Lipschitz Robustness of Timed I/O Systems*

Thomas A. Henzinger[1], Jan Otop[1,2] and Roopsha Samanta[1]

[1] IST Austria
[2] University of Wrocław

Abstract. We present the first study of robustness of systems that are both timed as well as reactive (I/O). We study the behavior of such timed I/O systems in the presence of *uncertain inputs* and formalize their robustness using the analytic notion of Lipschitz continuity: a timed I/O system is K-(Lipschitz) robust if the perturbation in its output is at most K times the perturbation in its input. We quantify input and output perturbation using *similarity functions* over timed words such as the timed version of the Manhattan distance and the Skorokhod distance. We consider two models of timed I/O systems — timed transducers and asynchronous sequential circuits. We show that K-robustness of timed transducers can be decided in polynomial space under certain conditions. For asynchronous sequential circuits, we reduce K-robustness w.r.t. timed Manhattan distances to K-robustness of discrete letter-to-letter transducers and show PSPACE-completeness of the problem.

1 Introduction

Real-time systems operating in physical environments, i.e., timed I/O systems, are increasingly commonplace today. An inherent problem faced by such computational systems is *input uncertainty* caused by sensor inaccuracies, imprecise environment assumptions etc. This means that the input data may be noisy and/or may have timing errors. In such scenarios, it is not enough for a system to be functionally correct. It is also desirable that the system be *continuous* or *robust*, i.e., the system behavior degrade smoothly in the presence of input disturbances [11]. We illustrate this property with two examples of timed I/O systems.

Example 1. Consider two timed I/O systems which process a sequence of ticks and calibrate the intervals between the ticks (see Fig. 1). In particular, the goal is to track if an interval is greater than some given Δ. The first timed I/O system $\mathcal{T}$ is an *offline* processor: upon arrival of each tick, $\mathcal{T}$ waits till the next tick, and outputs $\top$ if the interval is less than or equal to Δ and $\bot$ otherwise. The second timed I/O system $\mathcal{T}'$ is an *online* processor: $\mathcal{T}'$ starts generating $\top$ immediately

* This research was supported in part by the European Research Council (ERC) under grant 267989 (QUAREM), by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award), and by the National Science Centre (NCN), Poland under grant 2014/15/D/ST6/04543.

upon arrival of each tick, and switches its output to $\perp$ after Δ time, until the arrival of the next tick.

Consider two periodic tick sequences: i_1 and i_2 as shown in Fig. 1. The duration between ticks in i_1, i_2 is Δ, $\Delta+\epsilon$, respectively. Hence, i_2 can be viewed as a timing distortion of i_1. While the output o_1 of $\mathcal{T}$ on i_1 is a constant sequence of $\top$, the output o_2 of $\mathcal{T}$ on i_2 consists of $\perp$ entirely. Thus, a small timing perturbation in the input of $\mathcal{T}$ can cause a large perturbation in its output. On the other hand, a small timing perturbation in the input of $\mathcal{T}'$ only causes a proportionally small perturbation in its output. Indeed, while the output o'_1 of $\mathcal{T}'$ on i_1 is also a constant sequence of $\top$, the output o'_2 of $\mathcal{T}'$ on i_2 is a sequence of $\top$, with periodic $\perp$ intervals of ϵ-duration. Thus, the behaviour of $\mathcal{T}$ is more robust to small input timing distortions than the behaviour of $\mathcal{T}'$.

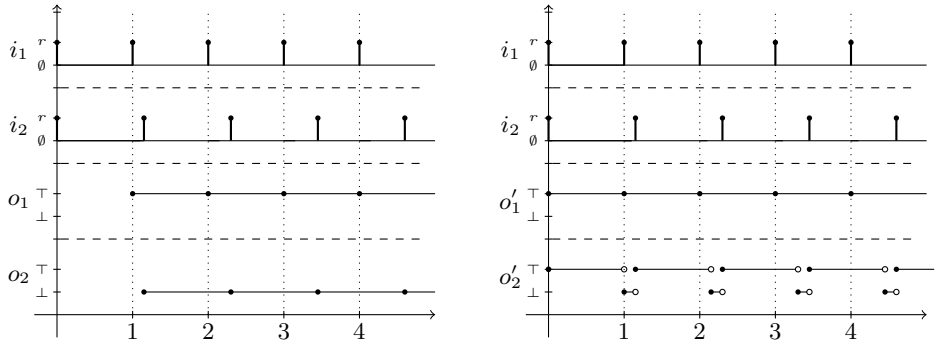


Fig. 1: System behaviour under timing distortion

Example 2. Consider two asynchronous sequential circuits $\mathcal{C}$ and $\mathcal{C}'$ shown in Fig. 2. For each circuit, the input is i, the output is $i \vee y$ and the value of variable y at time t equals the value of variable z at time $t-1$. In circuit $\mathcal{C}$, variable z equals $i \vee y$ and in circuit $\mathcal{C}'$, variable z equals i. Initially y is set to 0.

Consider inputs i_1 and i_2 such that i_1 is constantly 0, and i_2 is 1 in the interval $[0, \epsilon)$ and 0 otherwise (see Fig. 2). Thus, i_2 can be viewed as representing a transient fault in i_1. The outputs of both $\mathcal{C}$ and $\mathcal{C}'$ for i_1 are constantly 0. For i_2, $\mathcal{C}$ produces a periodic sequence that equals 1 exactly in the intervals $[0, \epsilon), [1, 1+\epsilon), [2, 2+\epsilon) \ldots$, whereas $\mathcal{C}'$ produces an output that equals 1 only in the intervals $[0, \epsilon)$ and $[1, 1+\epsilon]$. Thus, the effect of a small input perturbation propagates forever in the output of $\mathcal{C}$. On the other hand, the effect of a small input perturbation is limited to a bounded time in the output of $\mathcal{C}'$. The behaviour of $\mathcal{C}$ is more robust to transient faults than the behaviour of $\mathcal{C}'$.

We present the first study of robustness of systems that are both timed as well as reactive (I/O). We formalize robustness of timed I/O systems as Lipschitz

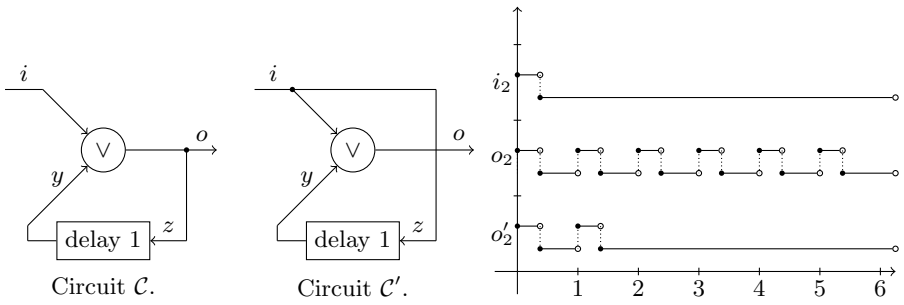


Fig. 2: System behaviour under transient fault

continuity [18, 19, 12]. A function is Lipschitz-continuous if its output changes proportionally to every change in the input. Given a constant K and *similarity functions* d_Σ, d_Γ for computing the input, output perturbation, respectively, a timed I/O system $\mathcal{T}$ is defined to be K-Lipschitz robust (or simply, K-robust) w.r.t. d_Σ, d_Γ if for all timed words w, v in the domain of $\mathcal{T}$ with finite $d_\Sigma(w, v)$, $d_\Gamma(\mathcal{T}(w), \mathcal{T}(v)) \leq K d_\Sigma(w, v)$.

In this work, we focus on K-robustness of two models of timed I/O systems — timed transducers (Ex. 1) and asynchronous sequential circuits (ASCs) (Ex. 2). We define a timed transducer as a timed automaton over an alphabet partitioned into an input alphabet d_Σ and an output alphabet d_Γ. A timed transducer defines a transduction over timed words, or a *timed relation*. An ASC is composed of a combinational circuit (CC), *delay elements* and *feedback loops* (see, for instance, Fig. 2). An ASC also defines a timed relation. However, timed transducers and ASCs are expressively incomparable. A simple ASC that delays its inputs by 1 time unit is not expressible by timed transducers — intuitively, the timed transducer at time 1 would need to remember arbitrarily many timed events from the interval $[0, 1)$. Conversely, a simple timed transducer that outputs 1 if the duration between preceding input events is greater than 1, and 0 otherwise cannot be expressed by any ASC.

We show that K-robustness of timed transducers is undecidable is general, and decidable under certain conditions on similarity functions. The key idea behind decidability is a reduction of K-robustness of timed transducers to emptiness of weighted timed automata. In particular, our decidability results include the following:

1. K-robustness w.r.t. timed Manhattan distances is PSPACE-complete,
2. K-robustness w.r.t. accumulated delay distances is PSPACE-complete under practically-viable environment assumptions (e.g., minimum symbol persistence), and,
3. K-robustness is PSPACE-complete if the input perturbation is computed as a Skorokhod distance and the output perturbation is computed as a timed Manhattan distance.

We reduce K-robustness of ASCs w.r.t. timed Manhattan distances to K-robustness of discrete letter-to-letter transducers, and show that K-robustness of ASCs is PSPACE-complete. The reduction consists of two steps. First, we show that on inputs that are *step functions*, ASCs behave like discrete letter-to-letter transducers. Second, we show that if an ASC is not K-robust w.r.t. timed Manhattan distances, there exists a witness consisting of a pair of inputs that are step functions.

The paper is organized as follows. We first recall necessary formalisms (Sec. 2) and present our models of timed I/O systems (Sec. 3). We formalize our notion of robustness for such systems (Sec. 4) and define the similarity functions of interest (Sec. 5). We then present our results on robustness analysis of timed transducers (Sec. 6) and ASCs (Sec. 7) w.r.t. various similarity functions.

Related work. Robustness of systems has been studied in different contexts such as robust control [13], timed automata [10, 5], discrete transducers [18, 19, 12] and sequential circuits [9]. However, none of these results are directly applicable to robustness of timed I/O systems. There are two main reasons. First, we are interested in robustness w.r.t. *input* perturbation. Second, timed I/O systems exhibit both discrete and continuous behavior. Robust control typically involves reasoning about continuous state-spaces and focuses on designing controllers that function properly in the presence of perturbation in various internal parameters of a system's model. The study of robustness of timed automata focuses on the design of models whose language is robust to infinitesimal timing perturbation (e.g. clock drifts) and does not focus on quantifying the effect of input perturbation on the output. Robustness analysis of finite-state transducers is limited to purely discrete systems and data. In [9], the authors study the robustness of synchronous sequential circuits modeled as discrete Mealy machines. Their notion of robustness bounds the persistence of the effect of a sporadic disturbance and is also limited to discrete data.

In other related work [16, 6, 3], the authors develop different notions of robustness for discrete reactive systems with ω-regular specifications interacting with uncertain environments. There has also been foundational work on continuity and robustness analysis of software programs manipulating numbers [17, 7, 8].

2 Preliminaries

2.1 Timed automata

We briefly present basic notions regarding timed automata. We refer the reader to [2] for a comprehensive survey on timed automata.

Timed words. Let $\mathbb{R}^+$, $\mathbb{Q}^+$ denote the set of all *nonnegative* real numbers, rational numbers, respectively. A (finite or infinite) *timed word* over an alphabet Σ is a word over $(\Sigma, \mathbb{R}^+)$: $(a_0, t_0)(a_1, t_1)\ldots$ such that $t_0, t_1, \ldots$ is a weakly increasing sequence. A pair (a, t) is referred to as *an event*. We denote by $\mathcal{TL}(\Sigma)$ the set of all timed words over Σ. For a timed word $w = (a_0, t_0)(a_1, t_1)\ldots$, we define untimed$(w) = a_0 a_1 \ldots$ as the projection of w on the Σ component.

Disjoint union of timed words. Let w_1, w_2 be timed words over the alphabet Σ. We define *the disjoint union* of w_1 and w_2, denoted $w_1 \oplus w_2$, as the union of events of w_1 and w_2, annotated with the index of the word (w_1 or w_2) it belongs to. E.g. $\langle a, 0.4\rangle\langle b, 2.1\rangle \oplus \langle b, 0.3\rangle\langle b, 0.4\rangle = \langle (b, 2), 0.3\rangle\langle (a, 1), 0.4\rangle\langle (b, 2), 0.4\rangle\langle (b, 1), 2.1\rangle$. The word $w_1 \oplus w_2$ is a timed word over the alphabet $\Sigma \times \{1, 2\}$.

Clocks. Let X be a set of clocks. A *clock constraint* is a conjunction of terms of the form $x \otimes c$, where $x \in X$, $c \in \mathbb{Q}^+$ and $\otimes \in \{<, \leq, =, \geq, >\}$. Let $B(X)$ denote the set of clock constraints. A *clock valuation* ν is a mapping $\nu : X \mapsto \mathbb{R}^+$.

Timed automata. A *timed automaton* $\mathcal{A}$ is a tuple $(\Sigma, L, l_0, X, \delta, F)$ where Σ is the alphabet of $\mathcal{A}$, L is a set of locations, $l_0 \in L$ is an initial location, X is a set of clocks, $\delta \subseteq L \times \Sigma \times B(X) \times 2^X \times L$ is a switch relation and $F \subseteq L$ is a set of accepting locations.

Semantics of timed automata. The semantics of a timed automaton $\mathcal{A}$ is defined using an infinite-state transition system $\mathrm{Pre}_{\mathcal{A}}$ over the alphabet $(\Sigma \cup \{\epsilon\}) \times \mathbb{R}^+$. A *state* q of $\mathrm{Pre}_{\mathcal{A}}$ is a pair (l, ν) consisting of a location $l \in L$ and a clock valuation ν. A state $q = (l, \nu)$ satisfies a clock constraint g, denoted $q \models g$, if the formula obtained from g by substituting clocks from X by their valuations in ν is true. There are two kinds of transitions in $\mathrm{Pre}_{\mathcal{A}}$: (i) *elapse of time*: $(l, \nu) \rightarrow^{\tau} (l, \nu')$ iff for every $x \in X$, $\nu'(x) = \nu(x) + \tau$ and (ii) *location switch*: $(l, \nu) \rightarrow^{a} (l', \nu')$ iff there is a switch of $\mathcal{A}$, (l, a, g, γ, l'), such that $(l, \nu) \models g$, and for each $x \in X$, $\nu'(x) = 0$ if $x \in \gamma$ and $\nu'(x) = \nu(x)$ otherwise. Consecutive elapses of time can be merged, therefore we assume that an elapse of time is followed by a location switch. The initial state of $\mathrm{Pre}_{\mathcal{A}}$ is the state (l_0, ν) where for each $x \in X$, $\nu(x) = 0$. The accepting states of $\mathrm{Pre}_{\mathcal{A}}$ are all states of the form $\langle l, \nu\rangle$, where $l \in F$. A *run* of $\mathcal{A}$ over a timed word $w = (a_0, t_0)(a_1, t_1)\ldots(a_k, t_k)$ is the sequence: $q_0 \rightarrow^{t_0} q_1 \rightarrow^{a_0} q_2 \rightarrow^{t_1 - t_0} q_3 \rightarrow^{a_1} \ldots \rightarrow^{a_k} q_{2k+2}$, where q_0 is the initial state of $\mathrm{Pre}_{\mathcal{A}}$. The run is accepting if q_{2k+2} is an accepting state. The set of accepting runs of $\mathcal{A}$ is denoted $[\mathcal{A}]$. We say a timed word w is accepted by $\mathcal{A}$ if there is a run over w in $[\mathcal{A}]$.

The *emptiness problem* for timed automata is as follows: given a timed automaton $\mathcal{A}$, decide whether $[\mathcal{A}]$ is nonempty. The emptiness problem is also referred to as the *reachability* problem as it is equivalent to reachability of an accepting state in $\mathrm{Pre}_{\mathcal{A}}$.

2.2 Weighted timed automata

A *weighted timed automaton* (WTA) is a timed automaton augmented by a function $C : L \cup \delta \mapsto \mathbb{Q}$ that associates *weights* with the locations and switches of the timed automaton. The *value* of a run $(l_0, \nu_0) \rightarrow^{\tau_0} (l_0, \nu_1) \rightarrow^{a_0} \ldots \rightarrow^{a_k} (l_k, \nu_{2k+2})$ is given by

$$\sum_{i=0}^{k} C(l_i)\tau_i + \sum_{i=0}^{k} C(e_i)$$

where e_i is the switch taken in the transition $(l_i, \nu_{2i+1}) \rightarrow^{a_i} (l_{i+1}, \nu_{2i+2})$. The value of a timed word w assigned by a WTA $\mathcal{A}$, denoted $\mathcal{L}_{\mathcal{A}}(w)$, is defined as the infimum over values of all accepting runs of $\mathcal{A}$ on w.

The *quantitative emptiness* problem for WTA is as follows: given a WTA $\mathcal{A}$ and $\lambda \in \mathbb{Q}$, decide whether $\mathcal{A}$ has an accepting run with value smaller than λ.

Theorem 3. *[4] The quantitative emptiness problem for WTA is* PSPACE*-complete.*

A WTA $\mathcal{A}$ is *functional* if for every timed word w, all accepting runs of $\mathcal{A}$ on w have the same value.

2.3 Discrete transducers

Discrete transducers. A discrete transducer $\mathcal{T}$ is a tuple $(\Sigma, \Gamma, Q, Q_0, E, F)$ where Σ is the input alphabet, Γ is the output alphabet, Q is a finite nonempty set of states, $Q_0 \subseteq Q$ is a set of initial states, $E \subseteq Q \times \Sigma \times \Gamma^* \times Q$ is a set of transitions, and F is a set of accepting states.

Semantics of discrete transducers. A run γ of $\mathcal{T}$ on an input word $s = s[1]s[2]\ldots s[n]$ is defined in terms of the sequence: (q_0, u_1), (q_1, u_2), $\ldots$, (q_{n-1}, u_n), (q_n, ϕ) where $q_0 \in Q_0$ and for each $i \in \{1, 2, \ldots, n\}$, $(q_{i-1}, s[i], u_i, q_i) \in E$. A run $(q_0, u_1), \ldots (q_{n-1}, u_n), (q_n, \phi)$ is *accepting* if $q_n \in F$. The output of $\mathcal{T}$ along a run is the word $u = u_1 \cdot u_2 \cdot \ldots \cdot u_n$ if the run is accepting, and is undefined otherwise. The transduction computed by a discrete transducer $\mathcal{T}$ is the relation $[\![\mathcal{T}]\!] \subseteq \Sigma^\omega \times \Gamma^\omega$ (resp., $[\![\mathcal{T}]\!] \subseteq \Sigma^* \times \Gamma^*$), where $(s, u) \in [\![\mathcal{T}]\!]$ iff there is an accepting run of $\mathcal{T}$ on s with u as the output along that run.

Types of discrete transducers. A discrete transducer $\mathcal{T}$ is called *functional* if the relation $[\![\mathcal{T}]\!]$ is a function. In this case, we use $[\![\mathcal{T}]\!](s)$ to denote the unique output word generated along any accepting run of $\mathcal{T}$ on input word s. A discrete transducer is a letter-to-letter transducer if in every transition (q, a, u, a') we have $|u| = 1$.

3 Models of Timed I/O Systems

In this section, we present two models of timed I/O systems whose robustness will be studied in the following sections. The reason for studying these models separately is that timed transducers and ASCs are expressively incomparable (as explained in the introduction).

3.1 Timed transducers

In the following, we define timed transducers, which extend classical discrete transducers.

Definition 4 (Timed transducer). *A timed transducer $\mathcal{T}$ is a timed automaton over an alphabet partitioned into an input alphabet Σ and an output alphabet Γ.*

Semantics of timed transducers. Given a timed transducer $\mathcal{T}$, we define a relation $[\![\mathcal{T}]\!] \subseteq \mathcal{TL}(\Sigma) \times \mathcal{TL}(\Gamma)$ by $[\![\mathcal{T}]\!] = \{(w, v) : w \in \mathcal{TL}(\Sigma), v \in \mathcal{TL}(\Gamma), \mathcal{T}$ accepts $w \oplus v\}$. We say that $v \in \mathcal{TL}(\Gamma)$ is an *output* of $\mathcal{T}$ on $w \in \mathcal{TL}(\Sigma)$ if $(w, v) \in [\![\mathcal{T}]\!]$.

Remark 5. Our model of timed transducers is similar to *timed automata with inputs and outputs* presented in [14]. The main difference is the absence of *deadlines* in our automaton model.

In the following proposition, we relate the discrete part of the relation defined by a timed transducer to the relation defined by a discrete transducer. For a timed relation $R \subseteq \mathcal{TL}(\Sigma) \times \mathcal{TL}(\Gamma)$, let untimed$(R)$ denote $\{(\text{untimed}(w), \text{untimed}(v)) : (w, v) \in R\}$.

Proposition 6. *(i): For every timed transducer $\mathcal{T}$ that has no cycles labeled by Γ, there exists a (nondeterministic) discrete transducer $\mathcal{T}^d$ of exponential size in size$(\mathcal{T})$ such that untimed$([\![\mathcal{T}]\!])$ and $[\![\mathcal{T}^d]\!]$ coincide. (ii): For every discrete transducer $\mathcal{T}^d$, there exists a timed transducer $\mathcal{T}$ that has no cycles labeled by Γ such that untimed$([\![\mathcal{T}]\!])$ and $[\![\mathcal{T}^d]\!]$ coincide.*

Functionality. A transducer is *timed-functional* iff $[\![\mathcal{T}]\!]$ is a function, i.e., for all $w \in \mathcal{TL}(\Sigma)$ and $v_1, v_2 \in \mathcal{TL}(\Gamma)$, if both $(w, v_1) \in [\![\mathcal{T}]\!]$ and $(w, v_2) \in [\![\mathcal{T}]\!]$, then $v_1 = v_2$. For a timed-functional transducer $\mathcal{T}$, we use $[\![\mathcal{T}]\!](w)$ to denote the unique output of $\mathcal{T}$ on w.

Proposition 7. *Deciding timed functionality of a timed transducer is* PSPACE*-complete.*

Observe that a timed transducer does not have to be timed-functional, even if it is deterministic when viewed as a timed automaton. Indeed, a trivial timed automaton that accepts every word over the alphabet $\Sigma \cup \Gamma$ is deterministic and is a timed transducer. However, it is not functional.

In Proposition 8, we present a sufficient condition for timed-functionality which can be checked in polynomial time. We further identify a class of transducers for which this condition is also necessary. A switch in a timed automaton is *rigid* iff it is guarded by a constraint containing equality. A location l in a timed automaton is unambiguous if for any pair of outgoing switches, their constraints g_1 and g_2 are strongly inconsistent, i.e., for all $x_1, \ldots, x_n, t \in \mathbb{R}^+$, $g_1(x_1, \ldots, x_n) \wedge g_2(x_1+t, \ldots, x_n+t)$ is false. A transducer is *safe* if every location with outgoing Σ switches is accepting.

Proposition 8. *(1) A deterministic timed transducer in which all switches labeled by Γ are (a) rigid, and (b) all locations with outgoing switches labeled by Γ are unambiguous, is functional. (2) Every function defined by a deterministic safe timed transducer is also defined by a deterministic safe timed transducer satisfying (a) and (b) from (1).*

3.2 Asynchronous Sequential Circuits

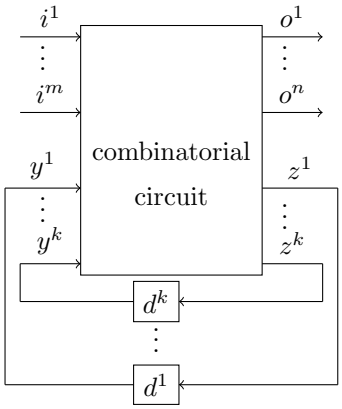


Fig. 3: A generic ASC.

The second model of timed I/O systems that we consider is an asynchronous sequential circuit (ASC). A generic ASC is shown in Fig. 3 and some example ASC's are shown in Fig. 2.

An ASC is an I/O system composed of a combinational circuit (CC) and *memory devices*, or *delay elements*. A CC is simply a Boolean logic circuit that computes Boolean functions of its inputs. A CC is *memoryless*: the values of the circuit's output variables at time instant t are functions of the values of the circuit's input variables at the same time instant t. A delay element is always labeled with some $d > 0$. The output of a d-delay element at time t equals its input at time $t - d$. We consider delays that are natural numbers (see Remark 11). ASC's may contain cycles, or *feedback loops*. Each such cycle is required to contain at least one delay element. Due to the presence of delay elements and feedback loops, an ASC has memory: the outputs of an ASC at time instant t are in general functions of its inputs at time instant t as well as at time instants $t' < t$. The inputs of the delay elements of an ASC are called *excitation variables*. The outputs of the delay elements of an ASC are called *secondary variables*. The relationships between input, output, excitation and secondary variables of an ASC are graphically represented in Fig. 3 and formally defined below.

Definition 9. *Let $\mathcal{C}$ be an ASC with input variables $\mathcal{I} = \{i^1, \ldots, i^m\}$, output variables $\mathcal{O} = \{o^1, \ldots, o^n\}$, excitation variables $\mathcal{Z} = \{z^1, \ldots, z^k\}$, secondary variables $\mathcal{Y} = \{y^1, \ldots, y^k\}$ and delay elements $\Delta = \{d^1, \ldots, d^k\}$. Let $i(t)$ and $\mathcal{I}(t)$ denote the values of input i and all inputs $\mathcal{I}$ at time t, respectively. One can*

similarly define $o(t)$, $\mathcal{Y}(t)$ etc. We have the following:

$$\forall j \in [1,k] : y^j(t) = \begin{cases} 0 & \text{if } t = [0, d^j) \\ z^j(t - d^j) & \text{if } t \geq d^j \end{cases}$$
$$\forall j \in [1,k] : z^j(t) = f^j(x^1(t), \ldots, x^m(t), y^1(t), \ldots, y^k(t))$$
$$\forall j \in [1,n] : o^j(t) = g^j(x^1(t), \ldots, x^m(t), y^1(t), \ldots, y^k(t)).$$

Here, $f^1, \ldots, f^k$ and $g^1, \ldots, g^n$ are Boolean functions. The input alphabet *of ASC $\mathcal{C}$, denoted Σ, is given by $\{0,1\}^m$. The* output alphabet *of $\mathcal{C}$, denoted Γ, is given by $\{0,1\}^n$. The ASC $\mathcal{C}$ defines a transduction $[\![\mathcal{C}]\!] \subseteq \mathcal{TL}(\Sigma) \times \mathcal{TL}(\Gamma)$ such that $[\![\mathcal{C}]\!]$ is a total function. Thus, the domain of $\mathcal{C}$ is given by* $\mathrm{dom}(\mathcal{C}) = \mathcal{TL}(\Sigma)$. *We use $[\![\mathcal{C}]\!](w)$ to denote the unique output of $\mathcal{C}$ on w.*

Remark 10. Our model of ASCs shares some similarities (such as delays) with models of discrete event systems ([20]). The main difference is that, in addition to timing relations, ASCs also express functional relations between inputs and outputs.

Remark 11 (Time stretching for ASCs). Let $s > 0$ and let $\lambda_s : R \mapsto R$ be *time stretching* defined for every $t \in R$ as $\lambda_s(t) = s \cdot t$. Consider an ASC with rational delays $\mathcal{C}$ and an ASC with rational delays $\mathcal{C}_s$ obtained from $\mathcal{C}$ by multiplying all delays by s. Observe that for every input $i(t)$ and the corresponding output $o(t)$ of $\mathcal{C}$, the signal $o(\lambda_s(t))$ is the output of $\mathcal{C}_s$ on input $i(\lambda_s(t))$. Thus, ASCs with rational delays do not introduce any behaviours that are significant for robustness over ASCs with integer delays.

4 Problem Statement

Similarity functions. In our work, we use similarity functions to measure the similarity between timed words. Let S be a set of timed words and let $\mathbb{R}^\infty$ denote the set $\mathbb{R} \cup \{\infty\}$. A similarity function $d : S \times S \to \mathbb{R}^\infty$ is a function with the properties: $\forall x, y \in S$: (1) $d(x,y) \geq 0$ and (2) $d(x,y) = d(y,x)$. A similarity function d is also a distance (function or metric) if it satisfies the additional properties: $\forall x, y, z \in S$: (3) $d(x,y) = 0$ iff $x = y$ and (4) $d(x,z) \leq d(x,y) + d(y,z)$. We emphasize that in our work we do not need to restrict similarity functions to be distances.

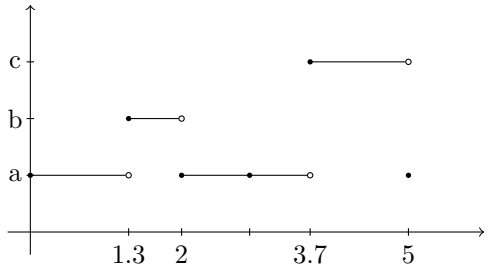
In this paper, we are interested in studying the *K-Lipschitz robustness* of timed-functional transducers and ASCs.

Definition 12 (K-Lipschitz Robustness of Timed I/O Systems). *Let $\mathcal{T}$ be a timed-functional transducer or an ASC with $[\![\mathcal{T}]\!] \subseteq \mathcal{TL}(\Sigma) \times \mathcal{TL}(\Gamma)$. Given a constant $K \in \mathbb{Q}$ with $K > 0$ and similarity functions $d_\Sigma : \mathcal{TL}(\Sigma) \times \mathcal{TL}(\Sigma) \to \mathbb{R}^\infty$ and $d_\Gamma : \mathcal{TL}(\Gamma) \times \mathcal{TL}(\Gamma) \to \mathbb{R}^\infty$, the timed I/O system $\mathcal{T}$ is called K-Lipschitz robust w.r.t. d_Σ, d_Γ if:*

$$\forall w, v \in \mathrm{dom}(\mathcal{T}) : \; d_\Sigma(w,v) < \infty \Rightarrow d_\Gamma([\![\mathcal{T}]\!](w), [\![\mathcal{T}]\!](v)) \leq K d_\Sigma(w,v).$$

5 Similarity Functions between Timed Words

Timed words as Càdlàg functions. Consider a timed word $w : (a_0, t_0)(a_1, t_1)\dots(a_k, t_k)$ over (Σ, I), where $I = [t_0, t_k]$ is an interval in $\mathbb{R}^+$. We define a Càdlàg function $w_C : I \mapsto \Sigma$ corresponding to w as follows: for each $j \in \{0, 1, \dots, k-1\}$, $w_C(t) = a_j$ if $t \in [t_j, t_{j+1})$, and $w_C(t_k) = a_k$. We define a timed word $\text{timed}(w_C) = (\alpha_0, \delta_0)(\alpha_1, \delta_1)\dots(\alpha_n, \delta_n)$ corresponding to the Càdlàg function w_C such that: for each $j \in \{0, 1, \dots, n\}$, $\alpha_j = w_C(\delta_j)$ and $\delta_j \in \{\delta_0, \dots, \delta_n\}$ iff w_C *changes value* at δ_j. The timed word $\text{timed}(w_C)$ can be interpreted as a *stuttering-free* version of the timed word w. The intervals $[\delta_0, \delta_1), [\delta_1, \delta_2), \dots, [\delta_{n-1}, \delta_n)$ are called *segments* of w.
Example. Let w be the timed word $(a, 0)(b, 1.3)(a, 2)(a, 2.9)(c, 3.7)(a, 5)$. Then w_C is given by the following Càdlàg function over the interval $[0, 5]$.



The timed word $\text{timed}(w_C) = (a, 0)(b, 1.3)(a, 2)(c, 3.7)(a, 5)$.

In what follows, let w, v be timed words over (Σ, I) with $I \subseteq \mathbb{R}^+$. Let w_C, v_C be Càdlàg functions over I, corresponding to w, v, as defined above. We present below several similarity functions between timed words, computed as the similarity function between their corresponding Càdlàg functions. We first present a similarity function between discrete words.
Generalized Manhattan distance. The *generalized Manhattan distance* over discrete words s, t is defined as: $d_M(s, t) = \sum_{i=1}^{max(|s|,|t|)} \texttt{diff}(s[i], t[i])$, where `diff` is a cost function that assigns costs for substituting letters. When $\texttt{diff}(a, b)$ is defined to be 1 for all a, b with $a \neq b$, and 0 otherwise, d_M is called the *Manhattan distance*.
Timed Manhattan distance. The *timed Manhattan distance* d_{TM} extends the generalized Manhattan distance by accumulating the pointwise distance, as defined by `diff`, between the Càdlàg functions corresponding to timed words. Given `diff` on Σ:

$$d_{TM}(w, v) = \int_I \texttt{diff}(w_C(x), v_C(x))dx.$$

Accumulated delay distance. The *accumulated delay distance* d_{AD} examines the timed words $\text{timed}(w_C)$ and $\text{timed}(v_C)$. If the projections of these timed words on their Σ components are equal, then the distance $d_{AD}(w, v)$

equals the sum of delays between the corresponding events; otherwise the distance is infinite. Let $\text{timed}(w_C) = (\alpha_0, \delta_0)(\alpha_1, \delta_1)\ldots(\alpha_n, \delta_n)$ and $\text{timed}(v_C) = (\beta_0, \tau_0)(\beta_1, \tau_1)\ldots(\beta_n, \tau_m)$.

$$d_{AD}(w, v) = \begin{cases} \sum_j |\delta_j - \tau_j| & \text{if untimed}(\text{timed}(w_C)) = \text{untimed}(\text{timed}(v_C)) \\ \infty & \text{otherwise.} \end{cases}$$

Skorokhod distance w.r.t. timed Manhattan distance. The *Skorokhod distance* d_S is a popular distance metric for continuous functions. Hence, it is also a natural choice for our Càdlàg functions. The Skorokhod distance permits *wiggling* of the function values as well as the timeline in order to *match up* the functions. The timeline wiggle is executed using continuous bijective functions, denoted λ, over the timeline. The first component of the Skorokhod distance measures the magnitude of the *timing distortion* resulting from a timeline wiggle λ. The second component of the Skorokhod distance measures the magnitude of the *function value mismatch* under λ. The Skorokhod distance is the least value obtained over all such timeline wiggles. The magnitudes of the timing distortion and function value mismatch can be computed and combined in different ways. In our work, the timing distortion is computed as the L_1 norm, the function value mismatch is computed as the timed Manhattan distance and the two are combined using addition. Let Λ be the set of all continuous bijections from the domain I of w_C and v_C onto itself.

$$d_S(w_C, v_C) = \inf_{\lambda \in \Lambda} \left(||\texttt{Id} - \lambda||_1 + d_{TM}(w_C, v_C \circ \lambda) \right),$$

where $\texttt{Id}$ is the identity function over I, $||.||_1$ is the L_1-norm over $\mathbb{R}^+$ and $\circ$ is the usual function composition operator.

We now present some helpful connections between the above distances. Let $d_{TM}^{=}$ denote a timed Manhattan distance with $\texttt{diff}$ given by: $\forall a, b \in \Sigma$, $\texttt{diff}(a, b) = 0$ if $a = b$ and $\texttt{diff}(a, b) = \infty$ otherwise. Let $D_{TM}^{\leq 1}$ denote a class of timed Manhattan distances, $d_{TM}^{\leq 1}$, with $\texttt{diff}$ satisfying: $\forall a, b \in \Sigma$, $\texttt{diff}(a, b) \leq 1$.

Proposition 13. *[Relations between distances] (i) The accumulated delay distance coincides with the Skorokhod distance w.r.t.* $d_{TM}^{=}$. *(ii) For any* $d_{TM}^{\leq 1} \in D_{TM}^{\leq 1}$, *the Skorokhod distance w.r.t.* $d_{TM}^{\leq 1}$ *coincides with* $d_{TM}^{\leq 1}$.

6 Robustness Analysis of Timed Transducers

To investigate K-robustness as a decision problem, one needs to have a finitary encoding of instances of the problem, in particular, of the similarity functions. We use weighted timed automata to represent similarity functions.

Timed-automatic similarity function. A timed similarity function d is *computed* by a WTA $\mathcal{A}$ iff for all $w, v \in \mathcal{TL}(\Sigma)$, $d(w, v) = \mathcal{L}_{\mathcal{A}}(w \oplus v)$. A timed

similarity function d computed by a WTA is called a *timed-automatic similarity function*.

Unfortunately, checking K-robustness of timed transducers w.r.t. timed-automatic similarity functions is undecidable. The undecidability result follows from a reduction from the universality problem for timed automata, which is undecidable [1].

Theorem 14. *K-robustness of timed transducers w.r.t. timed-automatic similarity functions is undecidable.*

K-robustness can, however, be decided using an automata-based, polynomial-space, sound (but incomplete) procedure: if the procedure certifies a transducer $\mathcal{T}$ to be K-robust, then $\mathcal{T}$ is indeed K-robust. This procedure becomes complete under additional assumptions.

Theorem 15. *(i) There exists a polynomial-space sound procedure that given timed-automatic similarity functions d_Σ, d_Γ and a timed transducer $\mathcal{T}$, decides K-robustness of $\mathcal{T}$ w.r.t. d_Σ, d_Γ.*

(ii) There exists a PSPACE*-complete procedure that given timed-automatic similarity functions d_Σ, d_Γ, with d_Γ computed by a functional WTA, and a timed transducer $\mathcal{T}$, decides K-robustness of $\mathcal{T}$ w.r.t. d_Σ, d_Γ.*

Proof Sketch. PSPACE-hardness in (ii) follows from a simple reduction from the emptiness problem for timed automata. To show containment in PSPACE, we construct an automaton $\mathcal{A}$ that accepts words that are counterexamples to K-robustness. More precisely, the automaton $\mathcal{A}$ accepts words $w \oplus v \oplus [\![\mathcal{T}]\!](w) \oplus [\![\mathcal{T}]\!](v)$ with value $K \cdot d_\Sigma(w, v) - d_\Gamma([\![\mathcal{T}]\!](w), [\![\mathcal{T}]\!](v))$. Therefore, an accepted word with value less than 0 corresponds to timed words w, v that form a counterexample for K-robustness of $\mathcal{T}$ w.r.t. d_Σ, d_Γ. The automaton $\mathcal{A}$ is a product automaton that includes a copy of the WTA computing d_Σ, with weights scaled by K, and a copy of the WTA computing d_Γ, with weights scaled by -1. Given words w, v, the value computed by the last WTA is smaller than $-d_\Gamma([\![\mathcal{T}]\!](w), [\![\mathcal{T}]\!](v))$ in general, and is exactly equal to $-d_\Gamma([\![\mathcal{T}]\!](w), [\![\mathcal{T}]\!](v))$ if the WTA is functional. It follows that our automata-theoretic procedure for checking K-robustness is sound in general and becomes complete when d_Γ is computed by a functional WTA. □

We now define several timed similarity functions that can be computed by functional and nondeterministic WTA.

Timed similarity functions computed by functional WTA. We show that the timed Manhattan and accumulated delay distances can be computed by functional WTA.

Lemma 16. *The timed Manhattan distance d_{TM} over timed words is computed by a functional WTA.*

To compute the timed Manhattan distance, the WTA simply tracks the value of `diff` between timed events using its weight function. The semantics

of WTA then imply that the value assigned by the automaton to a pair of timed words is precisely the timed Manhattan distance between them.

Lemma 17. *Let D, B be any nonnegative real numbers. The accumulated delay distance d_{AD} over timed words w, v such that:*
1. the duration of any segment in w_C, v_C is greater than D and
2. the delay $|\delta_j - \tau_i|$ between corresponding events in w_C, v_C is less than B,
is computed by a functional WTA.

The WTA tracks with its weight function the number of unmatched events. Again, the semantics of WTA imply that the value assigned by the automaton to a pair of timed words is precisely the accumulated delay distance. To make sure that every event is matched to the right event, i.e., the untimed parts are equal, the automaton implements a buffer to store the unmatched events. The assumptions on the minimal duration of events and the maximal delay between the corresponding events imply that the buffer's size is bounded.

Timed similarity functions computed by nondeterministic WTA. A (restricted) Skorokhod distance can be computed by a nondeterministic WTA. We first prove the following lemma characterizing an essential subset of the set Λ of all timing distortions.

Lemma 18. *[Skorokhod distance is realized by a piecewise linear function] Let w, v be timed words. Let η be the number of segments in v. For every $\epsilon > 0$, there exists a piecewise linear function λ consisting of η segments such that $|(||\mathtt{Id} - \lambda||_1 + d_{TM}(w_C, v_C \circ \lambda)) - d_S(w_C, v_C)| \leq \epsilon$.*

Observe that for piecewise linear functions λ, the value of $||\mathtt{Id}||_1 - \lambda$ coincides with the accumulated delay distance between v_C and $v_C \circ \lambda$. This fact, combined with Lemma 18, allows us to compute the Skorokhod distance using a WTA that non-deterministically guesses λ and computes the sum of the accumulated delay between v_C and $v_C \circ \lambda$ and the timed Manhattan distance between w_C and $v_C \circ \lambda$.

Lemma 19. *Let D, B be any nonnegative real numbers. The Skorokhod distance d_S over timed words w, v restricted to timeline wiggles λ such that:*
1. the duration of any segment in v_C, $v_C \circ \lambda$ is greater than D and
2. the delay $|\delta_j - \tau_i|$ between corresponding events in v_C, $v_C \circ \lambda$ is less than B,
is computed by a nondeterministic WTA.

Remark 20. Physical systems typically have a bounded rate at which they can generate/process data. Hence, bounding the minimum possible duration of timed symbols is not a severe restriction from the modeling perspective. Moreover, if an input is delayed arbitrarily, it makes little sense to constraint the system behavior. Hence, for robustness analysis, it is also reasonable to bound the maximum delay between corresponding events.

Summary of decidability results. We summarize the decidability results for timed transducers that follow from Theorem 15 and Lemmas 16, 17 and 19.

1. K-robustness is PSPACE-complete for timed Manhattan distances.
2. K-robustness is PSPACE-complete for accumulated delay distances (under environment assumptions from Lemma 17).
3. K-robustness is PSPACE-complete if the input perturbation is computed as a Skorokhod distance (under environment assumptions from Lemma 19) and the output perturbation is computed as a timed Manhattan distance.

7 Robustness Analysis of Asynchronous Sequential Circuits

In this section, we show that robustness of ASCs w.r.t. the timed Manhattan distances is PSPACE-complete. The decision procedure is by reduction to discrete letter-to-letter transducers. Our argument consists of two steps and relies on the use of *step functions* — Càdlàg functions that change values only at integer points. First, we show that on inputs that are step functions, *ASCs* behave like discrete letter-to-letter transducers. Second, we show that if an ASC is not K-robust w.r.t. the timed Manhattan distances, there exists a counterexample consisting of a pair of inputs that are step functions. Therefore, we can reduce K-robustness of ASCs to K-robustness of discrete letter-to-letter transducers, which can be solved employing techniques from [12].

ASCs transforming step functions. There is a natural correspondence between step functions $f : [0, T] \mapsto \{0, 1\}^k$ and words over the alphabet $\{0, 1\}^k$. The function f defines the word $word(f) = f(0)f(1) \dots f(T-1)$ and, conversely, a word $w \in (\{0, 1\}^k)^*$ defines a step function $func(w)$ such that $word(func(w)) = w$. We aim to show that the behavior of an ASC on a step function f is captured by a discrete transducer on word $word(f)$.

First, observe that an ASC with integer delays transforms step functions into step functions. Indeed, the output at time t depends on the input and secondary variables at time t, which are equal to the values of excitation variables at times $\{t - d^1, \dots, t - d^k\}$. The excitation variables at times $\{t - d^1, \dots, t - d^k\}$ depend on inputs and secondary variables at times $\{t - d^1, \dots, t - d^k\}$. As delays are integers, by unraveling the definition of the output variables (resp., excitation and secondary variables) at time t, we obtain that the variables depend solely on (a subset of) inputs at times $t, t-1, \dots, frac(t)+1, frac(t)$, where $frac(t)$ is the fractional part of t. Therefore, if an input is a step function, then excitation, secondary and output variables are all step functions. Moreover, the value of the step function output in the interval $[j, j+1)$ with $j \in \mathcal{N}$ can be computed using the input value in the interval $[j, j+1)$ and the values of excitation variables in the intervals $[j - d^1, j + 1 - d^1), \dots [j - d^k, j + 1 - d^k)$. Therefore, we can define a discrete letter-to-letter transducer that simulates the given ASC. Such a transducer remembers in its states the values of the excitation variables in the last $\max(d^1, \dots, d^k)$ intervals.

Lemma 21. *(1) If the input to an ASC is a step function, the output is a step function. (2) Given an ASC $\mathcal{C}$, one can compute in polynomial space a discrete*

letter-to-letter transducer $\mathcal{T}_{\mathcal{C}}$ such that for every step function f, the output of $\mathcal{C}$ on f is $func([\![\mathcal{T}_{\mathcal{C}}(word(f))]\!])$.

Remark 22. The transducer $\mathcal{T}_{\mathcal{C}}$ in Lemma 21 can be constructed in polynomial space, meaning that its sets of states and accepting states are succinctly representable and we can decide in polynomial time whether a given tuple (q, a, b, q') belongs to the transition relation of $\mathcal{T}_{\mathcal{C}}$.

Counterexamples to K-robustness of ASCs. Consider an ASC with integer delays that is not K-robust w.r.t. d_Σ, d_Γ. Then, there are two input functions f_1, f_2, satisfying $d_\Gamma([\![\mathcal{C}]\!](f_1), [\![\mathcal{C}]\!](f_2)) > K \cdot d_\Sigma(f_1, f_2)$, that are counterexamples to K-robustness. We show that there exists a pair of step functions g_1, g_2 that are counterexamples to K-robustness as well. Recall that the output of the ASC at time t depends only on inputs at times $t, t-1, \ldots, frac(t)+1, frac(t)$. Hence, we argue that if f_1, f_2 are counterexamples to K-robustness, then for some $x \in [0, 1)$, f_1, f_2 restricted to the domains $\Delta_1^x = \{y \in \mathrm{dom}(f_1) \mid frac(y) = x\}$, $\Delta_2^x = \{y \in \mathrm{dom}(f_2) \mid frac(y) = x\}$, respectively, are also counterexamples to K-robustness. Since the sets Δ_1^x, Δ_2^x are discrete, we can define step functions g_1, g_2 based on f_1, f_2 restricted to Δ_1^x, Δ_2^x, respectively..

Lemma 23. *Let $\mathcal{C}$ be an ASC with integer delay elements. If $\mathcal{C}$ is not K-robust w.r.t. timed Manhattan distances d_Σ, d_Γ, then there exists a pair of step functions g_1, g_2 such that $d_\Gamma([\![\mathcal{C}]\!](f_1), [\![\mathcal{C}]\!](f_2)) > K \cdot d_\Sigma(f_1, f_2)$.*

K-robustness of discrete transducers. We next present a decidability result that follows from [12]. Deciding K-robustness of letter-to-letter transducers w.r.t. generalized Manhattan distances reduces to quantitative non-emptiness of weighted automata with Sum-value function [12]. The latter problem can be solved in nondeterministic logarithmic space, assuming that the weights are represented by numbers of logarithmic length. Hence, we obtain the following result for *short* generalized Manhattan distances, i.e., distances whose `diff` values are represented by numbers of logarithmic length.

Lemma 24. *Deciding K-robustness of letter-to-letter transducers w.r.t. short generalized Manhattan distances is in* NLogspace.

We can now characterize the complexity of checking K-robustness of ASCs.

Theorem 25. *Deciding K-robustness of ASCs with respect to timed Manhattan distances is* PSpace*-complete.*

Proof. Observe that the timed Manhattan distance between step functions f, g equals the generalized Manhattan distance between the words $word(f), word(g)$ corresponding to step functions f, g. This, together with Lemmas 21 and 23, allows us to reduce checking K-robustness of ASCs w.r.t. timed Manhattan distances to checking K-robustness of the corresponding letter-to-letter transducers w.r.t. generalized Manhattan distances. It then follows from Lemma 24

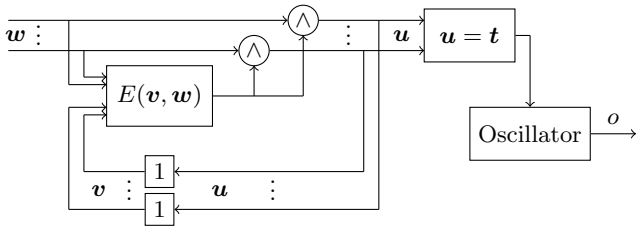


Fig. 4: The diagram of an ASC from the reduction of the reachability in succinctly represented graphs to K-robustness of ASCs.

that checking K-robustness of ASCs is in PSPACE. Note that we consider short generalized Manhattan distances whose descriptions are logarithmic in the exponential size of the letter-to-letter transducer.

The PSPACE-hardness of checking K-robustness of ASCs is obtained by a reduction from the reachability problem for *succinctly represented graphs*, which is PSPACE-complete [15]. Succinctly represented graphs are given indirectly by a propositional formula $E(\boldsymbol{v}, \boldsymbol{w})$, where $\boldsymbol{v}, \boldsymbol{w}$ are vectors of n variables. The vertexes of the graph are binary sequences of length n, and two sequences are connected by an edge iff the formula $E(\boldsymbol{v}, \boldsymbol{w})$ on these sequences holds. Consider the graph G represented by the formula $E(\boldsymbol{v}, \boldsymbol{w})$ and its vertex $\boldsymbol{t}$. We claim that the ASC given in Fig. 4 is K-robust iff the vertex $\boldsymbol{t}$ is not reachable from the zero vector $(0, \ldots, 0)$ in G. Due to Lemma 23 it suffices to focus on inputs that are step functions f, or discrete words $word(f)$. The input is interpreted as a sequence of vertexes of G. The ASC in Fig. 4 consists of (a) a circuit $E(\boldsymbol{v}, \boldsymbol{w})$ which checks whether there is an edge between $\boldsymbol{v}$ and the input $\boldsymbol{w}$, (b) a unit that tests whether $\boldsymbol{u}$ equals the target vertex $\boldsymbol{t}$ and, (c) an oscillator (2) which outputs 0 when the input is 0, and once the input is 1, outputs 1 until the end of the input. Initially, $\boldsymbol{v}$ is the zero vector. If there is an edge between $\boldsymbol{v}$ and $\boldsymbol{w}$, $\boldsymbol{u}$ is set to $\boldsymbol{w}$, and hence, $\boldsymbol{v}$ equals $\boldsymbol{w}$ in the next step and $\boldsymbol{w}$ is checked for equality with $\boldsymbol{t}$. If $\boldsymbol{w} = \boldsymbol{t}$, the oscillator is activated. Otherwise, if there is no edge between $\boldsymbol{v}$ and $\boldsymbol{w}$, $\boldsymbol{u}$ is set to the zero vector, which corresponds to transitioning back to the initial vertex; $\boldsymbol{v}$ equals the zero vector in the next step and the zero vector is checked for equality with $\boldsymbol{t}$.

If $\boldsymbol{t}$ is not reachable from the zero vector, the output of the ASC is always 0, and hence the ASC is K-robust for every K. Conversely, we claim that if t is reachable from the zero vector, then the ASC is not K-robust for any K. Indeed, consider a shortest path from the zero vector to the target vertex $\mathbf{0}, \boldsymbol{v}_1, \ldots, \boldsymbol{t}$ and consider the following two inputs: $i_1 = \mathbf{0}, \boldsymbol{v}_1, \ldots, \boldsymbol{t}, \mathbf{0}^K$, the path leading to activation of the oscillator followed by K inputs that are zero vectors, and, $i_2 = \mathbf{0}, \boldsymbol{v}_1, \ldots, \boldsymbol{t'}, \mathbf{0}^K$, which is obtained from i_1 by changing one bit in $\boldsymbol{t}$. Observe that the oscillator in ASC is not activated on the input i_2, hence the output is 0. Therefore, while the timed Manhattan distance between the inputs is 1, the timed Manhattan distance between the outputs is $K + 1$, for any chosen K.

□

Remark 26. Recall that the domain of an ASC $\mathcal{C}$ with input alphabet $\Sigma = \{0,1\}^m$ is given by $\mathrm{dom}(\mathcal{C}) = \mathcal{TL}(\Sigma)$. For any timed Manhattan distance $d_{TM}^{\leq 1}$ over $\mathrm{dom}(\mathcal{C})$ such that $\forall a, b \in \Sigma$, $\mathtt{diff}^{\leq 1}(a,b) \leq 1$, Proposition 13 states that the Skorohod distance w.r.t. $d_{TM}^{\leq 1}$ coincides with $d_{TM}^{\leq 1}$. Hence, K-robustness w.r.t. such Skorokhod distances is PSPACE-complete as well.

8 Conclusions

In this paper, we investigated the K-Lipschitz robustness problem for timed I/O systems using an automata-theoretic framework. For timed transducers, we showed that K-robustness can be decided in polynomial space for an interesting class of similarity functions. For ASCs, we reduce K-robustness w.r.t. timed Manhattan distances to K-robustness of discrete transducers and show PSPACE-completeness of the problem.

The essence of our framework is the use of weighted timed automata for computing similarity functions. This motivates further study of weighted timed automata; in particular, development of more expressive weighted timed automata (with nice decidability properties) immediately improves our results.

We also plan to study robustness of other models such as probablistic systems and explore specific application domains such as robotics.

References

1. Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.
2. Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *SFM*, volume 3185 of *LNCS*, pages 1–24. Springer, 2004.
3. R. Bloem, K. Greimel, T. Henzinger, and B. Jobstmann. Synthesizing Robust Systems. In *Formal Methods in Computer Aided Design (FMCAD)*, pages 85–92, 2009.
4. Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem on weighted timed automata. *FMSD*, 31(2):135–175, October 2007.
5. Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robustness in timed automata. In Parosh Aziz Abdulla and Igor Potapov, editors, *Reachability Problems - 7th International Workshop, RP 2013, Uppsala, Sweden, September 24-26, 2013 Proceedings*, volume 8169 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.
6. P. Cerny, T. Henzinger, and A. Radhakrishna. Simulation Distances. In *Conference on Concurrency Theory (CONCUR)*, pages 253–268, 2010.
7. S. Chaudhuri, S. Gulwani, and R. Lublinerman. Continuity Analysis of Programs. In *Principles of Programming Languages (POPL)*, pages 57–70, 2010.
8. S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. Navidpour. Proving Programs Robust. In *Foundations of Software Engineering (FSE)*, pages 102–112, 2011.
9. L. Doyen, T. A. Henzinger, A. Legay, and D. Ničković. Robustness of Sequential Circuits. In *Application of Concurrency to System Design (ACSD)*, pages 77–84, 2010.

10. V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust Timed Automata. In *HART*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 1997.
11. T. A. Henzinger. Two Challenges in Embedded Systems Design: Predictability and Robustness. *Philosophical Transactions of the Royal Society*, 366:3727–3736, 2008.
12. Thomas A Henzinger, Jan Otop, and Roopsha Samanta. Lipschitz robustness of finite-state transducers. In *FSTTCS 2014*, volume 1, page 431, 2014.
13. K. Zhou and J. C. Doyle and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1996.
14. Moez Krichen and Stavros Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.
15. Antonio Lozano and José L Balcázar. The complexity of graph problems for succinctly represented graphs. In *Graph-Theoretic Concepts in Computer Science*, pages 277–286. Springer, 1990.
16. R. Majumdar, E. Render, and P. Tabuada. A Theory of Robust Omega-regular Software Synthesis. *ACM Transactions on Embedded Computing Systems*, 13, 2013.
17. R. Majumdar and I. Saha. Symbolic Robustness Analysis. In *IEEE Real-Time Systems Symposium*, pages 355–363, 2009.
18. R. Samanta, J. V. Deshmukh, and S. Chaudhuri. Robustness Analysis of Networked Systems. In *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, pages 229–247, 2013.
19. R. Samanta, J. V. Deshmukh, and S. Chaudhuri. Robustness Analysis of String Transducers. In *ATVA*, pages 427–441. LNCS 8172, Springer, 2013.
20. Christos Stergiou, Stavros Tripakis, Eleftherios Matsikoudis, and Edward A Lee. On the verification of timed discrete-event models. In *Formal Modeling and Analysis of Timed Systems*, pages 213–227. Springer, 2013.