mathbb{Q}_{\geq 0}$. Determine whether* $\texttt{Rob}_T(\nu,L) \neq \varnothing$ *is undecidable. It holds even if T is io-unambiguous.*

Proof. Let A be a $\texttt{Sum}$-automaton weight by integers. The proof goes by reduction from determining whether all words admits a run of non-positive cost in A which is known to be undecidable [10, 2]. From A, we construct L as the set of *non* accepting runs of A union Σ^*, the threshold ν as the maximal absolute weight of A and T such that:

$$\texttt{Mean}_T = \bigcup \begin{array}{l} \{(w,w) \mapsto 0 \mid w \in \Sigma^*\} \\ \{(w,r_w) \mapsto X_{r_w} + \nu|w| \mid r_w \text{ run of } A \text{ over } w \in \Sigma^* \text{ with value } X_{r_w}\} \end{array}$$

We can construct T as the disjoint union between a single-state transducer with weights zero realising the identity, and a transducer that outputs all the possible runs of A on its input, such that each T-transition simulating an A-transition t of value x (in A) has value $\nu + x$, which is positive by definition of ν. Hence T is indeed weighted over non-negative numbers. Note that T is io-unambiguous: if the input and output are fixed, there is at most one run of T. Now, we show that $\texttt{Rob}_T(\nu,L) = \varnothing$ iff $\forall w \cdot A(w) \leq 0$, i.e.

$$\forall w_1 \exists w_2 \in \overline{L}\ \texttt{Mean}_T(w_1,w_2) \leq \nu \text{ iff } \forall w\ A(w) \leq 0.$$

We have the following equivalences: $\forall w_1 \exists w_2 \in \overline{L} \cdot \mathtt{Mean}_T(w_1,w_2) \leq \nu$ iff for all w_1, there exists an accepting run r of A on w_1 such that $\mathtt{Mean}_T(w_1,r) \leq \nu$, i.e. $\mathtt{Sum}_T(w_1,r) \leq \nu|w_1|$ and by definition of T, it is equivalent to asking that $\mathtt{Sum}_A(r)+\nu|w_1| \leq \nu|w_1|$, i.e. $\mathtt{Sum}_A(r) \leq 0$. Hence, the latter statement is equivalent to the fact that for all words w_1, there exists an accepting run of A of value ≤ 0. Since A takes the minimal value of all accepting runs to compute the value of a word, it is equivalent to saying that for all w_1, $A(w_1) \leq 0$, i.e., A is universal, concluding the proof. ◀

4.3 Discounted sum measure

For DSum-transducer, we conjecture that $\mathtt{Rob}_T(\nu,L)$ is in general non-regular. This claim is substantiated by the fact that DSum-automata over $\mathbb{Q}$ and ω-words have in general non-regular cut-point languages, i.e. the set of words of DSum value below a given threshold is in general non-regular [9]. With a proof similar to that of Theorem 17 for Mean-transducers, it is possible to show that the universality problem for DSum-automata, which is open to the best of our knowledge, reduces to checking the emptiness of the robust language of a DSum-transducer.

Following an approach that originates from the theory of probabilistic automata, it is has been shown that cut-point languages are regular when the threshold is ϵ-isolated [9]. Formally, a threshold $\nu \in \mathbb{Q}$ is ϵ-isolated, for $\epsilon > 0$ and for some DSum-transducer T if, for all accepting runs r of T, $\mathtt{DSum}_T(r) \in [0,\nu-\epsilon] \cup [\nu+\epsilon,+\infty)$. It is *isolated* if it is ϵ-isolated for some ϵ. Our objective now is to show that when ν is isolated, then $\mathtt{Rob}_T(\nu,L)$ is regular and one can effectively construct an automaton recognizing it. We will also give a (possibly non-terminating) algorithm which, when it terminates, returns an automaton recognising $\mathtt{Rob}_T(\nu,L)$, and which is guaranteed to terminate whenever ν is ϵ-isolated for some ϵ. Towards these results, we first give intermediate useful results. For a state q of T, we call *continuation* of q any run from q leading to some accepting state of T. By extension, we also call continuation of a run r any continuation of the last state of r. A transducer T is said to be *trim* if all its states admits some continuation. Note that any transducer can be transformed into an equivalent trim one in PTime, just by removing states that do not admit any continuation (this can be tested in PTime).

▶ **Lemma 18.** *Let T be a trim* DSum*-transducer and $\nu \in \mathbb{Q}$. If ν is ϵ-isolated for some ϵ, then there exists $n^* \in \mathbb{N}$ such that any run r of length at least n^* satisfies one of the following properties:*

1. $\mathtt{DSum}(r) \leq \nu-\epsilon$ *and any continuation r' of r satisfies* $\mathtt{DSum}(rr') \leq \nu-\epsilon$
2. $\mathtt{DSum}(r) \geq \nu+\epsilon/2$ *and any continuation r' of r satisfies* $\mathtt{DSum}(rr') \geq \nu+\epsilon$.

Proof of this lemma is provided in the appendix.

We now show how to construct better and better regular under-approximations of the set of *non*-robust words, show that they "finitely" converge to the set of non-robust words when ν is isolated.

▶ **Lemma 19.** *Let T be a* DSum*-transducer, $\nu \in \mathbb{Q}$ and L a regular language (given as a DFA). For all n, we can construct an NFA A_n such that:*

1. $L(A_n) \subseteq L(A_{n+1})$
2. $L(A_n) \subseteq \overline{\mathtt{Rob}_T(\nu,L)} \cap \mathrm{dom}(T)$

Moreover, if ν is isolated, there exists n^* such that $L(A_{n^*}) = \overline{\mathtt{Rob}_T(\nu,L)} \cap \mathrm{dom}(T)$.

Proof of this lemma is provided in the appendix.

We also show that one can test whether given n, we have $\overline{\texttt{Rob}_T(\nu,L)}\cap\mathrm{dom}(T)\subseteq L(A_n)$, as stated by the following lemma:

▶ **Lemma 20.** *Given a regular language N (given as some NFA), it is decidable to check whether $\overline{\texttt{Rob}_T(\nu,L)}\cap\mathrm{dom}(T)\subseteq N$ holds.*

Proof. We take the synchronised product of T, $\overline{L}$ (on the output) and $\overline{N}$ (on the input), project the output, and check whether a path from an initial to a final vertex exists with discounted sum $\leq\nu$. ◀

Those results allow us to define the following semi-algorithm:

1. **Compute-Rob**(T,ν,L)
2. **for** n **from** 1 **to** $+\infty$
3.   **compute** A_n // as in Lemma 19
4.   **if** $\overline{\texttt{Rob}_T(\nu,L)}\cap\mathrm{dom}(T)\subseteq L(A_n)$ **return** A_n // using Lemma 20

▶ **Lemma 21.** *The algorithm **Compute-Rob**(T,ν,L) satisfies the following properties:*

1. *if it terminates, then it returns an automaton recognising $\overline{\texttt{Rob}_T(\nu,L)}\cap\mathrm{dom}(T)$,*
2. *if ν is isolated, it terminates.*

Proof. If it terminates at steps n, then by Lemma 19 and the test at line 4 we know that $L(A_n)=\overline{\texttt{Rob}_T(\nu,L)}\cap\mathrm{dom}(T)$, and if ν is isolated, the test will eventually succeed. ◀

Note that the algorithm may terminate even if ν is not isolated. It is the case for instance when the threshold is ϵ-isolated for "long" runs only, but not necessarily for small runs, in the sense that it is only required that for some n, any accepting runs of length at least n satisfies either $\texttt{DSum}(r)\leq\nu-\epsilon$ or $\texttt{DSum}(r)\geq\nu+\epsilon$. As a corollary of Lemma 21, $\texttt{Rob}_T(\nu,L)$ is regular when ν is isolated: it suffices to run Algorithm **Compute-Rob**, complement the automaton and restrict its language to $\mathrm{dom}(T)$.

▶ **Theorem 22.** *Let T be a $\texttt{DSum}$-transducer and $\nu\in\mathbb{Q}$ and L a regular language. If ν is isolated, then $\texttt{Rob}_T(\nu,L)$ is regular.*

5 Implementation and Case Study

We describe an evaluation of the ideas presented thus far and their application to two case studies: one involving robustness of control strategies to human mistakes and the other involving glucose values for patients with type-1 diabetes. We have implemented in Python the threshold synthesis problem (Problem 6) for the discounted and average costs. Our implementation supports the specification of a language L specified as an NFA, a weighted transducer T and a property P specified as some DFA. The implementation is available upon request.

5.1 Robustness of Human Control Strategies

An industrial motor operates under many gears $g_1,\ldots,g_5$. Under fault, the human operator must take control of the machine and achieve the following: *If the system goes into a fault the operator must ensure that (a) the system is immediately set in gears $3-5$. Subsequently, for the next 5 cycles: (b) it must never go to gear g_1 or g_2; and (c) must shift and stay at a higher gear g_4 or g_5 after the 5^{th} cycle until the fault is resolved.*

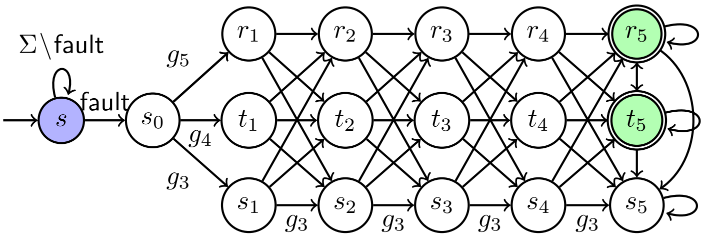

Figure 3 Finite state automaton P showing a desired property for the automatic transmission system. All incoming edges to $s_1,\ldots,s_5$ have label g_3, incoming edges to $t_1,\ldots,t_5$ have label g_4 and $r_1,\ldots,r_5$ have incoming edges labelled g_5. All edges not shown lead to a rejecting sink state.

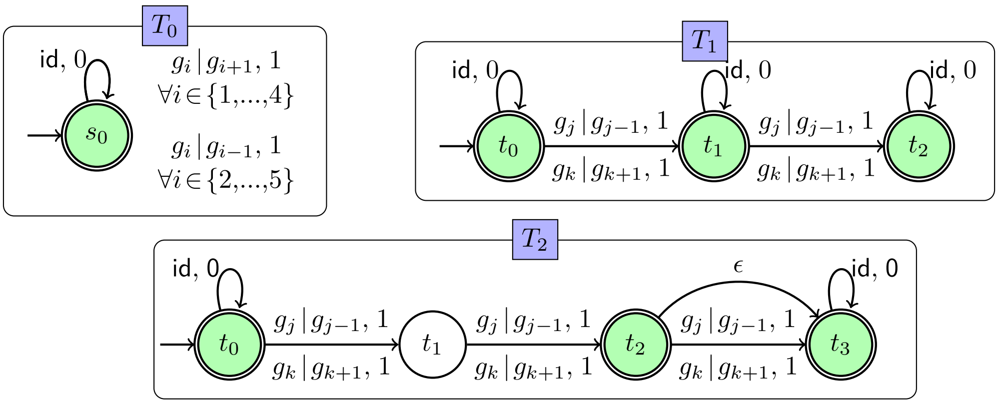

Figure 4 Transducers modeling potential human operator mistakes along with their costs: T_0 allows arbitrarily many mistakes whereas T_1 restricts the number of mistakes to at most 2, whereas T_2 models a "bursty" set of mistakes. The edge $a\,|\,b,w$ denotes a replacement of the letter a by b with a cost w. For convenience T_2 uses an ϵ transition that can be removed.

Figure 3 shows a finite state machine P that accepts all words satisfying this property: fault is not in the operator's control but $g_1,\ldots,g_5$ are operator actions. Consider that the operator can perform this task in two different ways: $\sigma_1:$ fault $g_4\ g_4\ g_4\ g_5\ g_5$ versus $\sigma_2:$ fault $g_3\ g_3\ g_3\ g_3\ g_4$. The input σ_1 induces the run $s, s_0, t_1, t_2, t_3, r_4, r_5$ whereas the input σ_2 induces the run $s,s_0,s_1,s_2,s_3,s_4,t_5$. Both σ_1,σ_2 satisfy the property of interest and as such there is nothing to choose one over the other. Suppose the human operator can make mistakes, especially since they are under stress. We will consider that the operator can substitute a command for gear g_i with g_{i-1} (for $i>1$) or g_{i+1} (for $i<5$). We use a weighted transducer T_0 shown in Figure 4 to model these substitutions. The transducer defines possible ways in which a string σ can be converted to σ' with a notion of cost for the conversion. In this example we consider two notions of cost: the `DSum`-cost, and the `Mean`-cost. These costs now allow us to compare σ_1 versus σ_2. For instance, under both notions we will discover that σ_1 is much more robust than σ_2. The robustness of σ_1 under both cost models is ∞ since any change to σ_1 under the transducer continues to satisfy the desired property. On the other hand σ_2 has a finite robustness, since operator mistakes can cause violations.

The use of a transducer allows for a richer specification of errors. For instance, transducer T_2 in Fig. 4 shows a model of "bounded" number of mistakes that assume that the operator makes at most 2 mistakes whereas T_3 in Fig. 4 shows a model with "bursty" mistakes that assume that mistakes occur in bursts of at least 2 but at most 3 mistakes at a time. These models are useful in capturing fine grained assumptions about errors that are often the case in

Table 1 Running times and robustness values computed for various input strings (the first letter **fault** is common to all the strings and is omitted). All timings are measured in seconds, ϵ denotes time <0.01 seconds.

String	T_0			T_1			T_2		
	Disc.	Avg.	Time	Disc.	Avg.	Time	Disc.	Avg.	Time
$g_4g_4g_4g_4g_5g_5$	∞	∞	ϵ	∞	∞	ϵ	∞	∞	ϵ
$g_3g_3g_3g_4g_4g_4$	2^{-5}	$\frac{1}{6}$	0.03	2^{-5}	$\frac{1}{6}$	0.03	$\frac{7}{32}$	$\frac{1}{2}$	0.03
$g_3g_4g_4g_4g_5g_4g_4g_4g_3g_4$	0	0	0.04	0	0	0.06	0	0	0.06
$g_3^{10}g_4^{10}$	0	0	0.07	0	0	0.09	0	0	0.1
$g_3^5g_4^{15}g_5^5g_4^3g_5$	$7.45e-9$	0.035	0.12	$7.45e-9$	0.035	0.2	$2.6e-8$	0.103	0.2
$g_3^4g_4^{25}g_5^{25}$	3.7e-9	0.019	0.15	3.73e-9	0.019	0.4	6.52e-9	0.056	0.3

the study of human error or errors in physical systems.

Using the prototype implementation, we report on the robustness of various inputs for this motivating example under the three transducer error models. The property P is as shown in Figure 3 and the transducers $T_0 - T_2$ are as shown in Fig. 4. Table 1 reports the robustness values for various input strings and the running time. We note that while our approach takes about 0.3 seconds for a string of length 50, the prototype can be made much more efficient to reduce the time to compute robustness. Also we note that discounted sum becomes smaller as the strings grow larger while the average robustness value does not. We conclude that average robustness is a more useful measure due to this property in this particular example.

5.2 Robust Pattern Matching in Type-1 Diabetes Data

We will now apply our ideas to the *robust pattern matching* problem for analyzing clinical data for patients with type-1 diabetes. People with type-1 diabetes are required to monitor their blood glucose levels periodically using devices such as continuous glucose monitors (CGMs). Data from CGMs is uploaded online and available for review by clinicians during periodic doctor visits. Many applications such as Medtronic Carelink(tm) support the automatic upload and visualization of this data by clinicians. Physicians are commonly interested in analyzing the data to reveal potentially dangerous patterns of blood glucose levels: (a) *Prolonged Hypoglycemia (P1):* Do the blood glucose levels stay below 70 mg/dl (hypoglycemia) for more than 3 hours continuously? [3] (b) *Prolonged Hyperglycemia (P2):* Do the blood glucose levels remain above 300 mg/dl (hyperglycemia) for more than 3 hours continuously? [4]; and (c) *Rebound Hyperglycemia (P3):* Do the blood glucose levels go below 70 mg/dl and then rise rapidly up to 300 mg/dl or higher within 2 hours? [5]

Note that these patterns specify "bad" events that should not happen. A straightforward and strict pattern matching approach based on specifying the properties above will "hide" potentially bad scenarios that "nearly" match the desired pattern for two main reasons. First, the CGM can be noisy and inaccurate in a way that depends on the actual blood glucose value measured and when it was last calibrated. (see Figure 5 and more detailed description below). Secondly, the cutoffs involved such as 70 mg/dl and 3 hours are not "set in stone". For instance, a clinician will consider a scenario wherein the patient's blood glucose levels stays at 71 mg/dl for 2.75 hours as a serious case of prolonged hypoglycemia even though such a scenario would

[3] Such an event can lead to dangerous (and silent) nighttime seizures.
[4] Such an event can lead to a potentially dangerous condition called diabetic ketacidosis.
[5] Rebound hyperglycemia can lead to large future swings in the blood glucose level, raising the burden on the patient for managing their blood glucose levels.

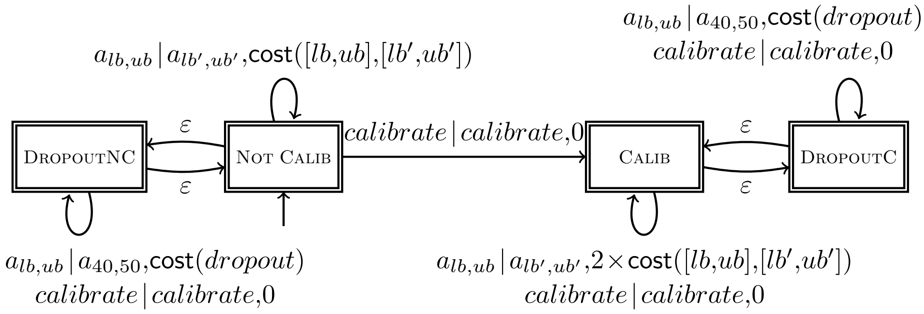

Figure 5 Transducer model for capturing the errors made by continuous glucose monitors.

not satisfy the property P1.

We propose to solve the approximate "pattern matching" problem. I.e, given a string w, a transducer T and a language L, we are looking for a word w' such that $w' \in L$ and $C_T(w',w)$ is *as small as possible*. In other words, we solve the threshold synthesis problem (Problem 6) for a language L that is the complement of P1 (P2 or P3).

We partition the range of CGM outputs [40,400] mg/dl into intervals of size 10 mg/dl over the range [40,80] mg/dl and 20 mg/dl intervals over the remaining range [80,400] mg/dl. This yields a finite alphabet Σ where $|\Sigma| = 20$. For instance $a_{60,70} \in \Sigma$ represents a range $[60,70] mg/dl$. CGMs provide a reading periodically at 5 minute intervals. This yields a string where each letter describes the interval that contains the glucose value.

Transducer. The CGM error model is given by a transducer that considers possible errors that a CGM can make (see Fig. 5). The transducer has four states: (a) Not Calib denoting that no calibration has happened, (b) Calib: denoting a calibration event in the past, (c) DropoutNC: a sensor drops out under the non calibrated mode and (d) DropoutC: a calibration event has happened and sensor drops out. The cost of changing a reading in the range $[lb,ub]$ to one in the range $[lb',ub']$ is denoted by a function cost(lb,ub,lb',ub') These costs are set to be higher for ranges $[lb,ub]$ that are close to hypoglycemia. Also note that we can model calibration events and the doubling of costs if the sensor is in the calibrated mode.

Property Specifications. We specify the three different properties described above formally using finite state machines over the alphabet Σ as defined above. The prolonged hypoglycemia property can be written as a regular expression: $\Sigma^*(a_{40,50}+a_{50,60}+a_{60,70})^{36}\Sigma^*$ which can be easily translated into an NFA with roughly 38 states. The number 36 represents a period of 180 minutes since CGM values are sampled at 5 minute intervals. Similarly, the other two properties are also easily expressed as NFAs.

Finally, we compose the transducer model with the properties P1-P3 individually and calculate the mean robustness. More precisely, for each sequence of measures w, we compute the minimal threshold ν such that w can be rewritten by T at mean cost ν into some w' satisfying P1 (and P2, P3 respectively). The discounted sum robustness is not useful in this situation since the patterns can match approximately anywhere in the middle of a trace. Also, in most cases the discounted sum robustness value was very close to zero for any discount factor <1 or became forbiddingly large for discount factors slightly larger than 1, due to the large size of the traces.

Patient Data. We used actual patient data involving nearly 50 patients with type-1 diabetes undergoing a clinical trial of an artificial pancreas device, and nearly 40 nights of data per patient, leading to an overall 2032 nights. Each night roughly corresponds to a 12 hour period when CGM data was recorded [20]. This is converted to a string of size 140 (or slightly larger, depending on how many calibration events occurred). The threshold synthesis problem (Problem 6) was solved for each of the input strings, and the results were sorted by the threshold robustness value for properties P1-P3.

Table 2 shows for each property, the total time taken to complete the analysis of the full patient data, and the number of matches obtained corresponding to various threshold values. As the table reveals, *no single trace matches any of the properties perfectly*. However, our approach is more nuanced, and thus, allows us to find numerous approximate matches that can be sorted by their robustness threshold values. Note that many of the input traces yield a threshold value of ∞: this signifies that no possible translation as specified by the transducer can cause the property to hold.

Table 2 Total time taken per property and number of matches for various ranges of the threshold.

Prop.	Total Time	Threshold Values synthesized				
		0	(0,0.1]	(0.1,1.0]	>1.0	∞
$P1$	$4hr10m31s$	0	8	2	95	1927
$P2$	$2hr10m30s$	0	28	13	0	1991
$P3$	$2h0m9s$	0	11	10	0	2011

Figure 6 shows two of the approximate pattern matches obtained with a small robustness value. Notice that the CGM values on the left do not satisfy the criterion for a "prolonged hypoglycemia" for 3 hours (P1) in a strict sense due to a single point at the end of the trace that is slightly above the 70 mg/dl threshold. Nevertheless, our approach assigns this trace a very low robustness. Likewise, the plot on the right shows a rapid rise from a hypoglycemia to a hyperglycemia within 120 minutes (P3) towards the beginning, except that the peak value just falls short of the threshold of 300 mg/dl.

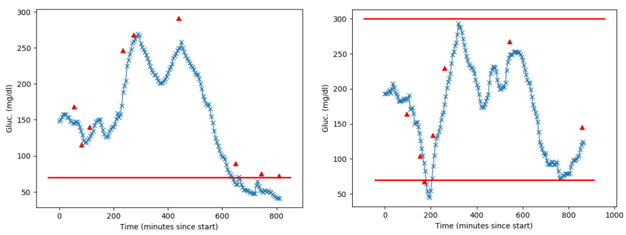

Figure 6 Examples of patterns with small robustness thresholds for properties P1 (left) with robustness value of 0.7, and P3 (right) with robustness 0.02. The red triangles show calibration events.

Note that related work in the area of monitoring cyber-physical systems (CPS) mentioned earlier [16, 14, 12, 1] can be used to perform approximate pattern matching using robustness of temporal properties over hybrid traces. However, we note important differences that are achieved due to the theory developed in this paper. For one, the use of a transducer can provide a nuanced model of how errors transform a trace, wherein the transformation itself changes based on the transducer state. A detailed transducer model of CGM errors remains beyond the scope of this study but will likely be desirable for applications to the analysis of patterns in type-1 diabetes data.

6 Conclusion

In conclusion, we have shown how notions of robustness can be defined through weighted transducers along with approaches for solving the threshold and kernel synthesis problems for various cost aggregators such as `Sum`, `DSum` and `Mean`. In the future, we will investigate these notions for richer classes of systems including timed and hybrid systems. We also plan to investigate connections to robust learning of automata from examples.

References

1. Takumi Akazaki and Ichiro Hasuo. Time robustness in MTL and expressivity in hybrid system falsification. In *Computer Aided Verification (CAV)*, volume 9207 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2015.
2. Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In Tevfik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, volume 6996 of *Lecture Notes in Computer Science*, pages 482–491. Springer, 2011. `doi:10.1007/978-3-642-24372-1\_37`.
3. Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1):3–34, 1995.
4. Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
5. Rajeev Alur, Sampath Kannan, Kevin Tian, and Yifei Yuan. On the complexity of shortest path problems on discounted cost graphs. In *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 44–55, 2013.
6. Rajeev Alur and Parthasarathy Madhusudan. Visibly pushdown languages. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 202–211. ACM, 2004.
7. Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donze, Georgios Fainekos, Oded Maler, Dejan Nickovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification*, volume 10457 of *LNCS*, pages 135–175, 2018.
8. Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *International Conference on Concurrency Theory*, pages 135–150. Springer, 1997.
9. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *Logical Methods in Computer Science*, 6(3), 2010. URL: `http://arxiv.org/abs/1007.4018`.
10. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Logic*, 11(4):23:1–23:38, July 2010. URL: `http://doi.acm.org/10.1145/1805950.1805953`, `doi:10.1145/1805950.1805953`.
11. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, 2018.
12. Jyotirmoy V. Deshmukh, Rupak Majumdar, and Vinayak S. Prabhu. Quantifying conformance using the skorokhod metric. *Formal Methods Syst. Des.*, 50(2-3):168–206, 2017.
13. Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer, 2009.
14. Alexandre Donze and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010.
15. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
16. Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.
17. Thomas A. Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of finite-state transducers. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 431–443. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. `doi:10.4230/LIPIcs.FSTTCS.2014.431`.
18. Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978. `doi:10.1016/0012-365X(78)90011-0`.

19 Dexter Kozen. Lower bounds for natural proof systems. In *Foundations of Computer Science*, pages 254–266, 1977.

20 David M. Maahs, Peter Calhoun, Bruce A. Buckingham, H. Peter Chase, Irene Hramiak, John Lum, Fraser Cameron, B. Wayne Bequette, Tandy Aye, Terri Paul, Robert Slover, R. Paul Wadwa, Darrell M. Wilson, Craig Kollman, and Roy W. Beck. A randomized trial of a home system to reduce nocturnal hypoglycemia in type 1 diabetes. *Diabetes Care*, 37(7):1885–1891, 2014. `doi:10.2337/dc13-2159`.

21 Aurelien Rizk, Gregory Batt, Francois Fages, and Sylvain Soliman. Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. *Theor. Comput. Sci.*, 412(26):2827–2839, 2011.

22 Roopsha Samanta, Jyotirmoy V. Deshmukh, and Swarat Chaudhuri. Robustness analysis of string transducers. In Dang Van Hung and Mizuhito Ogawa, editors, *Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2013. Proceedings*, volume 8172 of *Lecture Notes in Computer Science*, pages 427–441. Springer, 2013. `doi:10.1007/978-3-319-02444-8\_30`.

23 Stefan Schwoon. *Model-checking pushdown systems.* PhD thesis, Technische Universität München, 2002.

24 Masaki Waga, Étienne André, and Ichiro Hasuo. Symbolic monitoring against specifications parametric in time and data. In *Computer Aided Verification*, pages 520–539, Cham, 2019. Springer International Publishing.

Appendix

Proof of Lemma 10

Proof. We first trim the graph G by removing all the vertices that cannot be reached from V_I and that cannot reach V_F as those vertices cannot participate to paths from V_I to V_F. The set of paths from V_I to V_F is empty iff the trimmed graph is empty and then the infimum is equal to $+\infty$. Now, we assume the trimmed graph to be non-empty, i.e. there is at least one path from V_I to V_F. In that case, the infimum value is guaranteed to be a non-negative rational number.

We now consider the three measures in turn. For $\mathtt{Sum}$, computing the infimum amounts to computing a shortest path in a finite graph with non-negative weights. Any PTime algorithm that solves this problem can be used, e.g. Dijkstra shortest path algorithm. In the case of sum, the infimum is always realized by a (simple) shortest path.

For $\mathtt{Mean}$, we first note that the infimum is either realized by a simple path from V_I to V_F of minimal $\mathtt{Mean}$ value, or it is equal to the minimal $\mathtt{Mean}$ value among the simple cycles in the graph. Indeed, if c is a cycle of $\mathtt{Mean}$ m which is smaller than the $\mathtt{Mean}$ value of any path from V_I to V_F then the family of paths $\rho_k = p \cdot c^k \cdot s$, where p is simple path from V_I to c and s is a simple path from c to V_F (such simple paths exist as the graph is trimmed), is such that $\lim_{k \mapsto +\infty} \mathtt{Mean}(\rho_k) = \mathtt{Mean}(c)$ and $\mathtt{Mean}(c)$ is the infimum. Now if all the simple cycles have a value larger than the infimum, they cannot participate to a path or a family of paths that realize the infimum as those cycles can be systematically removed and give paths with smaller values. Now, we note that the minimum value of simple paths from V_I to V_F can be computed in PTime by a simple dynamic program that considers the minimal values of paths of lengths at most equal to the number of states in the trimmed graph. Moreover, the minimum mean value of simple cycles in the trimmed graph can be computed in PTime using the Karp algorithm [18]. It is easy to see that the infimum is feasible iff it equals the minimum $\mathtt{Mean}$ value of simple paths.

We now turn to the $\mathtt{DSum}$ measure. Remember that the graph is trimmed according to V_I and V_F. Theorem 1 of [5] tells us that we can compute for all $v \in V$, the infimum of $\mathtt{DSum}$ values x_v of paths reaching the target V_F from v, in PTime. According to Lemma 1 of [5], and similarly to the case of $\mathtt{Mean}$, for all $v_I \in V_I$, the infimum $\mathtt{DSum}$ value x_{v_I} of paths from v_I to some $v_F \in V_F$ is either realized by a simple path or by a family of paths of the form $p \cdot c^k \cdot s$. This is because if it is beneficial to include a cycle c to reduce the cost of a path from v_I to v_F then it is beneficial to repeat the cycle arbitrarily many times. In particular, the infimum value is feasible only when there exists a simple path with this value. In order to decide the feasibility of the values x_{v_I} for all $v_I \in V_I$, we consider a subgraph where we keep only those edges $e = (v,v')$ such that the optimal value x_v of v can be realised through the vertex v'. Formally, we construct $G' = (V,E')$ with $E' \subseteq E$ and such that $(v,v') \in E'$ if $x_v = \lambda x_{v'} + \mathtt{W}(v,v')$. We claim that, V_F is reachable from v in G' iff x_v is feasible in G from v, hence testing feasibility boils down to checking the existence of a path in G'.

The left-to-right implication comes by induction on the length of the path π to reach some $v_F \in V_F$ from v. If $v \in V_F$ then $|\pi| = 0$, $x_v = 0$ and this value is feasible. Assume $v \notin V_F$ and $\pi = (v,v')\pi'$. By induction hypothesis, $x_{v'}$ is feasible by some path π'' from v' to V_F. By construction of G' we have $x_v = \lambda x_{v'} + \mathtt{W}(v,v')$. Hence x_v is feasible by $(v,v')\pi''$. For the right-to-left implication, if $v \in V_F$ it is trivial, so assume that $v \notin V_F$ and let $\pi = (v,v')\pi'$ a path that realises x_v. Assume $x_v > \lambda x_{v'} + \mathtt{W}(v,v')$. This contradicts the optimality of x_v, as π witnesses a better discounted value from v to V_F. Assume $x_v < \lambda x_{v'} + \mathtt{W}(v,v')$, then since π realises x_v, we have $x_v = \mathtt{W}(v,v') + \lambda \mathtt{DSum}(\pi')$. It implies $\mathtt{DSum}(\pi') < x_{v'}$. This contradicts the minimality of $x_{v'}$, as then π' witnesses a better value for paths from v' to V_F. Hence $x_v = \lambda x_{v'} + \mathtt{W}(v,v')$ and (v,v') is an edge of G'. By induction on the length of π, we can also conclude that π' is a path of G' and then π is a path of G' from v to V_F. ◀

Proof of Lemma 14

Proof. First, we show that the complement of $\mathtt{Rob}_T(\nu,L)$, defined as

$$\overline{\mathtt{Rob}_T(\nu,L)} = \{w_1 \mid \exists w_2 \cdot \mathsf{Sum}_T(w_1,w_2) < \nu \wedge w_2 \notin L\}$$

is regular. First, let us assume that L is given by some NFA A, let $\overline{A}$ be a DFA recognizing the complement of L. We first transform T into $T \otimes \overline{A}$, which simulates T and controls that the output words belong to $\overline{L}$. In particular, it rejects whenever the rewriting by T is in L. It is obtained as a product of T with $\overline{A}$ run on the output, with set of states $Q_T \times Q_{\overline{A}}$. It accepts whenever the final pair of states (p,q) is a pair of accepting states both for T and $\overline{A}$. Then, we have the following:

$$\overline{\mathtt{Rob}_T(\nu,L)} = \{w_1 \mid \exists w_2 \cdot \mathsf{Sum}_{T\otimes\overline{A}}(w_1,w_2) < \nu\}$$

Now, by definition of $\mathsf{Sum}_{T\otimes\overline{A}}(w_1,w_2)$ we have $w_1 \in \overline{\mathtt{Rob}_T(\nu,L)}$ iff there exists a word w_2 and an accepting run r over (w_1,w_2) such that $\mathsf{Sum}(r) < \nu$. Therefore, we can project $T\otimes\overline{A}$ on its input dimension (thus, we just ignore the outputs) and obtain a $\mathtt{Sum}$-automaton that we call U such that $\overline{\mathtt{Rob}_T(\nu,L)} = \{w_1 \mid U(w_1) < \nu\}$, where $U(w_1)$ is defined as $+\infty$ if there is no accepting run of U on w_1, and as the minimal sum of the accepting runs on w_1 otherwise. Complementing again, we get: $\mathtt{Rob}_T(\nu,L) = \{w_1 \mid U(w_1) \geq \nu\}$. Now, we apply directly Lemma 13 on U to conclude for regularity. The state-complexity is again given by Lemma 13 and the fact that U has $n_T \times n_L$ states. ◀

Proof of Lemma 18

Proof. Let r be a run of length n of T. Since T is trim, there exists a continuation r' of r, and moreover we have $\mathtt{DSum}(rr') = \mathtt{DSum}(r) + \lambda^n \mathtt{DSum}(r')$. We have $\mathtt{DSum}(r') \leq \sum_{i=0}^{+\infty} \lambda^i \mu = \mu(1-\lambda)^{-1}$ where μ is the largest absolute weight of T. We let $B_n = \lambda^n \mu (1-\lambda)^{-1}$. Let n^* be the smallest non-negative integer such that $B_{n^*} \leq \epsilon/2$ (it exists since B_n is strictly decreasing of limit 0). Assume that the length of r is greater than n^* i.e. $n \geq n^*$. As a consequence $B_n \leq B_{n^*}$. Since ν is ϵ-isolated, we have two cases:

i. If $\mathtt{DSum}(rr') \leq \nu - \epsilon$ then $\mathtt{DSum}(r) \leq \nu - \epsilon$ since $\mathtt{DSum}(r) \leq \mathtt{DSum}(rr')$ by non-negativity of the weights of T
ii. If $\mathtt{DSum}(rr') \geq \nu + \epsilon$ then $\mathtt{DSum}(r) \geq \nu + \epsilon - \lambda^n \mathtt{DSum}(r')$. Moreover $\lambda^n \mathtt{DSum}(r') \leq B_n \leq B_{n^*} \leq \epsilon/2$ by construction. So $-\lambda^n \mathtt{DSum}(r') \geq -\epsilon/2$ which implies $\mathtt{DSum}(r) \geq \nu + \epsilon/2$.

We have just shown that either $\mathtt{DSum}(r) \leq \nu - \epsilon$ by (i) or $\mathtt{DSum}(r) \geq \nu + \epsilon/2$ by (ii). We prove now that, for all continuation r' of r we have (i) implies $\mathtt{DSum}(rr') \leq \nu - \epsilon$ and (ii) implies $\mathtt{DSum}(rr') \geq \nu + \epsilon$. In the first case, assume by contradiction that (i) holds and some continuation r' of r satisfies $\mathtt{DSum}(rr') \geq \nu + \epsilon$. As a consequence $\lambda^n \mathtt{DSum}(r') \geq 2\epsilon$, which is impossible since $\lambda^n \mathtt{DSum}(r') \leq B_n \leq B_{n^*} \leq \epsilon/2$. In the second case, if $\mathtt{DSum}(r) \geq \nu + \epsilon/2$ then any continuation r' of r satisfies $\mathtt{DSum}(rr') \geq \mathtt{DSum}(r) > \nu + \epsilon/2$. Since ν is ϵ-isolated, we get $\mathtt{DSum}(rr') \geq \nu + \epsilon$. ◀

Proof of Lemma 19

Proof. For all n, we let $B_n = \lambda^n W (1-\lambda)^{-1}$, as in the proof of Lemma 18. A run r on a pair (w_1,w_2) is called *bad* if $\mathtt{DSum}(r) \leq \nu$, $w_2 \notin L$ and r is accepting. Not that necessarily,

$w_1 \notin \mathtt{Rob}_T(\nu,L)$. The run r is called *dangerous* if $|r| \geq n$ and $\mathtt{DSum}(r) \leq \nu - B_n$. A dangerous run r can possibly be extended to a bad run rr'. It is possible iff there exists a continuation r' of r such that the output of rr' is not in L. Note that the cost of rr' does not matter because the largest value r' can achieve is B_n, keeping $\mathtt{DSum}(rr')$ smaller than ν. Hence, when a dangerous run is met, only a regular property has to be tested to extend it to a bad run. We exploit this idea in the automata construction. Namely, A_n will accept words for which there exists a bad run of length n at most, or a dangerous run of length n which can be extended to a bad run.

- *Automata construction* Let $\mathsf{Runs}_T^{\leq n}$ be the runs of T of length at most n, and Q its set of states. We assume that for all $(w_1,w_2) \in R_T$, $w_2 \notin L$ holds. This can be ensured by taking the synchronised product of T (on its outputs) with an automaton recognizing the complement of L. Let us now build the NFA A_n. Its set of states is $\mathsf{Runs}_T^{\leq n} \cup Q$. Its transitions are defined as follows: for all T-runs r of length $n-1$ at most ending in some state q, for all $\sigma \in \Sigma_\varepsilon$, if there exists a transition t of T from state q on reading σ, then we create the transition $r \xrightarrow{\sigma} rt$ in A_n. From any run r of length n, we consider two cases: if r is not dangerous, then r has no outgoing transitions in A_n. If r is a dangerous run, then we add some ε-transition to its last state: $r \xrightarrow{\varepsilon} p$ where p is the last state of r. Finally, we add a transition from any state q to any state q' on σ in A_n whenever there is a transition from q to q' on input σ in T. Accepting states are bad runs of $\mathsf{Runs}_T^{\leq n}$ and accepting states of T.
- *Correctness* Let us show that the family A_n satisfies the requirements of the lemma. First, we show that $L(A_n) \subseteq L(A_{n+1})$. Let $w \in L(A_n)$ and ρ some accepting run of A_n on w. To simplify the notations, we assume here in this proof that runs of A_n, A_{n+1} and T are just sequences of states rather than sequences of transitions. By definition of A_n, ρ can be decomposed into two parts $\rho_1\rho_2$ such that $\rho_1 \in (\mathsf{Runs}_T^{\leq n})^*$ and $\rho_2 \in Q^*$ with an ϵ-transition from the last state of ρ_1 to the first of ρ_2. We consider two cases. If $|\rho_2| = 0$, then $\rho = \rho_1$ and by definition of A_{n+1}, ρ is still an accepting run of A_{n+1}. In the other case, there is a dangerous run r of T such that ρ_1 can be written $\rho_1 = r[:1]r[:2]...r[:n]$ where $r[:i]$ is the prefix of r up to position i, and $\rho_2 = q_1q_2...q_k$ is a proper run of T. Note that q_1 is the last state of r by construction of A_n. Moreover, $r\rho_2$ is bad. Since r was dangerous at step n, we also get that rq_2 is dangerous at step $n+1$, in the sense that $|rq_2| = n+1$ and $\mathtt{DSum}(rq_2) \leq \nu - B_{n+1}$, by definition of B_{n+1} and the fact that $\mathtt{DSum}(r) \leq \nu - B_n$. So, we get that the sequence of states $\rho_1.(rq_2).q_2...q_k$ is a run of A_{n+1} on w is accepting in A_{n+1} (note that rq_2 here is a state of A_{n+1} and there is an ϵ-transition from (rq_2) to q_2), concluding the first part of the proof.

Now, suppose that ν is ϵ-isolated for some ϵ. Then, take n^* as given by Lemma 18 and let us show that $\overline{\mathtt{Rob}_T(\nu,L)} \cap \mathrm{dom}(T) \subseteq L(A_{n^*})$ (the other inclusion has just been proved for all n). Let $w \in \mathrm{dom}(T)$ such that $w \notin \mathtt{Rob}_T(\nu,L)$. There exists $(w_1,w_2) \in R_T$ and an accepting run r of T on it such that $\mathtt{DSum}(r) \leq \nu$ and $w_2 \notin L$. In other words, r is bad. If $|r| \leq n^*$, then $r[:1]r[:2]...r[:|r|]$ is an accepting run of A_{n^*} on w, and we are done. Now suppose that $|r| > n^*$. Since ν is ϵ-isolated, we have $\mathtt{DSum}(r) \leq \nu - \epsilon$. By Lemma 18, we also get that $\mathtt{DSum}(r[:n^*]) \leq \nu - \epsilon$. By definition of n^* being the smallest integer such that $B_{n^*} < \epsilon/2$, we get $\mathtt{DSum}(r[:n^*]) \leq \nu - B_{n^*}$, hence $r[:n^*]$ is dangerous. We can conclude since then $r[:1]r[:2]...r[:n^*]r[n^*]r[n^*+1]...r[|r|]$ is an accepting run of A_{n^*} on w. ◀