Synthesizing Robust Software

Paulo Tabuada

Rupak Majumdar (MPI), Ayca Balkan, Sina Caliskan, Elaine Render, Matthias Rungger, Yasser Shoukry

Cyber-Physical Systems Laboratory
Department of Electrical Engineering
University of California at Los Angeles
Software systems are designed based on assumptions about their environment.
- Software systems are designed based on assumptions about their environment.
- But the real environment is either unknown at design time or changing over time.
Software systems are designed based on assumptions about their environment. But the real environment is either unknown at design time or changing over time. Hence, the assumptions will be violated and current design methodologies offer no assurance on how software behaves when such violations occur.
Robustness

- Software systems are designed based on assumptions about their environment.
- But the **real** environment is either unknown at design time or changing over time.
- Hence, the assumptions **will be** violated and current design methodologies offer no assurance on how software behaves when such violations occur.

  Ideally, we would like a modest deviation from the assumptions to lead to a modest deviation from the nominal correctness guarantees.
Software systems are designed based on assumptions about their environment. But the real environment is either unknown at design time or changing over time. Hence, the assumptions will be violated and current design methodologies offer no assurance on how software behaves when such violations occur.

Ideally, we would like a modest deviation from the assumptions to lead to a modest deviation from the nominal correctness guarantees.
Robustness

Our starting point:

- Robustness is a very familiar concept in control theory;
Robustness

Our starting point:

- Robustness is a very familiar concept in control theory;
- It is well understood that the models used for controller design (assumptions) are precious but always wrong:
  - Weight of a car (1 passenger vs 5 passengers);
  - Aerodynamic characteristics of a car (surfboard on the top of the car or bicycle mounted on a rack in the back);
  - etc.
Robustness

Our starting point:

- Robustness is a very familiar concept in control theory;
- It is well understood that the models used for controller design (assumptions) are precious but always wrong:
  - Weight of a car (1 passenger vs 5 passengers);
  - Aerodynamic characteristics of a car (surfboard on the top of the car or bicycle mounted on a rack in the back);
  - etc.
- The most basic controller designs do not explicitly address robustness, but they are robust against unmodeled disturbances.
Robustness

Our starting point:

- Robustness is a very familiar concept in control theory;
- It is well understood that the models used for controller design (assumptions) are precious but always wrong:
  - Weight of a car (1 passenger vs 5 passengers);
  - Aerodynamic characteristics of a car (surfboard on the top of the car or bicycle mounted on a rack in the back);
  - etc.
- The most basic controller designs do not explicitly address robustness, but they are robust against unmodeled disturbances.
- Can the same be done for software?
Robustness

What is known about software robustness?

In Computer Science:

- Recent work by Bloem, Chatterjee, Chaudhuri, Gulwani, Henzinger, Jobstman, Majumdar, ...
- Older work by Dijkstra (self-stabilizing algorithms).
Robustness
What is known about software robustness?

In Computer Science:

- Recent work by Bloem, Chatterjee, Chaudhuri, Gulwani, Henzinger, Jobstman, Majumdar, ...
- Older work by Dijkstra (self-stabilizing algorithms).

In Control Theory:

- There is a subfield of control theory called robust control;
- The following classification will be useful:
  - State based methods (modern view);
  - Input-output based methods (older view originated by the analysis of amplifiers and other circuits).
State based robustness

We start with a plain automaton.

Definition

A finite-state automaton is a triple \( A = (Q, \Sigma, \delta) \) consisting of:

- A finite set of states \( Q \);
- A finite set of actions \( \Sigma \);
- A transition function \( \delta : Q \times \Sigma \rightarrow Q \).
State based robustness

We start with a plain automaton.

**Definition**

A finite-state automaton is a triple $A = (Q, \Sigma, \delta)$ consisting of:

- A finite set of states $Q$;
- A finite set of actions $\Sigma$;
- A transition function $\delta : Q \times \Sigma \rightarrow Q$.

How to reason about modest deviations from the nominal behavior?
State based robustness

We introduce metric automata.

Definition

A finite-state metric automaton is a sextuple $A_\beta = (Q, d, \Sigma, X, \beta, \delta)$ consisting of:

- A finite set of states $Q$;
- A metric $d : Q \times Q \to \mathbb{R}_0^+$;
- A finite set of actions $\Sigma$;
- A finite set of environment actions $X$ including a special symbol $\epsilon$ denoting nominal (no disturbance) behavior;
- A parameter $\beta \in \mathbb{R}_0^+$ defining the “power” of the disturbance;
- A transition function $\delta : Q \times \Sigma \times X \to Q$. 
State based robustness

We introduce **metric** automata.

### Definition

A finite-state **metric** automaton is a sextuple $A_\beta = (Q, d, \Sigma, X, \beta, \delta)$ consisting of:

- A finite set of states $Q$;
- A metric $d : Q \times Q \rightarrow \mathbb{R}_0^+$;
- A finite set of actions $\Sigma$;
- A finite set of environment actions $X$ including a special symbol $\epsilon$ denoting nominal (no disturbance) behavior;
- A parameter $\beta \in \mathbb{R}_0^+$ defining the “power” of the disturbance;
- A transition function $\delta : Q \times \Sigma \times X \rightarrow Q$.

It seems that we are explicitly modeling the disturbances through the transition function $\delta$. 
We assume that we have:

$$d(\delta(q, \sigma, \epsilon), \delta(q, \sigma, x)) \leq \beta \quad \forall q \in Q, \sigma \in \Sigma, x \in X.$$
State based robustness

Disturbance model

We assume that we have:

\[ d(\delta(q, \sigma, \epsilon), \delta(q, \sigma, x)) \leq \beta \quad \forall q \in Q, \; \sigma \in \Sigma, \; x \in X. \]

The parameter \( \beta \) does not need to be known: results will be parameterized by \( \beta \).
State based robustness

**Definition**

A winning strategy \((S : Q \rightarrow 2^\Sigma)\) for the automaton \(A_0\) and (reachability or Büchi) winning objective \(F \subseteq Q\) is \(\gamma\)-robust if for any \(\beta \in \mathbb{R}_0^+\) it is winning for the automaton \(A_{\beta}\) with (reachability, Büchi) winning objective \(B_{\gamma\beta}(F)\):

\[
B_{\gamma\beta}(F) = \{q \in Q \mid d(q, F) \leq \gamma\beta\}.
\]
State based robustness

Definition

A winning strategy \( (S : Q \rightarrow 2^\Sigma) \) for the automaton \( A_0 \) and (reachability or Büchi) winning objective \( F \subseteq Q \) is \( \gamma \)-robust if for any \( \beta \in \mathbb{R}^+_0 \) it is winning for the automaton \( A_\beta \) with (reachability, Büchi) winning objective \( \mathcal{B}_{\gamma \beta}(F) \):

\[
\mathcal{B}_{\gamma \beta}(F) = \{ q \in Q \mid d(q, F) \leq \gamma \beta \}.
\]

- Note that if there are no disturbances, \( \beta = 0 \) and \( \mathcal{B}_{\gamma \beta}(F) = F \).
Definition

A winning strategy \((S : Q \rightarrow 2^\Sigma)\) for the automaton \(A_0\) and (reachability or Büchi) winning objective \(F \subseteq Q\) is \(\gamma\)-robust if for any \(\beta \in \mathbb{R}^+_0\) it is winning for the automaton \(A_\beta\) with (reachability, Büchi) winning objective \(B_{\gamma\beta}(F)\):

\[
B_{\gamma\beta}(F) = \{ q \in Q \mid d(q, F) \leq \gamma\beta \}.
\]

- Note that if there are no disturbances, \(\beta = 0\) and \(B_{\gamma\beta}(F) = F\).
- The parameter \(\gamma\) describes how much \(F\) is inflated to obtain \(B_{\gamma\beta}(F)\).
State based robustness

**Definition**

A winning strategy $(S: Q \rightarrow 2^\Sigma)$ for the automaton $A_0$ and (reachability or Büchi) winning objective $F \subseteq Q$ is $\gamma$-robust if for any $\beta \in \mathbb{R}_0^+$ it is winning for the automaton $A_\beta$ with (reachability, Büchi) winning objective $B_{\gamma\beta}(F)$:

$$B_{\gamma\beta}(F) = \{ q \in Q \mid d(q, F) \leq \gamma\beta \}.$$

- Note that if there are no disturbances, $\beta = 0$ and $B_{\gamma\beta}(F) = F$.
- The parameter $\gamma$ describes how much $F$ is inflated to obtain $B_{\gamma\beta}(F)$.
- The map transforming environment strategies to the language accepted by $A_\beta$ is uniformly continuous with modulus of continuity $\gamma$. 
State based robustness
Verification and synthesis

Given an automaton $A_0$, $\gamma \in \mathbb{R}_0^+$, and a strategy $S$ one can ask:

- **Verification**: Is $S$ $\gamma$-robust?
- **Optimal verification**: What is the smallest $\gamma \in \mathbb{R}_0^+$ for which $S$ is $\gamma$-robust?
Given an automaton $A_0$, $\gamma \in \mathbb{R}_0^+$, and a strategy $S$ one can ask:

- **Verification**: Is $S$ $\gamma$-robust?
- **Optimal verification**: What is the smallest $\gamma \in \mathbb{R}_0^+$ for which $S$ is $\gamma$-robust?
- **Synthesis**: Can we synthesize a $\gamma$-robust strategy?
- **Optimal synthesis**: What is the smallest $\gamma \in \mathbb{R}_0^+$ for which we can synthesize a $\gamma$-robust strategy?
State based robustness
Verification and synthesis

Given an automaton $A_0$, $\gamma \in \mathbb{R}_0^+$, and a strategy $S$ one can ask:

- **Verification**: Is $S$ $\gamma$-robust?
- **Optimal verification**: What is the smallest $\gamma \in \mathbb{R}_0^+$ for which $S$ is $\gamma$-robust?
- **Synthesis**: Can we synthesize a $\gamma$-robust strategy?
- **Optimal synthesis**: What is the smallest $\gamma \in \mathbb{R}_0^+$ for which we can synthesize a $\gamma$-robust strategy?

All the above problems can be reduced to dynamic programming and are thus polynomially solvable.
On the positive side, state based robustness:

- handles $\omega$-regular objectives (Büchi and parity) as a simple extension of reachability;
- the analysis makes use of an extension of rank functions inspired on control techniques (Lyapunov functions).
On the positive side, state based robustness:

- handles $\omega$-regular objectives (Büchi and parity) as a simple extension of reachability;
- the analysis makes use of an extension of rank functions inspired on control techniques (Lyapunov functions).

On the less positive side, state based robustness:

- requires a metric. What if I have two different automata defining the same language? How to reason about robustness before having an implementation with states?
Input/output based robustness
Towards a definition

- Rather than automata we now consider transducers $f : \Sigma^* \rightarrow \Lambda^*$;
- Rather than a metric we use cost functions $I : \Sigma^* \rightarrow \mathbb{N}^+$ and $O : \Lambda^* \rightarrow \mathbb{N}^+$ to place costs on input and output strings, respectively;

Well known requirements in control theory that recently appeared as two separate notions of robustness proposed by Henzinger and co-workers.
Rather than automata we now consider transducers $f : \Sigma^* \rightarrow \Lambda^*$;

Rather than a metric we use cost functions $I : \Sigma^* \rightarrow \mathbb{N}_0^+$ and $O : \Lambda^* \rightarrow \mathbb{N}_0^+$ to place costs on input and output strings, respectively;
Rather than automata we now consider transducers $f : \Sigma^* \rightarrow \Lambda^*$;

Rather than a metric we use cost functions $I : \Sigma^* \rightarrow \mathbb{N}_0^+$ and $O : \Lambda^* \rightarrow \mathbb{N}_0^+$ to place costs on input and output strings, respectively;

A notion of robustness should have the following two properties:

- Bounded disturbances should lead to bounded consequences;
Input/output based robustness
Towards a definition

- Rather than automata we now consider transducers $f : \Sigma^* \rightarrow \Lambda^*$;
- Rather than a metric we use cost functions $I : \Sigma^* \rightarrow \mathbb{N}_0^+$ and $O : \Lambda^* \rightarrow \mathbb{N}_0^+$ to place costs on input and output strings, respectively;
- A notion of robustness should have the following two properties:
  - Bounded disturbances should lead to bounded consequences;
  - The effect of a sporadic disturbance should disappear in finitely many steps;
Rather than automata we now consider transducers $f : \Sigma^* \rightarrow \Lambda^*$; 

Rather than a metric we use cost functions $I : \Sigma^* \rightarrow \mathbb{N}_0^+$ and $O : \Lambda^* \rightarrow \mathbb{N}_0^+$ to place costs on input and output strings, respectively;

A notion of robustness should have the following two properties:

- Bounded disturbances should lead to bounded consequences;
- The effect of a sporadic disturbance should disappear in finitely many steps;
- Well known requirements in control theory that recently appeared as two separate notions of robustness proposed by Henzinger and co-workers.
Input/output based robustness

A definition

Adapting and simplifying the control theoretic notion of Input-to-State Dynamic Stability we propose:

Definition

Given parameters \( \gamma, \eta \in \mathbb{N} \), we say the transducer \( f : \Sigma^* \rightarrow \Lambda^* \) is \((\gamma, \eta)\)-input-output stable if for each \( \sigma \in \Sigma^* \) we have

\[
O(f(\sigma)) \leq \max_{\sigma' \preceq \sigma} \{ \gamma I(\sigma') - \eta (|\sigma| - |\sigma'|) \}.
\]

The parameter \( \gamma \) is called the robustness gain. It measures how much the disturbance is amplified.

The parameter \( \eta \) is called the rate of decay. It measures how quickly the effects of a disturbance disappear.

The notion of \((\gamma, \eta)\)-input-output stability captures the two desired properties.

The verification and synthesis problems are polynomially solvable when \( f, I, \) and \( O \) are described by finite-state (cost/weighted) automata;
Adapting and simplifying the control theoretic notion of Input-to-State Dynamic Stability we propose:

**Definition**

Given parameters $\gamma, \eta \in \mathbb{N}$, we say the transducer $f : \Sigma^* \rightarrow \Lambda^*$ is $(\gamma, \eta)$-input-output stable if for each $\sigma \in \Sigma^*$ we have

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

- The parameter $\gamma$ is called the *robustness gain*. It measures how much the disturbance is amplified.
Input/output based robustness

A definition

Adapting and simplifying the control theoretic notion of Input-to-State Dynamic Stability we propose:

**Definition**

Given parameters $\gamma, \eta \in \mathbb{N}$, we say the transducer $f : \Sigma^* \rightarrow \Lambda^*$ is $(\gamma, \eta)$-*input-output stable* if for each $\sigma \in \Sigma^*$ we have

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

- The parameter $\gamma$ is called the *robustness gain*. It measures how much the disturbance is amplified.
- The parameter $\eta$ is called the *rate of decay*. It measures how quickly the effects of a disturbance disappear.
Input/output based robustness
A definition

Adapting and simplifying the control theoretic notion of Input-to-State Dynamic Stability we propose:

**Definition**

Given parameters $\gamma, \eta \in \mathbb{N}$, we say the transducer $f : \Sigma^* \rightarrow \Lambda^*$ is $(\gamma, \eta)$-*input-output stable* if for each $\sigma \in \Sigma^*$ we have

$$O(f(\sigma)) \leq \max_{\sigma' \leq \sigma} \{ \gamma I(\sigma') - \eta (|\sigma| - |\sigma'|) \}.$$ 

- The parameter $\gamma$ is called the *robustness gain*. It measures how much the disturbance is amplified.
- The parameter $\eta$ is called the *rate of decay*. It measures how quickly the effects of a disturbance disappear.
- The notion of $(\gamma, \eta)$-input-output stability captures the two desired properties.
Adapting and simplifying the control theoretic notion of Input-to-State Dynamic Stability we propose:

**Definition**

Given parameters $\gamma, \eta \in \mathbb{N}$, we say the transducer $f : \Sigma^* \rightarrow \Lambda^*$ is $(\gamma, \eta)$-input-output stable if for each $\sigma \in \Sigma^*$ we have

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

- The parameter $\gamma$ is called the *robustness gain*. It measures how much the disturbance is amplified.
- The parameter $\eta$ is called the *rate of decay*. It measures how quickly the effects of a disturbance disappear.
- The notion of $(\gamma, \eta)$-input-output stability captures the two desired properties.
- The verification and synthesis problems are polynomially solvable when $f$, $I$, and $O$ are described by finite-state (cost/weighted) automata;
Issues for discussion:

- It should be possible to robustify any synthesis methodology.
- How to migrate these ideas from automata and transducers to programing languages in order to make them usable by programmers?
- Specifying robustness requires the programer to rank all the possible environment behaviors and all the possible program behaviors. This is not always easy to do!
Issues for discussion:

- It should be possible to robustify any synthesis methodology.
- How to migrate these ideas from automata and transducers to programming languages in order to make them usable by programmers?
- Specifying robustness requires the programmer to rank all the possible environment behaviors and all the possible program behaviors. This is not always easy to do!

I am looking forward to working with all of you to make robustness practical.
Robustness

Relevant recent references:

- *Robust Discrete Synthesis Against Unspecified Disturbances*
  Rupak Majumdar, Elaine Render, and Paulo Tabuada
  14th International Conference on Hybrid Systems: Computation and Control 2011.

- A theory of $\omega$-regular robust software synthesis
  Rupak Majumdar, Elaine Render, and Paulo Tabuada

- *Input-Output Stability for Discrete Systems*
  Paulo Tabuada, Ayca Balkan, Sina Caliskan, Yasser Shoukry, and Rupak Majumdar
  Submitted to EMSOFT 2012.

For preprints send an email to tabuada@ee.ucla.edu.