Input-Output Robustness for Discrete Systems [*]

Paulo Tabuada
University of California
Los Angeles
tabuada@ee.ucla.edu

Ayca Balkan
University of California
Los Angeles
abalkan@ucla.edu

Sina Y. Caliskan
University of California
Los Angeles
caliskan@ee.ucla.edu

Yasser Shoukry
University of California
Los Angeles
yshoukry@ee.ucla.edu

Rupak Majumdar
Max Planck Institute
for Software Systems
rupak@mpi-sws.org

ABSTRACT

Robustness is the property that a system only exhibits small deviations from the nominal behavior upon the occurrence of small disturbances. While the importance of robustness in engineering design is well accepted, it is less clear how to verify and design discrete systems for robustness. We present a theory of *input-output robustness* for discrete systems inspired by existing notions of *input-output stability (IO-stability)* in continuous control theory. We show that IO-stability captures two intuitive goals of robustness: bounded disturbances lead to bounded deviations from nominal behavior, and the effect of a sporadic disturbance disappears in finitely many steps. We show that existing notions of robustness for discrete systems do not have these two properties. For systems modeled as finite-state transducers, we show that IO-stability can be verified and the synthesis problem can be solved in polynomial time. We illustrate our theory using a reference broadcast synchronization protocol for wireless networks.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/program verification

General Terms

Design

Keywords

Robustness, stability, discrete systems, automata

1. MOTIVATION

A robust system is one that only modestly deviates from the nominal correct behavior upon the occurrence of small disturbances. Although it is accepted that engineered systems should be robust, it is less clear how to define robustness for discrete systems, such as finite-state transition systems.

In this paper, we present a theory of robustness for discrete systems. Our starting point is the research on robustness in continuous control theory, where one designs a controller for a "nominal" behavior assuming no disturbances and provides two guarantees: bounded disturbances have bounded consequences, and "nominal" behavior is eventually resumed after disturbances disappear. We adapt this view of robustness to the discrete setting by taking an input-output perspective of discrete systems. We define systems as transducers $f : \Sigma^* \to \Lambda^*$ mapping streams over an alphabet of inputs $\Sigma = \Sigma^c \times \Sigma^d$ to streams over an alphabet of outputs Λ. The alphabet Σ^c represents system inputs (also called *control* inputs) and the alphabet Σ^d represents *disturbances*. Typically, the designer designs the system assuming a nominal model of the environment, modeled by the stream $\perp^*$ for a special symbol $\perp \in \Sigma^d$. Disturbances represent deviations from the nominal model due to mismatches between the assumptions made about the environment at design time and the environment at run time. What goal should a robust design have? In keeping with robust continuous control, we postulate the following two natural requirements. First, every small disturbance should lead to a small deviation from the nominal behavior. Second, we require the effect of a sporadic disturbance to disappear over time. That is, if the environment deviates from the nominal for one step and subsequently follows the nominal environment, we require the effect of the deviation to disappear in finitely many steps. In this paper, we present a theory of robustness that captures both requirements.

To proceed, we must quantify "small" disturbances and "close" behaviors. In control theory, the elements in the sets Σ and Λ have well defined physical meaning —*e.g.*, voltages, currents, pressures, velocities— and thus these sets are equipped with norms that quantify the magnitude of their elements. In our setting, we endow Σ^* and Λ^* with quantitative information. We assume the existence of maps I and O taking strings in Σ^* and Λ^* respectively to *costs* in $\mathbb{N}_0$. Moreover, we require that $O(f(\sigma)) = 0$ for $\sigma \in \Sigma^*$ iff the behavior is correct.

[*]This work was partially supported by the NSF awards 0820061, 0953994, 1035916, and by the NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering.

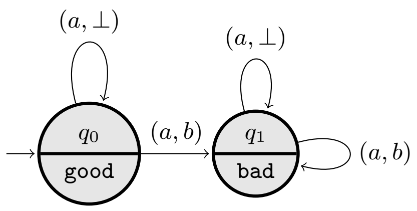

Figure 1: Automaton satisfying inequality (1) but for which the effects of sporadic disturbances do not disappear.

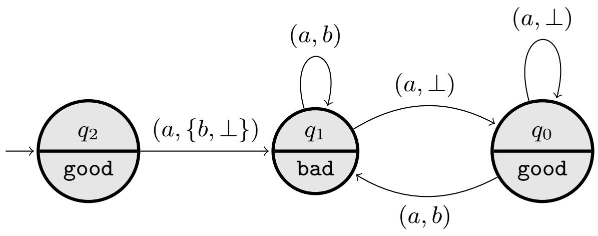

Figure 2: Automaton where the effects of sporadic disturbances disappear in finite time, but where the system deviates from the nominal behavior even when there is no disturbance.

The requirement that small disturbances lead to small deviations from nominal behavior is formulated as the inequality:

$$O(f(\sigma)) \leq \gamma\, I(\sigma) \qquad \forall \sigma \in \Sigma^*, \tag{1}$$

requiring the consequences $O(f(\sigma))$ to be no greater than the disturbance magnitude $I(\sigma)$ amplified by a gain γ. This inequality, or a variation of it, appeared in the recent work of Tarraf et al. on robustness of systems with finite input and output alphabets [20] and in Bloem et al. [4]. The same inequality also appeared in studies of robustness in control theory as old as [22], see [23] for a historical account. While it captures the first requirement on robust designs, it ignores the second, and we show through an example why this is not totally satisfactory.

Consider the automaton in Figure 1. The alphabets are $\Sigma^c = \{a\}$, $\Sigma^d = \{\perp, b\}$, $\Lambda = \{\texttt{good}, \texttt{bad}\}$, and $O(\lambda) = 0$ for any string $\lambda \in \Lambda^*$ whose last symbol is `good` and $O(\lambda) = 100$ for any string whose last symbol is `bad`. For the input strings we have $I(\sigma^c, \sigma^d) = 0$ for any string such that σ^d consists of only $\perp$s and $I(\sigma^c, \sigma^d) = 10$ for any string such that σ^d contains at least one b. It is simple to see that the inequality $O(f(\sigma)) \leq 10\, I(\sigma)$ holds, and thus the system is robust if we take inequality (1) as the definition of robustness. However, it is also clear that a sporadic disturbance described by the input string $b \perp^*$ will force the output to be `bad` for every time step after the first. Intuitively, we would expect that a robust system would return to normal behavior once a disturbance disappears. As this example shows,[1] the inequality (1) *does not* require such behavior and thus needs to be strengthened.

This leads us to our second requirement, the effect of a sporadic disturbance should disappear in finite time. This property appeared in the work of Doyen et al., see [9], as a different notion of robustness. Requiring the effects of sporadic disturbances to disappear in finite time does not guarantee that bounded disturbances lead to bounded consequences as the automaton in Figure 2 shows. The output alphabet and the output cost functions are the same as in the previous scenario. The alphabet Σ^d is now given by $\Sigma^d = \{\perp, b\}$ and $I(\sigma^c, \sigma^d)$ is 0 for any string σ^d containing only $\perp$'s and is 10 for any string σ^d containing at least one b. We then see that although the effect of the sporadic disturbance $b \perp^*$, disappears in one step, the system deviates from the nominal behavior even in the absence of the disturbances, e.g., when $\sigma^d = \perp$. It is easily seen that in this case it is not possible to find any γ that satisfies inequality (1) since $I(\sigma) = 0$ and $O(f(\sigma)) = 100$ for $\sigma = (a, \perp)$.

Interestingly, for linear control systems under the assumption of observability and controllability, the inequality (1) *does imply* that the effect of sporadic disturbances fades with time.[2] This observation explains why no additional requirement appears in the control literature dealing with robustness of linear control systems. The relevant notion of robustness that captures these two effects was introduced in Sontag's celebrated paper [19] addressing nonlinear control systems. Although the notion introduced by Sontag had a tremendous impact in control theory, it does not lend itself to the *quantitative* analysis, based on costs, that one typically expects when dealing with software systems. The reason is that for nonlinear control systems it is extremely difficult to obtain quantitative results and one settles for qualitative results ensuring the existence of certain bounds but not insisting on precise values for the constants appearing in these bounds. Therefore, instead of using the notion proposed by Sontag, we based our work on a recent notion proposed by Grüne [12] that is qualitatively equivalent to Sontag's notion but that allows for quantitative reasoning. Precisely, we say a transducer f is *input-output stable* (IOS) w.r.t. cost functions I and O if

$$O(f(\sigma)) \leq \max_{\sigma' \preceq \sigma} \{\gamma I(\sigma') - \eta(|\sigma| - |\sigma'|)\} \quad \forall \sigma \in \Sigma^*, \tag{2}$$

where $\preceq$ is the prefix ordering and γ and η are parameters. Here, γ describes how much the disturbance in the input is amplified at the output of the system and η quantifies the rate at which the system returns to its nominal behavior after a sporadic disturbance disappears. Our definition is inspired by Grüne's definition for nonlinear systems. A precise technical comparison is given in Section 6.

We show that (2) satisfies the two requirements for robust systems. Additionally, when the transducer f and costs I and O are modeled using finite-state automata, we can verify IOS (and find parameters γ and η) in polynomial time. In case a system is not IOS, we give a procedure to synthesize a controller enforcing IOS, again in polynomial time. We demonstrate our theory on a time synchronization protocol for wireless networks.

[1]This phenomenon also depends on the choice of the functions I and O. For control systems, the aforementioned phenomenon happens when I and O are both the infinity norm. When I and O are L_p norms ($p \neq \infty$), a persistent disturbance of small magnitude can lead to consequences of arbitrary magnitude if we wait sufficiently long. This is a different but equally undesired phenomenon.

[2]For control systems the relevant notion is to have the disturbance effects disappearing asymptotically rather than in finite time.

Finally, we demonstrate that our definition is compatible with IOS for control systems [12] by showing that if a control system is IOS then any transducer ε-approximate bisimilar to it is ε-practical IOS.

2. PROBLEM FORMULATION

2.1 Transducers and Cost Functions

An alphabet Σ is a finite set of letters. We write Σ^* for the set of words or strings over the alphabet Σ and ε for the empty word. The set of non-empty words over the alphabet Σ is denoted by Σ^+. Given a string $\sigma \in \Sigma^*$ we denote by $|\sigma|$ its length and by σ_{i-1} its ith element. We also use the notation $|S|$ to denote the cardinality of a set S. This should cause no confusion since sets are denoted by upper case letters while strings are denoted by lower case letters. For words $\sigma, \sigma' \in \Sigma^*$, we say σ' is a *prefix* of σ, written $\sigma' \preceq \sigma$, if there exists $\sigma'' \in \Sigma^*$ such that $\sigma'\sigma'' = \sigma$. In order to distinguish between elements of Σ and Σ^* we reserve the symbol s for letters and the symbol σ for words.

Let Σ and Λ be alphabets. A function $f : \Sigma^* \to \Lambda^*$ is called a *transducer* if for each $\sigma, \sigma' \in \Sigma^*$ such that $\sigma \preceq \sigma'$, we have $f(\sigma) \preceq f(\sigma')$.

In order to define robustness we assume an alphabet $\Sigma = \Sigma^c \times \Sigma^d$ where Σ^c is called the *control alphabet* and Σ^d is the *disturbance alphabet*. Every element $s \in \Sigma$ is thus a pair (s^c, s^d) where s^c is called a *control* input and s^d is a *disturbance* input. Furthermore, we assume Σ^d to contain a special element $\perp$ denoting the absence of disturbances.

We define the *nominal* behavior of a transducer $f : \Sigma^* \to \Lambda^*$ as the set of pairs of strings $(\sigma, f(\sigma)) \in \Sigma^* \times \Lambda^*$, where $\sigma^d \in \perp^*$, that is, there is no disturbance input.

In order to define robustness for transducers we associate cost functions to behaviors. Given an alphabet Σ, a *cost function* $c : \Sigma^* \to \mathbb{N}_0$ maps strings from Σ^* to the natural numbers. Given a cost function and a string σ, we define $c_\infty(\sigma) = \max_{\sigma' \preceq \sigma} c(\sigma')$. For a transducer $f : \Sigma^* \to \Lambda^*$, we denote the cost function on input strings Σ^* by I while the cost function on output strings Λ^* is denoted by O.

2.2 Input-Output Stability for Transducers

We can draw a formal analogy between a transducer and the response of a differential equation to a continuous-time input signal. This formal analogy leads to the following notion of Input-Output Stability inspired by Grüne's notion of stability in [12] and Sontag's notion of stability in [19].

Definition 1. [IOS] Let $f : \Sigma^* \to \Lambda^*$ be a transducer and let $I : \Sigma^* \to \mathbb{N}_0$ and $O : \Lambda^* \to \mathbb{N}_0$ be cost functions. Given parameters $\gamma, \eta \in \mathbb{N}$, we say the transducer f is (γ, η)-*input-output stable* (or (γ, η)-IOS) w.r.t. (I, O) if for each $\sigma \in \Sigma^*$ we have

$$O\left(f\left(\sigma\right)\right) \leq \max_{\sigma' \preceq \sigma} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(|\sigma| - |\sigma'|\right)\right\}. \qquad (3)$$

The parameter γ is called the *robustness gain* and the parameter η is called the *rate of decay*. We say a transducer f is *input-output stable* (or IOS) w.r.t. (I, O) if there exist $\eta, \gamma \in \mathbb{N}$ such that f is (γ, η)-IOS w.r.t. (I, O).

For ease of exposition we require the robustness gain and the decay rate to be natural numbers. The extension to the rational case is conceptually simple and thus not pursued in this paper.

We show that our notion of IOS captures two important properties: bounded disturbances lead to bounded consequences, and that the effects of a sporadic disturbance disappear in finite time.

For the first property, note that for every $\sigma \in \Sigma^*$, we have

$$\begin{aligned} O_\infty(f(\sigma)) &= \max_{\sigma' \preceq \sigma} O(f(\sigma')) \\ &\leq \max_{\sigma' \preceq \sigma} \max_{\sigma'' \preceq \sigma'} \left\{\gamma\, I\left(\sigma''\right) - \eta\left(|\sigma'| - |\sigma''|\right)\right\} \\ &\leq \max_{\sigma' \preceq \sigma} \max_{\sigma'' \preceq \sigma'} \left\{\gamma\, I\left(\sigma''\right)\right\} = \gamma\, I_\infty\left(\sigma\right). \qquad (4) \end{aligned}$$

Therefore, any bounded disturbance produces a bounded consequence. Moreover, the robustness gain γ measures how much the disturbance will be amplified by the transducer f.

Consider now a sporadic disturbance modeled as a string $s\sigma \in \Sigma^*$ such that $I(s\sigma') = 0$ for any prefix σ' of σ with $|\sigma'| > 0$. We regard the equality $I(s\sigma') = 0$ as the statement that the disturbance, as measured by I, disappears after s. We now note that it follows from (3) that for every σ' of length j we have:

$$O(f(s\sigma')) \leq \max\{0, \gamma\, I(s) - \eta j\}. \qquad (5)$$

Therefore, after $\lceil \gamma\, I(s)/\eta \rceil$ steps the effect of the disturbance has disappeared. Note how the number of steps depends on the magnitude of the effect of the disturbance, as measured by $\gamma\, I(s)$, and also on the rate of decay η.

Notice that in our formulation, we place only mild assumptions on cost functions. In particular, we do not require the cost functions to be monotonic. If the cost is monotonic then a disturbance that occurs at a certain time will never be forgotten, thus enabling the output to deviate from the desired one for all future time. Alternatively, we can have a cost that is not monotonic and thus forgets the effect of a disturbance. Such costs are adequate in situations where it is acceptable that upon the occurrence of a disturbance the system can restart the computation of the correct output.

We also note that no relation between the input cost and the presence of disturbances in the input is required for our results to hold. While one typically expects that in the absence of disturbances the cost of the input strings should be zero, there are situations where this is not the case, and it may be natural to consider the presence of certain errors in the computation. For example, consider computing a mathematical expression involving real numbers on a microcontroller using fixed-point arithmetic. In this case, it is convenient to have the input cost on input strings without disturbances to be non-zero so that we allow the output string to have non-zero output cost even in the absence of external disturbances.

3. VERIFICATION AND SYNTHESIS

We study the following verification problems.

(γ, η)-IOS Verification Given transducer f, input and output cost functions I and O respectively, and parameters γ and η, is the transducer f (γ, η)-IOS for I and O?

IOS Verification Given transducer f and input and output cost functions I and O respectively, does there exist γ and η such that f is (γ, η)-IOS for I and O? (If so, find such γ and η.)

We also consider the *synthesis problem.* A *controller* is a map:

$$C : \Sigma^* \times \Sigma^c \to \Sigma^c \tag{6}$$

transforming the history of past inputs $\sigma \in \Sigma^*$ and a given control input request s^c into the control input $C(\sigma, s^c)$ to be provided to the system. Composing a transducer f with a controller C results in a new transducer $f_C : \Sigma^* \to \Lambda^*$, called the *controlled system*, defined, intuitively, by filtering the input stream by C. Formally, define the *filtration* $\mathsf{filter}_C : \Sigma^* \to \Sigma^*$ of a stream by a controller C inductively as:

$$\begin{aligned} \mathsf{filter}_C(\varepsilon) &= \varepsilon \\ \mathsf{filter}_C(\sigma \cdot (s^c, s^d)) &= \mathsf{filter}_C(\sigma) \cdot (C(\sigma, s^c), s^d) \end{aligned}$$

The composition f_C is then given by $f_C(\sigma) = f(\mathsf{filter}_C(\sigma))$. The synthesis problem is the following.

Synthesis Given transducer f, cost functions (I, O), and parameters (γ, η), does there exist a controller C such that f_C is (γ, η)-IOS w.r.t. (I, O)?

We will study the verification and synthesis questions for transducers and cost functions described by finite-state (weighted) automata.

Definition 2. An automaton $A = (Q, q_0, \Sigma, \delta, \Lambda, H)$ consists of:

- a finite set of states Q;
- an initial state $q_0 \in Q$;
- a set of inputs Σ;
- a transition function $\delta : Q \times \Sigma \to Q$;
- a set of outputs Λ;
- and an output function $H : Q \to \Lambda$.

A run r of A on an input string $\sigma \in \Sigma^*$ is a string $r \in Q^*$ such that $r_0 = q_0$ and $r_{i+1} = \delta(r_i, \sigma_i)$ for $i = 0, 1, \ldots, |\sigma|-1$. A run r of A on input $\sigma \in \Sigma^*$ defines an *output run* $\lambda \in \Lambda^*$ by $\lambda_i = H(r_i)$ for $i = 0, 1, \ldots, |\sigma| - 1$.

We denote by δ^* the extension of δ to $Q \times \Sigma^*$ defined in the usual manner: $\delta^*(q, \varepsilon) = q$ and $\delta^*(q, \sigma s) = \delta(\delta^*(q, \sigma), s)$. The set of *predecessors* of a state $q \in Q$, denoted by $\mathrm{Pre}(q)$, is defined by $\mathrm{Pre}(q) = \{q' \in Q \mid \exists\, s \in \Sigma \quad \delta(q', s) = q\}$.

An automaton A defines a transducer $\mathsf{xduce}(A) : \Sigma^* \to \Lambda^*$ as follows:

$$f(\sigma) = H(\delta^*(q_0, \sigma_0))H(\delta^*(q_0, \sigma_0\sigma_1)) \ldots .$$

In addition to transducers we also consider cost functions described by finite-state weighted automata. A weighted automaton A is an automaton $A = (Q, q_0, \Sigma, \delta, \mathbb{N}_0, H)$ whose set of outputs or *weights* is $\mathbb{N}_0$ and whose output map satisfies $H(q_0) = 0$. The *cost* of a string $\sigma \in \Sigma^*$ is given by $H(\delta^*(q_0, \sigma))$. A weighted automaton A *defines* the cost function I_A as follows: $I_A(\sigma) = H(\delta^*(q_0, \sigma))$. We define the size of an automaton as the number of bits required to encode an automaton when all weights are given in unary.

3.1 Solving the Verification Problems

In this section we show how to solve the verification problem for transducers, input cost, and output cost defined by finite-state automata. It will be convenient to combine the automaton A^f describing the transducer, the automaton A^I describing the input cost I, and the automaton A^O describing the output cost O into the automaton A obtained by synchronizing A^I on the same input of A^f and synchronizing A^O on the output of A^f. Formally, A is defined by:

- $Q = Q^I \times Q^f \times Q^O$ with $q_0 = (q_0^I, q_0^f, q_0^O)$;
- $\Sigma = \Sigma^f$;
- $\Lambda = \mathbb{N}_0 \times \mathbb{N}_0$ and $H(q^I, q^f, q^O) = (H^I(q^I), H^O(q^O))$;

and the transition function:

$$\delta((q^I, q^f, q^O), s) = \left(\delta^I(q^I, s), \delta^f(q^f, s), \delta^O(q^O, H^f \circ \delta^f(q^f, s))\right).$$

In the following, we will abuse notation by using $H^I(q)$ and $H^O(q)$ to denote $H^I(q^I)$ and $H^O(q^O)$, respectively, for $q = (q^I, q^f, q^O)$. Without loss of generality we assume every state of A to be reachable from the initial state.

In addition to A we will also work with the set of all functions from Q to $M = \{0, 1, \ldots, \overline{m}\}$, denoted by M^Q, and where $\overline{m} = \max_{q \in Q} \gamma H^I(q)$. The set M^Q is a lattice with order relation denoted by $\sqsubseteq$ and defined for any $f_1, f_2 \in M^Q$ by $f_1 \sqsubseteq f_2$ if $f_1(q) \le f_2(q)$ for every $q \in Q$. The join of f_1 and f_2 is given by $\max\{f_1, f_2\}$ while the meet is given by $\min\{f_1, f_2\}$. We note that M^Q is a finite lattice and all the operators used in this paper satisfy the conditions of Tarski's theorem [21] so that the least fixed point of a monotonic operator exists and can be algorithmically found in $|Q| \cdot |M|$ steps.

3.1.1 (γ, η)-IOS Verification Problem

We formulate the solution of the (γ, η)-IOS verification problem as a fixed-point computation for the operator $F : M^Q \to M^Q$ defined by:

$$F(W)(q) = \max\left\{\gamma H^I(q), W(q), \min_{q' \in \mathrm{Pre}(q)} W(q') - \eta\right\}. \tag{8}$$

THEOREM 1. *Let A^f be a finite-state automaton. Let A^I and A^O be finite-state weighted automata defining costs I and O, respectively. Given $\eta, \gamma \in \mathbb{N}$, the transducer $\mathsf{xduce}(A^f)$ is (γ, η)-IOS with respect to (I, O) iff the infimal fixed point of F, denoted by W^*, satisfies the following inequality for every $q \in Q$:*

$$H^O(q) \le W^*(q). \tag{9}$$

PROOF. Let $f = \mathsf{xduce}(A^f)$. Any fixed point V of F satisfies:

$$V(q) = \max\left\{\gamma H^I(q), V(q), \min_{q' \in \mathrm{Pre}(q)} V(q') - \eta\right\}$$

which implies:

$$V(q) \ge \max\left\{\gamma H^I(q), \min_{q' \in \mathrm{Pre}(q)} V(q') - \eta\right\}.$$

Hence, the infimal fixed-point is characterized by the equality:

$$W^*(q) = \max\left\{\gamma H^I(q), \min_{q' \in \mathrm{Pre}(q)} W^*(q') - \eta\right\}. \tag{10}$$

$$\begin{aligned}
U(q) &= \min_{\sigma\in\Sigma^* \mid \delta^*(q_0,\sigma)=q} \max\left\{ \max_{\sigma'\preceq\sigma''} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma''\right| - \left|\sigma'\right| + 1\right)\right\}, \gamma\, I(\sigma)\right\} \\
&= \min_{\sigma''\in\Sigma^* \mid \delta^*(q_0,\sigma'')\in \mathrm{Pre}(q)} \max\left\{ \max_{\sigma'\preceq\sigma''} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma''\right| - \left|\sigma'\right|\right)\right\} - \eta, \gamma H^I(q)\right\} \\
&= \max\left\{ \min_{\sigma''\in\Sigma^* \mid \delta^*(q_0,\sigma'')\in \mathrm{Pre}(q)} \max_{\sigma'\preceq\sigma''} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma''\right| - \left|\sigma'\right|\right)\right\} - \eta, \gamma H^I(q)\right\} \\
&= \max\left\{ \min_{q'\in\mathrm{Pre}(q)} U(q') - \eta, \gamma H^I(q)\right\}. \qquad (7)
\end{aligned}$$

Let now:

$$U(q) = \min_{\sigma\in\Sigma^* \mid \delta^*(q_0,\sigma)=q} \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma\right| - \left|\sigma'\right|\right)\right\}.$$

We now show that U is the infimal fixed point of F by showing that U satisfies (10). Let σ'' be the prefix of σ of length $|\sigma|-1$. Using σ'' we can write U as (7).

We can now use the fact that U is the infimal fixed point of F to finalize the proof. By definition of U, for any string $\sigma\in\Sigma^*$ such that $\delta^*(q_0,\sigma)=q$ we have:

$$U(q) \le \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma\right| - \left|\sigma'\right|\right)\right\}. \qquad (11)$$

Hence, if the inequality $H^O(q) \le W^*(q)$ holds we conclude that f is IOS by chaining $O(f(\sigma)) = H^O(q)$, $H^O(q) \le W^*(q)$, $W^*(q) = U(q)$, and (11).

Conversely, if f is IOS then:

$$O(f(\sigma)) = H^O(q) \le \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma\right| - \left|\sigma'\right|\right)\right\}$$

for every $\sigma\in\Sigma^*$ such that $\delta^*(q_0,\sigma)=q$ and thus $H^O(q) \le U(q) = W^*(q)$. □

3.1.2 IOS Verification Problem

We now check, for a transducer and cost functions given by automata, if there exists a choice of γ and η for which the transducer is (γ,η)-IOS. In addition to answer this question algorithmically, we also provide a subset of all the possible choices of γ and η. This will be done through an operator $G : (M\times\{0,1,\ldots,|Q|\})^Q \to (M\times\{0,1,\ldots,|Q|\})^Q$ closely related to F. We use the notation $G(f,q) = (G_W(f,q), G_D(f,q))$ to define G as:

$$G_W(W)(q) = \max\left\{H^I(q), W(q), \min_{q'\in\mathrm{Pre}(q)} W(q')\right\}$$

$$G_D(D)(q) = \begin{cases} D(q')+1 & \text{if } \min_{q'\in\mathrm{Pre}(q)} W(q') > W(q) \\ D(q) & \text{otherwise.}\end{cases}$$

Note that we now work on the lattice $(M\times\{0,1,\ldots,|Q|\})^Q$ with order given by $(W,D)\sqsubseteq(W',D')$ when for any $q\in Q$, we have one and only one of the following cases:

- $W(q) < W'(q)$,
- $W(q) = W'(q)$, $D(q) \le D'(q)$.

It is easy to check that G is monotone with respect to this order.

Theorem 2. *Let A^f be a finite-state automaton. Let A^I and A^O be finite-state weighted automata defining the costs I and O, respectively. There exist $\gamma,\eta\in\mathbb{N}$ so that $\mathsf{xduce}(A^f)$ is (γ,η)-IOS with respect to (I,O) iff the least fixed point of G, denoted by (W^*,D^*), satisfies the following inequality for every $q\in Q$:*

$$W^*(q) = 0 \implies H^O(q) = 0. \qquad (12)$$

Furthermore, when (12) holds, $\mathsf{xduce}(A^f)$ is (γ,η)-IOS for any γ satisfying:

$$H^O(q) > \gamma H^I(q) \implies H^O(q) < \gamma W^*(q), \qquad \forall q\in Q \qquad (13)$$

and any η (dependent on the choice of γ) satisfying:

$$\eta \le \left\lfloor \frac{\beta}{d} \right\rfloor, \qquad (14)$$

where

$$d = \max_{q\in Q} D^*(q),$$

$$\beta = \min_{q\in Q \mid H^O(q) > \gamma H^I(q)} \gamma W^*(q) - H^O(q).$$

Proof. Let $f = \mathsf{xduce}(A^f)$. We first show that if f is (γ,η)-IOS for some γ and η then (12) holds. By definition of IOS and since $\eta(|\sigma|-|\sigma'|)\ge 0$ we have:

$$\begin{aligned}
O\left(f\left(\sigma\right)\right) &\le \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right) - \eta\left(\left|\sigma\right| - \left|\sigma'\right|\right)\right\} \\
&\le \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right)\right\} \qquad (15)
\end{aligned}$$

for every $\sigma\in\Sigma^*$. Hence, we also have:

$$\begin{aligned}
O(f(\sigma)) &= H^O(q) \\
&\le \min_{\sigma\in\Sigma^* \mid \delta^*(q_0,\sigma)=q} \max_{\sigma'\preceq\sigma} \left\{\gamma\, I\left(\sigma'\right)\right\} = V^*(q)
\end{aligned}$$

where V^* is the infimal fixed point of F for $\eta = 0$. Note that $V^*(q) = 0$ implies $H^O(q) = 0$ since H^O is a non-negative function. Furthermore, $W^* = V^*/\gamma$ and thus:

$$W^*(q) = 0 \implies V^*(q) = 0 \implies H^O(q) = 0.$$

We now show that (12) implies IOS. We will do so by constructing γ and η and we will conclude from the construction that any choice of γ satisfying (13) and any choice of η satisfying (14) works.

We start by showing that under the stated assumptions there exists $\gamma > 0$ so that $O(f(\sigma)) < \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma')\}$. We first note that W^* equals the infimal fixed point of F for $\gamma = 1$ and $\eta = 0$. It then follows from the proof of Theorem 1 that W^* is given by:

$$W^*(q) = \min_{\sigma\in\Sigma^* \mid \delta^*(q_0,\sigma)=q} \max_{\sigma'\preceq\sigma} \{I(\sigma')\}. \qquad (16)$$

For any q we can find $\gamma_q > 0$ so that $H^O(q) < \gamma_q W^*(q)$. Note that when $W^*(q) = 0$ we have $H^O(q) = 0$, by assumption, so any γ_q works in this case but for simplicity we take

$\gamma_q = 0$. With $\gamma = \max_{q\in Q} \gamma_q$ we have:

$$\begin{aligned} O(f(\sigma)) = H^O(q) &< \gamma W^*(q) \\ &\leq \gamma \max_{\sigma'\preceq\sigma}\{I(\sigma')\} = \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma')\} \end{aligned}$$

for every $\sigma \in \Sigma^*$ such that $\delta^*(q_0,\sigma) = q$. If we define $\beta_q > 0$ by $-\beta_q = H^O(q) - \gamma W^*(q)$ for every $q \in Q$ for which $W^*(q) \neq 0$ and define $\beta > 0$ by $\beta = \min_{q\in Q}\beta_q$ we have $H^O(q) = \gamma W^*(q) - \beta_q \leq \gamma W^*(q) - \beta$. In order to express β in the form $\eta(|\sigma| - |\sigma'|)$ we note that the evaluation of the fixed point (W^*, D^*) at q tells us that the run of A on any $\sigma \in \Sigma^*$ such that $\delta^*(q_0,\sigma) = q$ visits a state q' such that $\gamma H^I(q') \geq W^*(q)$. Furthermore, if we denote by $q' = q_1q_2\ldots q_{D(q)}q_{D(q)+1} = q$ the states visited between q' and q we also know that $\gamma H^I(q_i) > \gamma H^I(q_{i+1})$ for $i = 1,\ldots,D(q)$. We now consider two cases. If $\gamma H^I(q) < H^O(q)$ then the inequality $H^O(q) < \gamma W^*(q)$ implies that $D^*(q) > 0$ since $W(q)$ had to be updated in order to increase to $W^*(q)$. Let $d = \max_{q\in Q} D^*(q)$ (note that d is no larger than the number of states in Q) and define η as $\eta = \lfloor \beta/d \rfloor$ to obtain:

$$\begin{aligned} O(f(\sigma)) = H^O(q) &\leq \gamma W^*(q) - \beta \\ &\leq \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma') - \beta\} \\ &\leq \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma') - \eta(|\sigma| - |\sigma'|)\}. \end{aligned}$$

The second case to be considered is $\gamma H^I(q) \geq H^O(q)$. In this case we directly have:

$$\begin{aligned} O(f(\sigma)) = H^O(q) &\leq \gamma H^I(q) \\ &= \gamma\, I(\sigma) \leq \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma') - \eta(|\sigma| - |\sigma'|)\} \end{aligned} \tag{17}$$

where the last inequality follows by taking $\sigma' = \sigma$. We then conclude that any choice of γ and η satisfying (13) and (14), respectively, renders f IOS. $\square$

Notice that our characterization gives a natural dynamic programming formulation for verification.

3.2 Solving the Synthesis Problem

We now consider the synthesis problem for a transducer $f : \Sigma^* \to \Lambda^*$. Recall that the purpose of synthesizing a controller C is to render f_C (γ,η)-IOS for a desired choice of γ and η. Once again we follow the perspicacious work of Grüne et al. [13], where it is shown how control systems can be augmented with a monitor for the stability property introduced in [12].

Let A be the automaton introduced in Section 3.1 obtained by synchronizing A^I on the same input of A^f and synchronizing A^O on the output of A^f. We now extend this automaton to $A^{\gamma\eta}$ by enlarging the set of states to:

$$Q^{\gamma\eta} = Q \times M, \qquad q_0^{\gamma\eta} = (q_0, 0)$$

and extending the transition function to:

$$\delta^{\gamma\eta}((q,m),s) = \left(\delta(q,s), \max\left\{m - \eta, \gamma H^I \circ \delta(q,s)\right\}\right).$$

Automaton $A^{\gamma\eta}$ monitors IOS in the sense that if the input $\sigma \in \Sigma^*$ takes the initial state to $(q,m) \in Q \times M$ then:

$$m = \max_{\sigma'\preceq\sigma}\{\gamma\, I(\sigma') - \eta(|\sigma| - |\sigma'|)\}. \tag{18}$$

This is proved by induction on the length of the trace. As a direct consequence, we can monitor IOS by testing the inequality $H^O(q) \leq m$, as stated in the next result.

Theorem 3. *Let A^f be a finite-state automaton. Let A^I and A^O be finite-state weighted automata defining the costs I and O, respectively. Given $\gamma,\eta \in \mathbb{N}$, the transducer $\mathsf{xduce}(A^f)$ is (γ,η)-IOS with respect to (I,O) iff every reachable state (q,m) of $A^{\gamma\eta}$ satisfies $H^O(q) \leq m$.*

Note that Theorem 3 provides an alternative algorithm to check (γ,η)-IOS: check that

$$S = \{(q,m) \in Q \times M \mid H^O(q) \leq m\}$$

is an invariant set of $A^{\gamma\eta}$. However, the fixed-point representation provided by the functional (8) is of independent interest, as well as key to Theorem 2.

Theorem 3 also suggests a synthesis algorithm for IOS: construct a controller rendering the set S invariant. This will require a modification of $A^{\gamma\eta}$ that we denote by $A^{\gamma\eta c}$, and define as follows. The new state set is:

$$Q^{\gamma\eta c} = \Sigma^c \times Q \times M = \Sigma^c \times Q^I \times Q^f \times Q^O \times M$$

and the new transition function is:

$$\delta\left((s^c, q^I, q^f, q^O, m), (s^{c\prime\prime}, s^{d\prime\prime})\right) = (s^{c\prime}, q^{I\prime}, q^{f\prime}, q^{O\prime}, m'),$$

with:

$$s^{c\prime} = s^{c\prime\prime}, \quad q^{I\prime} = \delta^I\left(q^I, \left(s^{c\prime\prime}, s^{d\prime\prime}\right)\right)$$

$$q^{f\prime} = \delta^f\left(q^f, (s^{c\prime\prime}, s^{d\prime\prime})\right), \quad q^{O\prime} = \delta^O\left(q^O, H^f\left(q^{f\prime}\right)\right),$$

$$m' = \max\left\{m - \eta, \gamma H^I\left(q^{I\prime}\right)\right\}.$$

We can now apply the usual invariance controller synthesis algorithm (see, e.g., [17]) to $A^{\gamma\eta c}$ to construct a (memoryless) controller forcing S to be an invariant set. This observation immediately leads to the next result.

Theorem 4. *Let A^f be a finite-state automaton, and let $f = \mathsf{xduce}(A^f)$. Let A^I and A^O be finite-state weighted automata defining the costs I and O, respectively. Given $\gamma,\eta \in \mathbb{N}$, there exists a controller C rendering f_C (γ,η)-IOS with respect to (I,O) iff there exists a controller rendering the set S invariant for $A^{\gamma\eta c}$.*

Since safety games can be solved in linear time [17], the complexity of synthesis is linear in the size of $A^{\gamma\eta c}$, i.e., it runs in $O(|A^I||A^f||A^O||M||\Sigma|)$ time.

4. EXAMPLE: ROBUSTNESS FOR THE RBS PROTOCOL

We consider the problem of time synchronization in wireless networks using the Reference Broadcast Synchronization (RBS) protocol [10]. To simplify the presentation we consider a network of three nodes: one master node and two slave nodes that we denote by node a and node b. Nodes transmit packets composed by a message identifier (small number of bits) and the message payload (large number of bits). We assume the packets to be encoded with an error correcting code that can always correct the message identifier but not necessarily the message payload when there are communication errors.

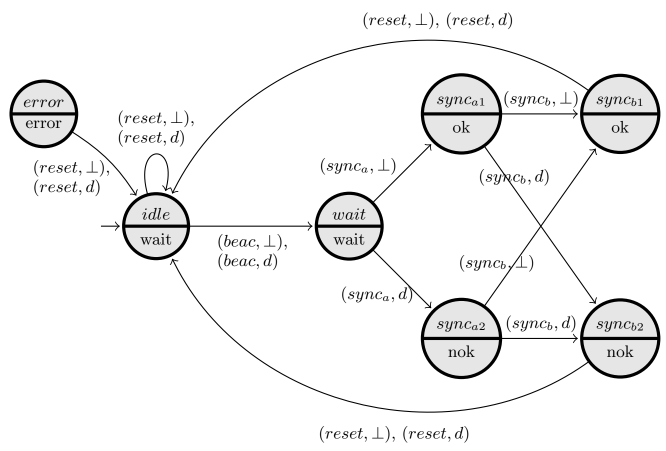
Figure 3: Automaton representation of the RBS time synchronization protocol. Not represented in the automaton are transitions to the error state. These depart from every state so as to make the automaton input-enabled.

The RBS protocol starts with the master node broadcasting a beacon message *beac* at time t_1. This packet contains no payload. Due to access, propagation, and queuing delays both slave nodes will receive the packet at global time t_2 which is represented in the local time of node a as t_{2a} and in the local time of node b as t_{2b}. Node a then sends a synchronization message $sync_a$ to node b containing t_{2a} as payload and then node b sends a synchronization message $sync_b$ to node a containing t_{2b} as payload. If there are no communication errors, node a can compute the offset of its clock with respect to the clock of node b as $t_{2a} - t_{2b}$ and similarly for node b.

Consider now the effect of communication errors modeled as a disturbance. Since the message *beac* contains no payload it can always be correctly decoded even in the presence of communication errors. The $sync_a$ message, however, contains t_{2a} as payload and a communication error may cause the payload to be incorrectly decoded as $t_{2a} + d$ where d the decoding error. In the presence of communication errors the computed time offset becomes $t_{2a} - t_{2b} + d$ and similarly for node b. The protocol also includes a *reset* message that can be sent at any time to re-start the synchronization process. Since the *reset* message has no payload it is not affected by communication errors. A second source of errors in the protocol are out of order messages. In this scenario, the protocol goes to the *error* state until a *reset* message is received. Figure 3 shows an automaton representation for this protocol. For simplicity, all the incoming edges to the *error* state, departing from every state, are not displayed. The input and output alphabets are given by:

$$\Sigma^c = \{beac, sync_a, sync_b, reset\}, \quad \Sigma^d = \{\perp, d\},$$

$$\Lambda = \{error, wait, ok, nok\}$$

the initial state is *idle* and the output corresponding to any given state is depicted in the bottom part of the state.

In order to study the robustness properties of the RBS protocol we need to equip Σ^* and Λ^* with quantitative information through the input and output cost functions. The input cost function is defined in two steps. First, we describe how disturbances are penalized by the cost function. We denote this cost function by I^d. It counts the number of d symbols that appear in σ^d up to a maximum of 4. Every time a d symbol appears, the input cost increases by 2 otherwise the input cost decreases by 1. The choice of the upper bound 4 is a consequence of the RBS protocol described by the automaton in Figure 3 where we can see that at most 4 disturbances can take place before returning to an *error* or *idle* state. A complete description is provided in Figure 4. In addition to I^d we also consider a cost function I^c penalizing the control inputs. This cost maps the sequences of strings of control inputs expected by the protocol to 0 and any other sequence to 9 or 10. We reserve 9 for the occur-

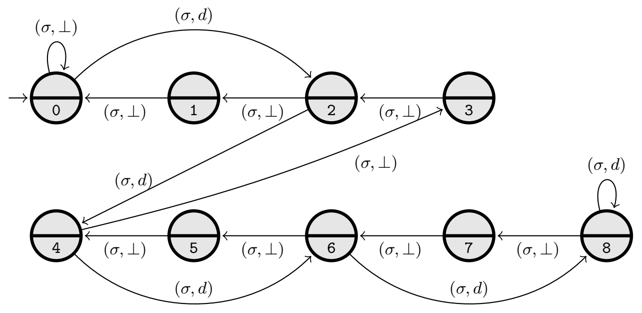
Figure 4: Input cost I^d penalizing disturbance inputs.

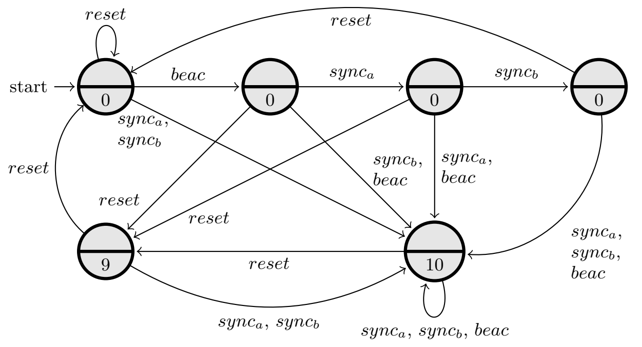
Figure 5: Input cost I^c penalizing control inputs.

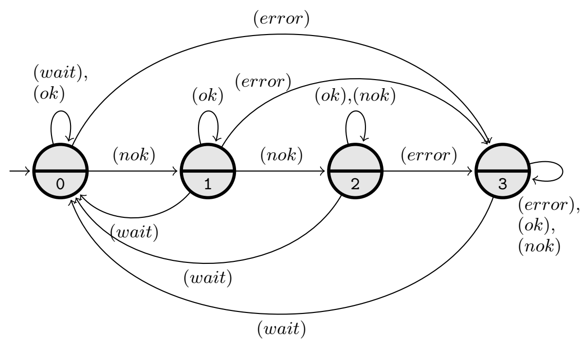
Figure 6: Output cost O.

rence of a *reset* message appearing in the middle of the protocol execution and use 10 for all other messages appearing out of order. We regard as more damaging a control input appearing out of order over a reset message out of order. A complete description of I^c is given in Figure 5. Finally, we define the input cost I by $I(\sigma) = \max\{I^c(\sigma), I^d(\sigma)\}$. Note that the values for I^c and I^d were chosen so that I ranges in the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

In order to completely specify the problem we also need to define the output cost O. The output cost counts the number of *nok* output symbols up to a maximum of 2 and penalizes the occurrence of the *error* output symbol with a 3 as depicted in Figure 6. The upper bound of 2 is a consequence of only having at most two consecutive *nok* output symbols. The choice of 2 and 3 reflects our belief that it is preferable to compute an incorrect estimate of the clock offsets than not being able to compute any estimate. Clearly, other choices of input and output cost functions are possible and they would lead to different conclusions.

Although the RBS protocol is guaranteed to work when there are no communication disturbances, we are interested in analyzing its behavior when such disturbances are present. By applying Theorem 1 with $\gamma = 2$ and $\eta = 1$ we conclude that the RBS protocol is $(2, 1)$-IOS. By using Theorem 2 we reach a stronger conclusion: IOS only holds for $\gamma > 0.5$. The range of allowable η changes according to the choice of γ. For example when $\gamma = 0.6$ and by applying Theorem 2 we find that $\eta < 0.0067$.

Moreover, for $\gamma \geq 1$, IOS holds for any value of $\eta > 0$. These values of γ and η give us very precise information about the robustness of the RBS protocol. We first note that if we consider only inputs strings $\sigma \in \Sigma^*$ satisfying $I(\sigma) \leq 8$ we are only considering disturbances due to communication errors. In that case IOS provides the guarantee $O(f(\sigma)) \leq 2$ and an analysis of the cost O shows that we never reach the *error* state, *i.e.*, the protocol is able to compute the clock offsets, albeit with an error. When $I(\sigma) > 8$ incorrect control inputs are being used and for this reason the protocol may enter the *error* state and not be able to compute an estimate of the clock offsets. Hence, smaller disturbances lead to smaller consequences. Recall that for an IOS transducer, the nominal behavior can be resumed in no more than $\lceil \gamma I(\sigma)/\eta \rceil$ steps once a disturbance disappears. In this present case, since IOS holds for arbitrarily large η the nominal behavior can be resumed in one time step. This can be readily verified by noticing that the reset input can be used for this purpose at any state of the system.

5. IOS AND REFINEMENT

Since IOS is a property of the input-output behavior of a system, and since refinement relations preserve input-output behaviors, we immediately have that if two automata A_1 and A_2 are bisimilar, then $\mathsf{xduce}(A_1)$ is (γ, η)-IOS w.r.t. (I, O) iff $\mathsf{xduce}(A_2)$ is (γ, η)-IOS w.r.t. (I, O). We generalize this observation to ε-approximate bisimulation relations [11, 18] using *practical IOS* of systems. We start with some notation. We denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{R}_0^+$ the set of non-negative real numbers, and, for $n \in \mathbb{N}$, we denote by $\mathbb{R}^n$ the n-dimensional Euclidean space.

5.1 Practical IO-stability and Approximate Bisimulation

IOS is an "ideal" property of transducers. We define the notion of *practical IOS*, that requires that the output and inputs of a transducer satisfy the relation (3) up to a parameter ε. Let $f : \Sigma^* \to \Lambda^*$ be a transducer and let $I : \Sigma^* \to \mathbb{R}_0$ and $O : \Lambda^* \to \mathbb{R}_0$ be cost functions. Given parameters $\gamma, \eta \in \mathbb{R}$, we say the transducer f is $(\varepsilon, \gamma, \eta)$*-practical input-output stable* (or $(\varepsilon, \gamma, \eta)$-practical IOS) w.r.t. (I, O) if for each $\sigma \in \Sigma^*$ we have

$$O(f(\sigma)) \leq \max_{\sigma' \preceq \sigma} \{\gamma I(\sigma') - \eta(|\sigma| - |\sigma'|)\} + \varepsilon. \tag{19}$$

In order to define approximate bisimulation we consider $\varepsilon \in \mathbb{R}_0^+$ and two automata $A_1 = (Q_1, q_{01}, \Sigma, \delta_1, \Lambda, H_1)$ and $A_2 = (Q_2, q_{02}, \Sigma, \delta_2, \Lambda, H_2)$ with the same sets of inputs and outputs. Let $I : \Sigma^* \to \mathbb{R}_0$ and $O : \Lambda^* \to \mathbb{R}_0$ be cost functions. A relation $R \subseteq Q_1 \times Q_2$ is said to be an ε*-approximate bisimulation* if $(q_1, q_2) \in R$ implies:

1. $|O(H_1(q_1)) - O(H_2(q_2))| \leq \varepsilon$;
2. for each $\sigma_1 \in \Sigma$ there is a $\sigma_2 \in \Sigma$ such that $|I(\sigma_1) - I(\sigma_2)| \leq \varepsilon$ and $(\delta_1(q_1, \sigma_1), \delta_2(q_2, \sigma_2)) \in R$; and
3. for each $\sigma_2 \in \Sigma$ there is a $\sigma_1 \in \Sigma$ such that $|I(\sigma_1) - I(\sigma_2)| \leq \varepsilon$ and $(\delta_1(q_1, \sigma_1), \delta_2(q_2, \sigma_2)) \in R$.

The automaton A_1 is ε-approximate bisimilar to A_2 if there is an ε-approximate bisimulation relation R such that $R(q_{01}, q_{02})$.

We now consider ε-bisimilar automata A_1 and A_2 and assume $\mathsf{xduce}(A_2)$ to be (γ, η)-IOS. Since A_1 is ε-bisimilar to A_2, the behavior of A_1 determined by any string $\sigma_1 \in \Sigma^*$ can be ε-bisimulated by the behavior of A_2 determined by a string $\sigma_2 \in \Sigma^*$. Making use of this fact we have the following sequence of inequalities where the first and third are a consequence of ε-bisimulation while the second is simply the definition of *IOS*.

$$\begin{aligned}
O(\mathsf{xduce}(A_1)(\sigma_1)) &\leq O(\mathsf{xduce}(A_2)(\sigma_2)) + \varepsilon \\
&\leq \max_{\sigma_2' \preceq \sigma_2} \{\gamma I(\sigma_2') - \eta(|\sigma_2| - |\sigma_2'|)\} \\
&\quad + \varepsilon \\
&\leq \max_{\sigma_1' \preceq \sigma_1} \{\gamma I(\sigma_1') + \gamma\varepsilon - \eta(|\sigma_1| - |\sigma_1'|)\} \\
&\quad + \varepsilon \\
&\leq \max_{\sigma_1' \preceq \sigma_1} \{\gamma I(\sigma_1') - \eta(|\sigma_1| - |\sigma_1'|)\} \\
&\quad + (\gamma + 1)\varepsilon.
\end{aligned}$$

The next result summarizes the previous discussion.

THEOREM 5. *Let A_1 be ε-approximate bisimilar to A_2. If A_1 is (γ, η)-IOS then A_2 is $((\gamma + 1)\varepsilon, \gamma, \eta)$-practical IOS.*

5.2 Relationship with Control Systems

Finally, we relate our notion of IOS with related notions for continuous time control systems. We assume the reader is familiar with standard control-theoretic notation, see e.g., [15]. We start with a notion of input-output stability for control systems (IODS) introduced in [12].

We consider nonlinear control systems of the form:

$$\dot{x} = f(x(t), u(t)), \quad x(0) = x_0 \tag{20}$$

where we assume that $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and the input function $u : \mathbb{R} \to \mathbb{R}^m$ have enough regularity properties to ensure existence and uniqueness of solutions. The trajectories of (20) with initial value x_0 at time $t = 0$ are denoted by $\varphi(t, x, u)$.

Let $|\cdot|$ denote the Euclidean norm, $|\cdot|_\infty$ the L_∞ norm for functions in $\mathcal{U}$, and for $t > 0$ and any measurable function $g : \mathbb{R} \to \mathbb{R}_0^+$, let $\operatorname{ess\,sup}_{\tau\in[0,t]} g(\tau)$ denote the essential supremum of g on $[0, t]$. The following definition is inspired by a similar definition in [12].

Definition 3. [IODS] The system (20) is called (γ, η)-*IO-dynamically stable* (IODS), if the inequality

$$|\varphi(t, x, u)| \leq \operatorname{ess\,sup}_{\tau\in[0,t]} \max\{0, \gamma|u(\tau)| - \eta(t - \tau)\}$$

holds for all $t \geq 0$, $x \in \mathbb{R}^n$, and all $u \in \mathcal{U}$.

The notion of IODS introduced in Definition 3 is a special case of Input-to-State Dynamical Stability (ISDS) introduced by Grüne in [12] as a quantitative version of the notion of Input-to-State Stability introduced by Sontag in [19]. This can be seen by defining the functions σ, γ, and μ in Definition 2.1 of [12] as:

$$\sigma \equiv \lambda r.\gamma\, r, \quad \gamma \equiv \lambda r.\gamma\, r, \quad \mu \equiv \lambda(r, s). \max\{0, r - \eta\, s\},$$

respectively. We also eliminated the distinction between overshoot gain and robustness gain in [12].

We now wish to relate IODS for a control system with the notion of practical IOS defined for discrete systems. First, we notice that the control system (20) defines an automaton $A_{dyn} = (\mathbb{R}^n, x_0, \mathcal{U}, \delta, \mathbb{R}^n, \lambda x.x)$, where $\delta(x, u) = x'$ if $\varphi(t, x, u) = x'$ for some $t \in \mathbb{R}_0^+$.[3] The results of [11, 18] show that under certain assumptions on the dynamics of (20), we can define a finite automaton that is ε-bisimilar to the automaton A_{dyn} obtained from the control system (20). Using Theorem 5, we obtain the following.

Theorem 6. *Let A_{dyn} be the automaton defined by* (20) *assumed to be (γ, η)-IODS and let A be an automaton that is ε-approximate bisimilar to A_{dyn}. Then A is $((\gamma + 1)\varepsilon, \gamma, \eta)$-practical IOS w.r.t. the cost functions $I(u) = |u|$ and $O(x) = |x|$.*

6. RELATED WORK

The notion of IO-stability discussed in this paper captures two very natural properties and it is natural to find different formalizations of related properties. For example, in fault-tolerant computation, one studies programs p that are F-tolerant for a class of faults F and a given specification S [3]. This definition requires the existence of a predicate T such that: 1) $S \Rightarrow T$; 2) T is invariant under program p and faults from F; 3) in the absence of faults, a computation that starts by satisfying T will eventually satisfy S. This notion can be captured in our setting by taking S to be the correct behaviors defined by $O_\infty(f(\sigma)) = 0$, by taking F to be the class of faults defined by $I(\sigma) \leq c$ for some $c \in \mathbb{N}$, and by taking T to be the set of streams satisfying $O_\infty(f(\sigma)) \leq \gamma\, c$. By definition of S and T, we clearly have $S \Rightarrow T$. Inequality (4) implies that T is invariant under the computations of f and faults, and inequality (5) implies that the system will eventually return to S. IOS not only captures the notion of fault-tolerance in [3], it *quantifies* it by providing a different set T for each disturbance bound c and a bound on the number of steps required to return to S as a function of c. Similarly, one can regard IOS as a quantitative version of the notion of self-stabilization introduced by Dijkstra in [8].

Closer to our work are the results reported in [20, 4, 9] that, as mentioned in the introduction, capture one but not all of the properties of IOS. Chaudhuri et al. [7] and Majumdar and Saha [16] have considered continuity properties of software implementations, checking that a deviation in a program's inputs cause a proportional deviation in its outputs. However, the notion of dissipation over time have not been considered in this setting.

As mentioned before, our notion of IOS is a simplification of the notion of ISDS introduced by Grüne in [12]. Although we believe these simplifications to be justified by polynomial-time verification algorithms (see Section 3), it remains to investigate if there are examples that require the full power of ISDS. The algorithms in Section 3.2 are inspired by the approach taken in the paper [13] to compute Lyapunov functions through viability theory. Although [13] only considered the verification problem, the extension to synthesis is conceptually clear. The algorithm for verification of IOS in Section 3.1 is inspired on the work of Alur et al. on verification and synthesis for ranking specifications [2]. Also related is the work reported in [14] where dynamic programing is used to characterize the stability properties of control systems. Although some of our algorithms have a dynamic programming flavor, the algorithms in [14] are not computationally efficient as they require solving a dynamic program for each different value of $I(\sigma)$.

Our use of finite-state automata as a means to define cost functions that depend on the evolution of states is inspired by similar representations in [1, 2, 6, 5]. The finite-state condition is usually needed for effective algorithms, and can be seen as abstractions of more general cost models.

7. CONCLUSION

We have provided a definition of robustness for discrete systems that satisfies two conditions: bounded disturbances have bounded consequences, and sporadic disturbances dissipate over time. We believe IOS forms a natural basis for developing a theory of robust behaviors. For example, one can prove "small gain" like theorems for the composition of systems.

Although many extensions of the proposed ideas are possible, we are particularly interested in providing a similar theory of robustness for infinite strings and liveness properties. Finally, while we study IOS in a discrete setting, the natural next step is to combine with IOS for continuous control systems [12]. This will allow reasoning about robustness of cyber-physical systems, in which discrete components interact with physical systems.

8. REFERENCES

[1] R. Alur and G. Weiss. RTComposer: a framework for real-time components with scheduling interfaces. In *EMSOFT 08: Embedded Software*, pages 159–168. ACM, 2008.

[2] Rajeev Alur, Aditya Kanade, and Gera Weiss. Ranking automata and games for prioritized

[3]Strictly speaking, the automata in Definition 2 are defined over finite state spaces. However, if we are not concerned with effective constructions, we can define automata over possibly infinite set of states (by dropping the finiteness requirement in the definition.

requirements. In *Proceedings of the 20th international conference on Computer Aided Verification*, CAV '08, pages 240–253, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] A. Arora and S.S. Kulkarni. Detectors and correctors: a theory of fault-tolerance components. In *Distributed Computing Systems, 1998. Proceedings. 18th International Conference on*, pages 436 –443, may 1998.

[4] R. Bloem, K. Greimel, T.A. Henzinger, and B. Jobstmann. Synthesizing robust systems. In *Formal Methods in Computer-Aided Design, 2009. FMCAD 2009*, pages 85 –92, nov. 2009.

[5] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *Logical Methods in Computer Science*, 6(3), 2010.

[6] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010.

[7] S. Chaudhuri, S. Gulwani, and R. Lublinerman. Continuity analysis of programs. In *POPL: Principles of Programming Languages*, pages 57–70. ACM, 2010.

[8] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.

[9] L. Doyen, T.A. Henzinger, A. Legay, and D. Nickovic. Robustness of sequential circuits. In *Application of Concurrency to System Design (ACSD), 2010 10th International Conference on*, pages 77 –84, june 2010.

[10] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of the 5th symposium on Operating systems design and implementation*, OSDI '02, pages 147–163. ACM, 2002.

[11] A. Girard. Approximately bisimilar finite abstractions of stable linear systems. *in Proceedings of 10th Internation Conference on Hybrid Systems: Computation and Control*, 4416:231–244, 2007.

[12] L. Grüne. Input-to-state dynamical stability and its lyapunov function characterization. *Automatic Control, IEEE Transactions on*, 47(9):1499 – 1504, sep 2002.

[13] L. Grüne and P. Saint-Pierre. An invariance kernel representation of ISDS Lyapunov functions. *Systems & Control Letters*, 55(9):736–745, 2006.

[14] S. Huang, M.R. James, D. Nesic, and P.M. Dower. Analysis of input-to-state stability for discrete time nonlinear systems via dynamic programming. *Automatica*, 41(12):2055–2065, 2005.

[15] H. K. Khalil. *Nonlinear systems.* Prentice-Hall, Inc., New Jersey, 2nd edition, 1996.

[16] R. Majumdar and I. Saha. Symbolic robustness analysis. In *IEEE Real-Time Systems Symposium*, pages 355–363. IEEE Computer Society, 2009.

[17] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E.W. Mayr and C. Puech, editors, *STACS 95: Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 900, pages 229–242. Springer-Verlag, 1995.

[18] G. Pola, A. Girard, and P. Tabuada. Approximately bisimilar symbolic models for nonlinear control systems. *Automatica*, 44(10):2508–2516, 2008.

[19] E.D. Sontag. Smooth stabilization implies coprime factorization. *Automatic Control, IEEE Transactions on*, 34(4):435 –443, apr 1989.

[20] Danielle C. Tarraf, Alexandre Megretski, and Munther A. Dahleh. A framework for robust stability of systems over finite alphabets. *IEEE Transactions on Automatic Control*, 53(5):1133–1146, 2008.

[21] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[22] G. Zames. Functional analysis applied to nonlinear feedback systems. *IEEE Transactions on Circuit Theory*, 10:392–404, 1963.

[23] G. Zames. Input-output feedback stability and robustness, 1959–85. *IEEE Control Systems Magazine*, 16(3):61–66, 1996.