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.

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 $\sigma$ 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 \( \varphi \) for a trace \( \sigma \) 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 \( \sigma, \sigma' \) in order to check whether traces that are in the epsilon neighborhood of a given trace \( \sigma \) also satisfy the formula: \( (\forall \sigma') \ d(\sigma, \sigma') < \varepsilon \Rightarrow \sigma' \models \varphi \).

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 \( \Sigma^* \), 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 \( \Sigma = \{a_1, a_2, b\} \). 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 \( w : a_1 a_2 a_2 a_2 \) into \( w' : a_1 b b a_2 \) 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 \( w'' : b b a_2 a_1 \) 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.

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 $\tau_i$ 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 $\nu$-neighborhood $N_{\nu} : \{ w' \in \Sigma^* \mid (\exists w \in N) \ c(w,w') \leq \nu \}$. Given a property $L \subseteq \Sigma^*$, we consider the following robustness problems:

- **Robust inclusion:** Given $N, \nu$ and $L$, check whether $N_{\nu} \subseteq L$.
- **Threshold synthesis:** Given $N, L$, find the largest threshold $\nu$ such that $N_{\nu} \subseteq L$.
- **Robust kernel synthesis:** given $N, \nu, L$, 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 $\nu$-robust for any threshold $\nu$ 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 $\nu$-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 $\nu$-robust words in $N = \Sigma^*$ is $(a_1 + a_1a_2 + a_1a_2a_2)^*$, and for $\nu \leq 2$, any word in $\Sigma^*$ is $\nu$-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 $\nu$ 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 $\Sigma$ be an alphabet. We denote the empty word by the symbol $\varepsilon \notin \Sigma$ and we write $\Sigma^*$ for the set of finite words over $\Sigma$. Let $\Sigma_0 = \Sigma \cup \{\varepsilon\}$. 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 $\Sigma$ is a tuple $A = (Q, \mathcal{I}_A, \mathcal{F}_A, \Delta)$ where $Q$ is the finite set of states, $\mathcal{I}_A \subseteq Q$ is the set of initial states, $\mathcal{F}_A \subseteq Q$ 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 $q_0 \in \mathcal{I}_A$ and ends into a final state $q_n \in \mathcal{F}_A$. We denote by $\text{AccRun}_A(u)$ the set of accepting runs of $A$ on the word $u$. The language defined by $A$ is the set of words $L(A) = \{ u \mid \text{AccRun}_A(u) \neq \emptyset \}$. The automaton $A$ is called deterministic (DFA for short) if $\mathcal{I}_A$ is a singleton and $\Delta$ is a function from $Q \times \Sigma$ to $Q$. We define the representation size of an automaton $A = (Q, \mathcal{I}_A, \mathcal{F}_A, \Delta)$ 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 $\varepsilon$-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 $C: \mathbb{N}^* \to \mathbb{Q}_{\geq 0}$, 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 $C: \mathbb{N}^* \to \mathbb{Q}_{\geq 0}$ be an aggregator function. A C-transducer $T$ is a tuple $(A, \mathcal{W})$ where $A = (Q, \mathcal{I}_A, \mathcal{F}_A, \Delta)$ is an NFA over $(\Sigma_0 \times \Sigma^*) \setminus \{(\varepsilon, \varepsilon)\}$ and the function $\mathcal{W}: \Delta \to \mathbb{N}$ associates weights to each transition.

Given a transition $t = (q, a, v, q') \in Q \times \Sigma_0 \times \Sigma^* \times Q$, we write $\text{Orig}(t) = q$, $\text{In}(t) = a$, $\text{Out}(t) = v$, and $\text{Dest}(t) = q'$. We say that a transition $t \in \Delta$ can be triggered by $T$ if it is in state $\text{Orig}(t)$
and reads In(t) on its input (note that it is always possible to read In(t) = ε). It, then, moves
to Dest(t) and rewrites its input into Out(t). A run r = t_1...t_n of T is a run of A. We write
In(r) = In(t_1)...In(t_n) and Out(r) = Out(t_1)...Out(t_n) and say that r is a run of T on the pair
of words (In(r),Out(r)). Let (u_1,u_2) = (In(r),Out(r)). If moreover r is accepting, we say that
(u_1,u_2) is accepted by T, and denote by 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 ∈ Σ^*. We denote
the weight sequence of r by w(r) = w(t_1)...w(t_n) and its corresponding (aggregated) cost is
C(r) = C(w(r)).

A transducer T defines a relation from Σ^* to itself, called a translation, denoted RT and
defined by: RT = {(u_1,u_2) | AccRun_T(u_1,u_2) ≠ ∅}. The domain of T, denoted dom(T) is the
set of words u_1 for which there exists u_2 such that (u_1,u_2) ∈ RT. The cost of a pair of words
(u_1,u_2) is given by:

\[ C_T(u_1,u_2) = \begin{cases} +∞ & \text{if } (u_1,u_2) /∈ RT \\ \min \{C(r) | r ∈ AccRun_T(u_1,u_2) \} & \text{otherwise.} \end{cases} \]

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 ν ∈ Q
and an input word u_1 ∈ dom(T), we define the threshold output language T_{≤ν}(u_1) of u_1 as:
T_{≤ν}(u_1) = \{ u_2 | C_T(u_1,u_2) ≤ ν \}. This notation extends naturally to languages N ⊆ Σ^* by
setting: T_{≤ν}(N) = \bigcup_{u_1 ∈ N∩dom(T)} T_{≤ν}(u_1).

**Assumption 3.** We restrict our attention to C-transducers T that satisfy the condition that
for all u ∈ dom(T), C_T(u,u) = 0 (in particular (u,u) ∈ RT). 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 λ ∈ Q∩(0,1) be a discount factor. Given a sequence of weights π = τ_1...τ_n,
those three functions are defined by:

\[ \text{Sum}(π) = \sum_{i=1}^{n} τ_i \quad \text{Mean}(π) = \begin{cases} 0 & \text{if } π = ε \\ \frac{\text{Sum}(π)}{n} & \text{otherwise} \end{cases} \quad \text{DSum}(π) = \sum_{i=1}^{n} λ^{(i-1)} τ_i \]

**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 C = Sum, this definition of Sum-automaton coincides with the classical
notion weighted automata over the semiring (\mathbb{N}\cup\{+∞\}, min,+) from [15].

**Robustness problems.** We study the following three fundamental problems related to
robustness for three different aggregator functions C ∈ \{Sum,Mean,DSum\}. Given a threshold
ν ∈ Q, a C-transducer T and a regular language L, a word u ∈ dom(T) is said to be ν-robust
(or just robust if ν is clear from the context) if T_{≤ν}(u) ⊆ L. In other words, all its rewritings
of cost ν at most are in L. A language N ⊆ Σ^* is said to be ν-robust if N∩dom(T) contains
only ν-robust words. Finally, the ν-robust kernel of T is the set Rob_T(ν,L) of ν-robust words:
Rob_T(ν,L) = \{ u ∈ dom(T) | T_{≤ν}(u) ⊆ L \}. We prove that as the error threshold grows, so does
the robust kernel.
Proposition 4. Given $\nu, \nu' \in \mathbb{Q}_{\geq 0}$, a C-transducer $T$ and a regular language $L$, we have that $\nu' \leq \nu \implies \text{Rob}_T(\nu', L) \subseteq \text{Rob}_T(\nu, L)$.

Proof. By definition $T_{\leq \nu}(u_1) = \{ u_2 \mid C_T(u_1, u_2) \leq \nu \}$. For all $u_1 \in \text{dom}(T)$ we have that $u_1 \in \text{Rob}_T(\nu, L)$ if and only if both $u_2 \in L$ and $C_T(u_1, u_2) \leq \nu$ hold. Clearly $u_1 \in \text{Rob}_T(\nu, L)$ implies $u_1 \in \text{Rob}_T(\nu', L)$ 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 $C \in \{\text{Sum, Mean, } \text{DSum}\}$.

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 $N \subseteq \text{Rob}_T(\nu, L)$, 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 $\mathbb{Q}_{\geq 0} = G \cup B$ into sets $G$ and $B$ of good and bad thresholds, i.e.

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

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 $u \in \text{Rob}_T(\nu, L)$.

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 $\text{Sum, Mean, DSum}$ measures is in PTIME.

In the following, we assume that $N = \text{dom}(T)$. 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 $G = \{ \nu \in \mathbb{Q}_{\geq 0} \mid \text{dom}(T) \subseteq \text{Rob}_T(\nu, L) \}$ 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. $\nu_{T,L} = \inf B = \inf \{ \nu \in \mathbb{Q}_{\geq 0} \mid \text{dom}(T) \not\subseteq \text{Rob}_T(\nu, L) \}$. As illustrated by the following example, computing $\nu_{T,L}$ allows us to compute $G = [0, \nu_{T,L}]$ and $B = [\nu_{T,L}, +\infty)$.

Example 8. Let $\Sigma = \{a, b, c\}$ and $C \in \{\text{Mean, DSum}\}$. Consider the best threshold problem for $T$ the C-transducer of Figure 2, $N = \text{dom}(T) = a^*$ 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 $\text{Mean}$ measure, the cost of a translation into $c^*$ is exactly 1 while the one into $b^*$ range over $[0, 1]$. Hence $\nu^{\text{Mean}}_{T,L} = 1$ and the set partition of good and bad thresholds is $G^{\text{Mean}} = [0, 1)$ and $B^{\text{Mean}} = [1, +\infty)$. In the case of
\[ \text{DSum} \text{ with discount factor 0.5, the cost of a translation into } c^* \text{ range over } [2, 2.5) \text{ while the one into } b^* \text{ range over } (0, 2). \text{ So } \nu_{T,L}^{\text{DSum}} = 2 \text{ and the thresholds are partitioned by } G^{\text{DSum}} = [0, 2) \text{ and } B^{\text{DSum}} = [2, +\infty). \]

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 \( T = (Q, Q_I, Q_F, \Delta, W) \) and \( A = (P, P_I, P_F, \delta) \), the synchronised product \( G_{T,A} = (E, V, \Sigma') : E \rightarrow 8 \) is such that:

1. \( V = Q \times P \)
2. \( E \) is the set of edges \( e = (q, p) \rightarrow (q', p') \) such that there exists \( a \in \Sigma' \) and a transition \( t = (q, a, u, q') \in \Delta \) such that \( p' = \delta(p, a) \) where \( \delta \) has been extended to words in the expected way. We say that \( e \) is compatible with \( t \).
3. For all \( e \in E \), \( w'(e) = \min\{w(t) \mid e \text{ is compatible with } t \} \).

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 \( \pi \) in this graph as a sequence of edges \( e_1...e_n \), we let \( C(\pi) = C(w'(e_1)...w'(e_n)) \).

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

\textbf{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. \( \nu_{T,L} = \inf \{C(\pi) \mid \exists s_0 \in V_I \exists s_f \in V_F \exists s_f \stackrel{\pi}{\rightarrow} G_{T,A} s_f \} \).

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

Suppose that \( \nu_{T,L} \) is strictly smaller than this infimum (that we denote \( m \)) and take some rational number \( \nu \) such that \( \nu_{T,L} < \nu < m \). Since \( \nu_{T,L} < \nu \), it is a bad threshold which means that there exists \( u_1 \in \text{dom}(T) \) such that \( u_1 \notin \text{Rob}_T(\nu, L) \). Hence there exists \( u_2 \notin L \) such that \( C_T(u_1, u_2) \leq \nu \), and by definition of \( G_{T,A} \), there exists a path \( \pi \) from \( V_I \) to \( V_F \) of value \( C(\pi) \leq \nu \). 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 \textit{PTime} and it is also decidable in \textit{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 \( G = (V, E, w : E \rightarrow \mathbb{Q}_{\geq 0}) \), 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 \( \text{PTime} \) for all \( C \in \{ \text{Sum}, \text{DSum}, \text{Mean} \} \). Moreover, we can decide in \( \text{PTime} \) 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 \( \text{Sum} \), the infimum can be computed in \( \text{PTime} \) using Dijkstra’s algorithm and it is always feasible. For \( \text{Mean} \), we first note that the infimum is the \( \text{Mean} \) 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 feasible but can be approximated as close as possible by iterating the cycle. So, the infimum is feasible iff it is the \( \text{Mean} \) value of a simple path. The minimal \( \text{Mean} \) values amongst simple paths and cycles can be computed in \( \text{PTime} \) with dynamic programming thanks to \([18]\). For \( \text{DSum} \), Theorem 1 of \([5]\) provides a \( \text{PTime} \) algorithm that computes for all \( v \in V \), the infimum of \( \text{DSum} \) 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 \( \text{NFA} \) and \( L \subseteq \Sigma^* \) given as a \( \text{DFA} \), the set partition of good and bad thresholds \((G, B)\) for \( C \in \{ \text{Sum}, \text{DSum}, \text{Mean} \} \) can be computed in \( \text{PTime} \).

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 \( \text{PTime} \) 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 \( G = [0,0) = \emptyset \).

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

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

4 Robust Kernel Synthesis

In this section, we show that the robust kernel is regular for \( \text{Sum} \)-transducers, and checking its emptiness is \( \text{PSpace} \)-complete. For \( \text{Mean} \), we show that it is not necessarily regular, and checking its emptiness is undecidable. For \( \text{DSum} \), 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 \( \text{Sum} \)-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 \( O((\nu+2)^n) \) 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 \( \nu \) to each word of its complement.

Then, \( U(w) \geq \nu \) iff all the accepting runs on \( w \) have value at least \( \nu \). 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 \( \nu \), by counting in its states. If \( Q \) is the set of states of \( U \), the set of states of \( D \) is \( 2^Q \times \{0,\ldots,\nu-1,\nu^+\} \), where \( \nu^+ \) intuitively means any value \( \geq \nu \). We extend natural addition to \( X = \{0,\ldots,\nu-1,\nu^+\} \) 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 \( P \rightarrow P' \) in \( D \) iff \( P' = \{(q',i)+j) \mid (q,i) \in P \land q + i \overset{\cdot j}{\rightarrow} U q' \} \). A state \( P \) is accepting if \( P \cap (\{Q\} \setminus \{0,\ldots,\nu-1\}) = \emptyset \), 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 \( 2^{\nu^+} \). But the following simple observation allows us to get a better state complexity. Consider an input word of the form \( wv \). 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 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 \( j+\nu \geq \nu \) 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 \( Dopt \) the resulting “optimised” DFA. Its states can be therefore seen as functions from \( Q \) to \( \{0,\ldots,\nu-1,\nu^+\} \), so that we get the claimed state-complexity. \( \blacksquare \)

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

Proof of this lemma is provided in the appendix.

**Theorem 15.** Let \( T \) be a 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 \( w \in Rob_T(\nu,L) \). The hardness holds even if \( \nu \) 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, \( Rob_T(\nu,L) \) is recognisable by a DFA with \( O((\nu+2)^{n_T \times n_L}) \) 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 \( A_1,\ldots,A_n \), over some alphabet \( \Gamma \). In particular, we construct \( T, \nu \) and a DFA \( A \) such that \( \bigcap_i L(A_i) \neq \emptyset \) iff there exists a robust word with respect to \( T, \nu \) and \( L \).

We define the alphabet as \( \Sigma = \Gamma \cup \{\#_1,\ldots,\#_{n-1}\} \) where we assume that \( \#_1,\ldots,\#_{n-1} \notin \Gamma \), and construct a transducer \( T \) which reads a word \( w^{-1} \) 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 $\#^k_i$ for all $i \in \{1,\ldots,n\}$. The identity rewriting has total weight 0 while the rewriting into $\#^k_i$ 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 $T_1,\ldots,T_n,T_\nu$. For all $i \in \{1,\ldots,n\}$, $T_i$ simulates $A_i$ on the input and outputs $\#$, whenever it reads an input letter different from $\vdash$, with weight 0. When reading $\vdash$ from an accepting state of $A_i$, it outputs $\vdash$ with weight 1, and if it reads $\vdash$ from a non-accepting state, it outputs $\vdash$ with weight 0. Finally, $T_\nu$ just realizes the identity function with weight 0. Note that $T$ has polynomial size in $A_1,\ldots,A_n$ and it is letter-to-letter and (input,output)-deterministic.

Now we prove that a word $w \vdash$ is robust if $w \in \bigcap_i L(A_i)$. Assume that there exists a robust word $w \vdash$ for the property $L = (\Gamma \cup \{-\})^*$ and threshold $\nu = 0$. Equivalently, it means that for all rewritings $\alpha \in \Sigma^*$, if $\Sigma_T(w \vdash,\alpha) \leq 0$ then $\alpha \in L$. It is equivalent to say that all its rewritings $\alpha$ satisfies either $\Sigma_T(w \vdash,\alpha) \geq 1$ or $\alpha \in L$. By definition of $T$, it is equivalent to say that all rewritings $\alpha$ are such that either $\alpha \in (\#^k_i)^* \vdash$ for some $i$ and $w \in L(A_i)$, or $\alpha = w \vdash$. Since $T$ necessarily rewrites $w \vdash$ into $w \vdash$, as well as into $(\#^k_1,\ldots,(\#^k_n)^k$, where $k = |w|+1$, the latter assumption is equivalent to saying that $w \in L(A_i)$ for all $i \in \{1,\ldots,n\}$, 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 Mean-transducer $T$ and $\nu \in \mathbb{Q}_{\geq 0}$, the language $\text{Rob}_T(\nu,L)$ is not necessarily regular, but recursive.

Proof. Consider the language $L = \{w | \exists i \in \mathbb{N}: w(i) = a\}$ on the alphabet $\Sigma = \{a,b\}$, i.e. the set of words on $\Sigma$ 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 $\nu$ 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. $\text{Rob}_T(\frac{1}{2},L) = \{w \ | w_{2a} > w_{2b}\}$, which is not regular. Note that in general $\text{Rob}_T(\nu,L)$ 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 Mean-transducer and $\nu \in \mathbb{Q}_{\geq 0}$. Determine whether $\text{Rob}_T(\nu,L) \neq \emptyset$ is undecidable. It holds even if $T$ is io-unambiguous.

Proof. Let $A$ be a 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 $\Sigma^*$, the threshold $\nu$ as the maximal absolute weight of $A$ and $T$ such that:

\[
\text{Mean}_T = \bigcup \{ (w,w) \mapsto 0 | w \in \Sigma^* \} \cup \{ (w,r_w) \mapsto X_{r_w} + \nu | w | r_w \text{ run of } A \text{ over } w \in \Sigma^* \text{ with value } X_{r_w} \}
\]

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 $\nu$. 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 $\text{Rob}_T(\nu,L) = \emptyset$ iff $\forall w : A(w) \leq 0$, i.e.

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


We have the following equivalences: \( \forall w_1 \exists w_2 \in L. \text{Mean}_T(w_1, w_2) \leq \nu \) if and only if all \( w_1 \), there exists an accepting run \( r \) of \( A \) on \( w_1 \) such that \( \text{Mean}_T(w_1, r) \leq \nu \), i.e. \( \text{Sum}_T(w_1, r) \leq \nu |w_1| \) and by definition of \( T \), it is equivalent to asking that \( \text{Sum}_A(r) + \nu |w_1| \leq \nu |w_1| \), i.e. \( \text{Sum}_A(r) \leq 0 \). Hence, the latter statement is equivalent to the fact that for all words \( w_1 \), there exists an accepting run of \( A \) of value \( \leq 0 \). 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 \( \text{DSum}-\text{transducer} \), we conjecture that \( \text{Rob}_T(\nu, L) \) is in general non-regular. This claim is substantiated by the fact that \( \text{DSum}-\text{automata} \) over \( \mathbb{Q} \) and \( \omega \)-words have in general non-regular cut-point languages, i.e. the set of words of \( \text{DSum} \) value below a given threshold is in general non-regular [9]. With a proof similar to that of Theorem 17 for \( \text{Mean}-\text{transducers} \), it is possible to show that the universality problem for \( \text{DSum}-\text{automata} \), which is open to the best of our knowledge, reduces to checking the emptiness of the robust language of a \( \text{DSum}-\text{transducer} \).

Following an approach that originates from the theory of probabilistic automata, it has been shown that cut-point languages are regular when the threshold is \( \varepsilon \)-isolated [9]. Formally, a threshold \( \nu \in \mathbb{Q} \) is \( \varepsilon \)-isolated, for \( \varepsilon > 0 \) and for some \( \text{DSum}-\text{transducer} \) \( T \) if, for all accepting runs \( r \) of \( T \), \( \text{DSum}_T(r) \in [0, \nu - \varepsilon] \cup [\nu + \varepsilon, +\infty) \). It is isolated if it is \( \varepsilon \)-isolated for some \( \varepsilon \).

Our objective now is to show that when \( \nu \) is isolated, then \( \text{Rob}_T(\nu, L) \) 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 \( \text{Rob}_T(\nu, L) \), and which is guaranteed to terminate whenever \( \nu \) is \( \varepsilon \)-isolated for some \( \varepsilon \). 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 \( \text{PTIME} \), just by removing states that do not admit any continuation (this can be tested in \( \text{PTIME} \)).

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

1. \( \text{DSum}(r) \leq \nu - \varepsilon \) and any continuation \( r' \) of \( r \) satisfies \( \text{DSum}(rr') \leq \nu - \varepsilon \)
2. \( \text{DSum}(r) \geq \nu + \varepsilon / 2 \) and any continuation \( r' \) of \( r \) satisfies \( \text{DSum}(rr') \geq \nu + \varepsilon \).

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 \( \nu \) is isolated.

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

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

Moreover, if \( \nu \) is isolated, there exists \( n^* \) such that \( L(A_{n^*}) = \text{Rob}_T(\nu, L) \cap \text{dom}(T) \).
Proof of this lemma is provided in the appendix.

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

\[ \text{Lemma 20.} \text{ Given a regular language } N \text{ (given as some NFA), it is decidable to check whether } \text{Rob}_T(\nu, L) \cap \text{dom}(T) \subseteq N \text{ holds.} \]

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

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

\[ \begin{align*}
1. & \textbf{Compute-Rob}(T, \nu, L) \\
2. & \text{for } n \text{ from 1 to } +\infty \\
3. & \textbf{compute } A_n \quad \text{ // as in Lemma 19} \\
4. & \text{if } \text{Rob}_T(\nu, L) \cap \text{dom}(T) \subseteq L(A_n) \text{ return } A_n \quad \text{ // using Lemma 20}
\end{align*} \]

\[ \text{Lemma 21.} \text{ The algorithm } \textbf{Compute-Rob}(T, \nu, L) \text{ satisfies the following properties:} \]

\[ \begin{align*}
1. & \text{if it terminates, then it returns an automaton recognising } \text{Rob}_T(\nu, L) \cap \text{dom}(T), \\
2. & \text{if } \nu \text{ is isolated, it terminates.}
\end{align*} \]

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

Note that the algorithm may terminate even if \( \nu \) is not isolated. It is the case for instance when the threshold is \( \epsilon \)-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 \( \text{DSum}(r) \leq \nu - \epsilon \) or \( \text{DSum}(r) \geq \nu + \epsilon \). As a corollary of Lemma 21, \( \text{Rob}_T(\nu, L) \) is regular when \( \nu \) is isolated: it suffices to run Algorithm \( \textbf{Compute-Rob} \), complement the automaton and restrict its language to \( \text{dom}(T) \).

\[ \text{Theorem 22.} \text{ Let } T \text{ be a } \text{DSum}-\text{transducer and } \nu \in \mathbb{Q} \text{ and } L \text{ a regular language. If } \nu \text{ is isolated, then } \text{Rob}_T(\nu, L) \text{ is regular.} \]

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: \textit{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 5th 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 $a \mid b \mid w$ denotes a replacement of the letter $a$ by $b$ with a cost $w$. For convenience $T_2$ uses an $\epsilon$ 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 $g_4, g_4, g_5, g_5$ versus $\sigma_2$: fault $g_3, g_5, g_3, g_5, g_4$. The input $\sigma_1$ induces the run $s, s_0, t_1, t_2, t_3, r_4, r_5$ whereas the input $\sigma_2$ induces the run $s, s_0, s_1, s_2, s_3, s_4, r_5$. Both $\sigma_1, \sigma_2$ 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 $\sigma$ can be converted to $\sigma'$ with a notion of cost for the conversion. In this example we consider two notions of cost: the $\text{DSum}$-cost, and the $\text{Mean}$-cost. These costs now allow us to compare $\sigma_1$ versus $\sigma_2$. For instance, under both notions we will discover that $\sigma_1$ is much more robust than $\sigma_2$. The robustness of $\sigma_1$ under both cost models is $\infty$ since any change to $\sigma_1$ under the transducer continues to satisfy the desired property. On the other hand $\sigma_2$ 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...
Running times and robustness values computed for various input strings (the first letter weak is common to all the strings and is omitted). All timings are measured in seconds, $\epsilon$ denotes time < 0.01 seconds.

<table>
<thead>
<tr>
<th>String</th>
<th>$T_0$ Disc.</th>
<th>$T_0$ Avg.</th>
<th>$T_1$ Disc.</th>
<th>$T_1$ Avg.</th>
<th>$T_2$ Disc.</th>
<th>$T_2$ Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_2$</td>
<td>2$^{-5}$</td>
<td>1$^{-2}$</td>
<td>$2^{-5}$</td>
<td>1$^{-8}$</td>
<td>$2^{-7}$</td>
<td>1$^{-7}$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_4$</td>
<td>0</td>
<td>0</td>
<td>7.45e$^{-9}$</td>
<td>0.035</td>
<td>7.45e$^{-9}$</td>
<td>0.035</td>
</tr>
<tr>
<td>$s_5$</td>
<td>3.7e$^{-9}$</td>
<td>0.019</td>
<td>3.73e$^{-9}$</td>
<td>0.019</td>
<td>6.52e$^{-9}$</td>
<td>0.056</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>$T_0$ Disc.</th>
<th>$T_0$ Avg.</th>
<th>$T_1$ Disc.</th>
<th>$T_1$ Avg.</th>
<th>$T_2$ Disc.</th>
<th>$T_2$ Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_6$</td>
<td>$2^{-3}$</td>
<td>1$^{-2}$</td>
<td>$2^{-3}$</td>
<td>1$^{-8}$</td>
<td>$2^{-7}$</td>
<td>1$^{-7}$</td>
</tr>
<tr>
<td>$s_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_8$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$s_9$</td>
<td>3.7e$^{-9}$</td>
<td>0.019</td>
<td>3.73e$^{-9}$</td>
<td>0.019</td>
<td>6.52e$^{-9}$</td>
<td>0.056</td>
</tr>
</tbody>
</table>

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? 5 (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.
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 \( \Sigma \) 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) \( \text{NOT CALIB} \) denoting that no calibration has happened, (b) \( \text{CALIB} \): denoting a calibration event in the past, (c) \( \text{DROPOUTNC} \): a sensor drops out under the non calibrated mode and (d) \( \text{DROPOUTC} \): 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 \( \text{cost}(lb,ub,lb',ub') \). 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 \( \Sigma \) as defined above. The prolonged hypoglycemia property can be written as a regular expression: \( \Sigma^*(a_{40,50}+a_{50,60}+a_{60,70})^{360}\Sigma^* \) 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 \( \nu \) such that \( w \) can be rewritten by \( T \) at mean cost \( \nu \) 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 $\infty$: this signifies that no possible translation as specified by the transducer can cause the property to hold.

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.

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.

### Table 2

<table>
<thead>
<tr>
<th>Prop.</th>
<th>Total Time</th>
<th>Threshold Values synthesized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0 (0.0, 1]</td>
</tr>
<tr>
<td>P1</td>
<td>4hr10m31s</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>2hr10m30s</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>2hr0m9s</td>
<td>0</td>
</tr>
</tbody>
</table>

### 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.

### 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 \textbf{Sum}, \textbf{DSum} and \textbf{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


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 $\text{Sum}$, computing the infimum amounts to computing a shortest path in a finite graph with non-negative weights. Any $\text{PTime}$ 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 $\text{Mean}$, we first note that the infimum is either realized by a simple path from $V_I$ to $V_F$ of minimal $\text{Mean}$ value, or it is equal to the minimal $\text{Mean}$ value among the simple cycles in the graph. Indeed, if $c$ is a cycle of $\text{Mean}$ $m$ which is smaller than the $\text{Mean}$ value of any path from $V_I$ to $V_F$ then the family of paths $p_k = p \cdot e^k \cdot s$, 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 $\lim_{k \to +\infty} \text{Mean}(p_k) = \text{Mean}(c)$ and $\text{Mean}(c)$ 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 $\text{PTime}$ 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 $\text{Mean}$ value of simple cycles in the trimmed graph can be computed in $\text{PTime}$ using the Karp algorithm [18]. It is easy to see that the infimum is feasible iff it equals the minimum $\text{Mean}$ value of simple paths.

We now turn to the $\text{DSum}$ 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 $\text{DSum}$ values $x_v$ of paths reaching the target $V_F$ from $v$, in $\text{PTime}$. According to Lemma 1 of [5], and similarly to the case of $\text{Mean}$, for all $v \in V_I$, the infimum $\text{DSum}$ 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 $p \cdot e^k \cdot s$. 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 $x_v = \lambda x_{v'} + \psi(v, v')$. 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 $\pi$ 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 $\pi'$ from $v'$ to $V_F$. By construction of $G'$ we have $x_v = \lambda x_{v'} + \psi(v, v')$. 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 $x_v > \lambda x_{v'} + \psi(v, v')$. This contradicts the optimality of $x_v$, as $\pi$ witnesses a better discounted value from $v$ to $V_F$. Assume $x_v < \lambda x_{v'} + \psi(v, v')$, then since $\pi$ realises $x_v$, we have $x_v = \psi(v, v') + \text{DSum}(\pi')$. It implies $\text{DSum}(\pi') < x_{v'}$. This contradicts...
the minimalcity of \( x_\nu \), as then \( \pi' \) witnesses a better value for paths from \( \nu' \) to \( V_F \). Hence \( x_\nu = \lambda x_\nu + W(\nu, \nu') \) and \( (\nu, \nu') \) is an edge of \( G' \). By induction on the length of \( \pi \), we can also conclude that \( \pi' \) is a path of \( G' \) and then \( \pi \) is a path of \( G' \) from \( \nu \) to \( V_F \). □

**Proof of Lemma 14**

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

\[
\text{Rob}_T(\nu,L) = \{w_1 | \exists w_2 : \text{Sum}_T(w_1,w_2) < \nu \land w_2 \not\in L\}
\]

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 \( \overline{A} \). 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:

\[
\overline{\text{Rob}_T(\nu,L)} = \{w_1 | \exists w_2 : \text{Sum}_{T \otimes \overline{A}}(w_1,w_2) < \nu\}
\]

Now, by definition of \( \text{Sum}_{T \otimes \overline{A}}(w_1,w_2) \) we have \( w_1 \in \overline{\text{Rob}_T(\nu,L)} \) iff there exists a word \( w_2 \) and an accepting run \( r \) over \( (w_1,w_2) \) such that \( \text{Sum}(r) < \nu \). Therefore, we can project \( T \otimes \overline{A} \) on its input dimension (thus, we just ignore the outputs) and obtain a \( \text{Sum} \)-automaton that we call \( U \) such that \( \text{Rob}_T(\nu,L) = \{w_1 | U(w_1) < \nu \} \), 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: \( \overline{\text{Rob}_T(\nu,L)} = \{w_1 | U(w_1) \geq \nu\} \). 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 \( \text{DSum}(rr') = \text{DSum}(r) + \lambda^n \text{DSum}(r') \). We have \( \text{DSum}(r') \leq \sum_{i=0}^{\infty} \lambda^i \mu = \mu(1-\lambda)^{-1} \) where \( \mu \) 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 \( \nu \) is \( \epsilon \)-isolated, we have two cases:

i. If \( \text{DSum}(rr') \leq \nu - \epsilon \) then \( \text{DSum}(r) \leq \nu - \epsilon \) since \( \text{DSum}(r) \leq \text{DSum}(rr') \) by non-negativity of the weights of \( T \)

ii. If \( \text{DSum}(rr') \geq \nu + \epsilon \) then \( \text{DSum}(r) \geq \nu + \epsilon - \lambda^n \text{DSum}(r') \). Moreover \( \lambda^n \text{DSum}(r') \leq B_n \leq B_{n^*} \leq \epsilon/2 \) by construction. So \( -\lambda^n \text{DSum}(r') \geq -\epsilon/2 \) which implies \( \text{DSum}(r) \geq \nu + \epsilon/2 \).

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

**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 \( \text{DSum}(r) \leq \nu \), \( w_2 \not\in L \) and \( r \) is accepting. Not that necessarily,
\[ w \not\in \text{Rob}_T(\nu, L). \]  

The run \( r \) is called \textit{dangerous} if \( |r| \geq n \) and \( \text{DSum}(r) \leq \nu - B_n \). 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 \( \text{DSum}(rr') \) smaller than \( \nu \). 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 \( \text{Runs}^{\leq n} \) be the runs of \( T \) of length at most \( n \), and \( Q \) its set of states. We assume that for all \((w_1, w_2) \in R_T \), \( w_2 \not\in L \) 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 \( \text{NFA} \ A_n \). Its set of states is \( \text{Runs}^{\leq n} \cup Q \). Its transitions are defined as follows: for all \( T \)-runs \( r \) of length \( n - 1 \) at most ending in some state \( q \), for all \( \sigma \in \Sigma_f \), if there exists a transition \( t \) of \( T \) from state \( q \) on reading \( \sigma \), 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 \( \varepsilon \)-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 \( \sigma \) in \( A_n \) whenever there is a transition from \( q \) to \( q' \) on input \( \sigma \) in \( T \). Accepting states are bad runs of \( \text{Runs}^{\leq n} \) 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 \( \rho \) 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 \( A_n \), \( \rho \) can be decomposed into two parts \( \rho_1, \rho_2 \) such that \( \rho_1 \in (\text{Runs}^{\leq n})^* \) and \( \rho_2 \in Q^* \) with an \( \varepsilon \)-transition from the last state of \( \rho_1 \) to the first of \( \rho_2 \). We consider two cases. If \( |\rho_2| = 0 \), then \( \rho = \rho_1 \) and by definition of \( A_{n+1} \), \( \rho \) is still an accepting run of \( A_{n+1} \). In the other case, there is a dangerous run \( r \) of \( T \) such that \( \rho_1 \) 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 \( r \rho_2 \) is dangerous at step \( n + 1 \), in the sense that \( |r \rho_2| = n + 1 \) and \( \text{DSum}(r \rho_2) \leq \nu - B_{n+1} \), by definition of \( B_{n+1} \) and the fact that \( \text{DSum}(r) \leq \nu - B_n \). So, we get that the sequence of states \( \rho_1, (r \rho_2), q_2...q_k \) is a run of \( A_{n+1} \) on \( w \) is accepting in \( A_{n+1} \) (note that \( r \rho_2 \) here is a state of \( A_{n+1} \) and there is an \( \varepsilon \)-transition from \( (r \rho_2) \) to \( q_2 \)), concluding the first part of the proof.

Now, suppose that \( \nu \) is \( \varepsilon \)-isolated for some \( \varepsilon \). Then, take \( n^* \) as given by Lemma 18 and let us show that \( \text{Rob}_T(\nu, L) \cap \text{dom}(T) \subseteq L(A_{n^*}) \) (the other inclusion has just been proved for all \( n \)). Let \( w \in \text{dom}(T) \) such that \( w \not\in \text{Rob}_T(\nu, L) \). There exists \((w_1, w_2) \in R_T \) and an accepting run \( r \) of \( T \) on it such that \( \text{DSum}(r) \leq \nu \) and \( w_2 \not\in L \). In other words, \( r \) is bad. If \( |r| \leq n^* \), then \( \rho = r[1]r[2]...r[n] \) is an accepting run of \( A_{n^*} \) on \( w \), and we are done. Now suppose that \( |r| > n^* \). Since \( \nu \) is \( \varepsilon \)-isolated, we have \( \text{DSum}(r) \leq \nu - \varepsilon \). By Lemma 18, we also get that \( \text{DSum}(r[n^*]) \leq \nu - \varepsilon \). By definition of \( n^* \) being the smallest integer such that \( B_{n^*} < \varepsilon / 2 \), we get \( \text{DSum}(r[n^*]) \leq \nu - B_{n^*} \), hence \( r[n^*] \) is dangerous. We can conclude since then \( r[1]r[2]...r[n^*]r[n^*]r[n^*+1]...r[n] \) is an accepting run of \( A_n^* \) on \( w \).