Falsification of Cyber-Physical Systems with Robustness-Guided Black-Box Checking

Masaki Waga$^{1,2,3}$

National Institute of Informatics$^{1}$, SOKENDAI$^{2}$, JSPS Research Fellow$^{3}$

April 2020, HSCC 2020

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), by JSPS Grants-in-Aid No. 15KT0012 & 18J22498
Testing is Important...

Tesla Model 3: Autopilot engaged during fatal crash

0 17 May 2019

The Tesla Model 3 after the crash

Top Stories

Trump lashes out at impeachment inquiry
Using a vulgarity, the Republican president accuses Democrats of dishonesty and even treason.
7 hours ago

'Grave moment' as N Korea tests submarine missile
27 minutes ago

HK rubber bullet blinders journalist in one eye
2 October 2019

Features

Testing is Important…

Tesla Model 3: Autopilot engaged during fatal crash

Falsification: Robustness + Optimization

\[ \rho(\mathcal{M}(\sigma, \phi)) = 30 \]

\[ \rho(\mathcal{M}(\sigma, \phi)) = -10 \]

Robustness Monitor
Spec: \( \phi \) in STL
(signal temporal logic)

Optimizer
(Minimize Robustness)
Falsification: Robustness + Optimization

\[ \rho(M(\sigma, \phi)) = 30 \]
\[ \rho(M(\sigma)) = -10 \]

Optimization

Optimizer
(Minimize Robustness)

Robustness Monitor
Spec: \( \varphi \) in STL
(signal temporal logic)
Falsification: Robustness + Optimization

\[ M(\sigma) = 30 \]

\[ \rho(M(\sigma), \phi) = -10 \]

Optimization

Robustness Monitor

Spec: \( \phi \) in STL (signal temporal logic)
Falsification: Robustness + Optimization

\[ \rho(\mathcal{M}(\sigma), \varphi) = 30 \]

Optimizer
(Minimize Robustness)

Robustness Monitor
Spec: \( \varphi \) in STL
(signal temporal logic)
Falsification: Robustness + Optimization

\[ \rho(\mathcal{M}(\sigma), \varphi) = 30 \]

Spec: \( \varphi \) in STL (signal temporal logic)

Optimizer (Minimize Robustness)

Robustness Monitor

M. Waga (NII)
Falsification: Robustness + Optimization

\[ \rho(\mathcal{M}(\sigma), \varphi) = 30 \]

\[ \rho(\mathcal{M}(\sigma), \varphi) = -10 \]

Robustness Monitor
Spec: \( \varphi \) in STL (signal temporal logic)

Optimizer
(Minimize Robustness)
Robustness of STL

<table>
<thead>
<tr>
<th>Boolean</th>
<th>Robust</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \leq 100$ $\not\models \square x &gt; 10$</td>
<td>$\rho\left(100, \square x &gt; 10\right) = 90$</td>
</tr>
<tr>
<td>$x \leq 15$ $\models \square x &gt; 10$</td>
<td>$\rho\left(15, \square x &gt; 10\right) = 5$</td>
</tr>
<tr>
<td>$x \leq 5$ $\not\models \square x &gt; 10$</td>
<td>$\rho\left(5, \square x &gt; 10\right) = -5$</td>
</tr>
<tr>
<td>$x \leq -60$ $\not\models \square x &gt; 10$</td>
<td>$\rho\left(-60, \square x &gt; 10\right) = -70$</td>
</tr>
</tbody>
</table>
Falsification is Oneshot

\[\mathcal{M}\]

\[\mathcal{M}(\sigma)\]

Spec: \(\varphi_1\)

Optimizer (Minimize Robustness)

\[\rho(\varphi_1, \sigma) = 30\]

Robustness Monitor Spec: \(\varphi_1\)

Spec. 1

\[\rho(\varphi_2, \sigma) = 75\]

Robustness Monitor Spec: \(\varphi_2\)

Spec. 2

M. Waga (NII)
Falsification is Oneshot

Optimizer (Minimize Robustness)

Robustness Monitor Spec: $\phi_1$

$\rho(\phi_1, \sigma) = 30$

Independent $\rightarrow$ No Reused Info.

Robustness Monitor Spec: $\phi_2$

$\rho(\phi_2, \sigma) = 75$

M. Waga (NII)
Falsification is Oneshot

Optimizer (Minimize Robustness)

$\rho(\varphi, \sigma) = 30$

Robustness Monitor Spec: $\varphi$

Version 1

$\sigma$

Optimizer (Minimize Robustness)

$\rho(\varphi, \sigma) = 75$

Robustness Monitor Spec: $\varphi$

Version 2

M. Waga (NII)
Falsification is Oneshot

Optimizer (Minimize Robustness)

Robustness Monitor
Spec: \( \varphi \)

\( \rho(\varphi, \sigma) = 30 \)

\( \rho(\varphi, \sigma) = 75 \)

Independent \( \rightarrow \) No Reused Info.
Q. How to reuse the previous exploration information
Q. How to reuse the previous exploration information

Our Approach:
Black-Box Checking (BBC)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$ ?

$M(u) \not\models \varphi$

No. cex $u$

Yes

$\tilde{M} \approx M$

Yes

No. cex $u$

No

$M(u) \not\models \varphi$

Yes

$\tilde{M} \models \varphi$

Yes
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$?

$M(u) \not\models \varphi$

No. cex $u$

No

Yes

Validate $\tilde{M}$

$\tilde{M} \approx M$

Yes

No

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

$\tilde{M} \models \varphi$

Yes

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$?

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

Validate $\tilde{M}$

Reuse $\tilde{M}$

$M(u) \not\models \varphi$

No. cex $u$

No

Yes

$\tilde{M} \approx M$

Yes

No

$\tilde{M} \approx M$

Yes

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$ ?

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

Validate $\tilde{M}$

Reuse $\tilde{M}$

$M(\mathcal{U}) \nvdash \varphi$

No. cex $\mathcal{U}$

$\checkmark + \tilde{M}$

$\tilde{M} \approx M$

$\times + \tilde{M}$

No

Yes

No

Yes

Difficult

[Peled+, FORTE’99]
Equivalence Test: $\tilde{M} \approx M$

- **Theoretically**: Automata-based conformance test e.g., W-method [Chow, TSE’78], Wp-method [Fujiwara+, TSE’91]
  - 😞 # of states is necessary for soundness

- **Practically**:
  - Random test
  - 😞 Not good at “rare” cex
Equivalence Test: $\tilde{M} \approx M$

- **Theoretically**: Automata-based conformance test e.g., $W$-method [Chow, TSE’78], $W_p$-method [Fujiwara+, TSE’91]
  - 😞 # of states is necessary for soundness

- ** Practically**:
  - Random test
  - 😞 Not good at “rare” cex

Q. How about hitting “rare” cex with robustness of STL?
Robustness-Guided Equivalence Test

- Equivalence Test with Robustness + Optimization
  - Search for $u$ s.t. $\tilde{M}(u) \neq M(u)$
  - Genetic algorithm or hill climbing
  - Fitness function: Robustness of STL

- $\tilde{M}$ is not directly evaluated
  - Discrete behavior of $\tilde{M}$ is not a problem

[Contribution]

Detail is later
Contributions

• Presented robustness-guided equivalence test
  • Also introduced a new robustness suitable to model checking

• Used it for black-box checking of CPS

• Experimental evaluation: outperforms the baseline
Outline

• Introduction

• Preliminary: Black-box checking
  • Automata learning, model checking, & equiv. test

• Main Contribution:
  Robustness-guided equivalence test
  • idea: Optimization + robustness

• Experiments
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \phi$?

Yes

Exploit $\tilde{M}$

No

Approximate $M$ by autom. $\tilde{M}$

$M(u) \not\models \phi$

No. cex $u$

Validated $\tilde{M}$

Yes

$\tilde{M} \approx M$

Yes

$\tilde{M}$ + $\checkmark$

No

$\times$ + $\tilde{M}$

$\times$ + $\checkmark$

M. Waga (NII)

[Peled+, FORTE’99]
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$?

No. cex $u$

$M(u) \not\models \varphi$

No

$\tilde{M} \approx M$

No

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

Validate $\tilde{M}$

Yes

Yes

Validate $\tilde{M}$

Yes

Yes

Yes

[Peled+, FORTE'99]
Automata Learning

1. Obtain suitable input/output pairs \((u, M(u))\) of \(M\)
   • Input/output: finitely discretized

2. Generate a candidate Myhill-Nerode equiv. rel.

3. Construct the corresponding Mealy machine \(\hat{M}\)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \phi$?

No. cex $u$

$M(u) \not\models \varphi$

No

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

Yes

$\tilde{M} \approx M$

Yes

Validate $\tilde{M}$

$\times + \tilde{M}$

$\checkmark + \tilde{M}$

[Peled+, FORTE’99]

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. \( \widetilde{\mathcal{M}} \) from CPS \( \mathcal{M} \)

Model Check \( \widetilde{\mathcal{M}} \models \varphi \) ?

Approximate \( \mathcal{M} \) by autom. \( \widetilde{\mathcal{M}} \)

Exploit \( \widetilde{\mathcal{M}} \)

Yes

No

No. cex \( u \)

\( \mathcal{M}(u) \not\models \varphi \)

No

Yes

Validate \( \widetilde{\mathcal{M}} \)

\( \mathcal{M}(u) \models \varphi \)

Yes

\( \widetilde{\mathcal{M}} \approx \mathcal{M} \)

\( \checkmark + \widetilde{\mathcal{M}} \)

[Peled+, FORTE'99]
Model Check: \( \tilde{M} \models \varphi \) ?

**Issue**: \( \varphi \) is not in LTL but in STL

**Our Solution**: Encode to LTL model checking

### STL

\[
\varphi ::= x \otimes c \mid \varphi \land \varphi \mid \cdots \mid \varphi \cup_i \varphi
\]

**Assumption**: signal is discrete-time with constant interval

### LTL

\[
\varphi ::= P \mid \varphi \land \varphi \mid \cdots \mid \varphi \cup \varphi
\]

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. \( \tilde{\mathcal{M}} \) from CPS \( \mathcal{M} \)

Model Check \( \tilde{\mathcal{M}} \models \varphi ? \)

No. cex \( u \)

\( \mathcal{M}(u) \not\models \varphi \)

Yes

\( \tilde{\mathcal{M}} \approx \mathcal{M} \)

Validated \( \tilde{\mathcal{M}} \)

Approximate \( \mathcal{M} \) by autom. \( \tilde{\mathcal{M}} \)

Exploit \( \tilde{\mathcal{M}} \)

\( \mathcal{M} \)

No

Yes

[Peled+, FORTE'99]

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{\mathcal{M}}$
from CPS $\mathcal{M}$

Model Check $\tilde{\mathcal{M}} \vdash \phi$?

No. cex $u$

$\mathcal{M}(u) \not\models \phi$

Yes

$\tilde{\mathcal{M}}(u) \not\models \phi$

Yes

Validate $\tilde{\mathcal{M}}$

Approximate $\mathcal{M}$ by autom. $\tilde{\mathcal{M}}$

Exploit $\tilde{\mathcal{M}}$

$\tilde{\mathcal{M}} \approx \mathcal{M}$

Yes

$\checkmark + \tilde{\mathcal{M}}$

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{\mathcal{M}}$ from CPS $\mathcal{M}$

Model Check $\tilde{\mathcal{M}} \models \varphi$?

No cex $u$

$\mathcal{M}(u) \not\models \varphi$

No

Approximate $\mathcal{M}$ by autom. $\tilde{\mathcal{M}}$

Exploit $\tilde{\mathcal{M}}$

Yes

$\tilde{\mathcal{M}} \approx \mathcal{M}$

Validate $\tilde{\mathcal{M}}$

Yes

$\checkmark$ + $\tilde{\mathcal{M}}$

[Peled+, FORTE’99]
Black-Box Checking (BBC)

Learn Autom. $\tilde{\mathcal{M}}$ from CPS $\mathcal{M}$

Model Check $\tilde{\mathcal{M}} \models \varphi$?

Approximate $\mathcal{M}$ by autom. $\tilde{\mathcal{M}}$

Exploit $\tilde{\mathcal{M}}$

No. cex $u$

No

Yes

$\mathcal{M}(u) \not\models \varphi$

$\tilde{\mathcal{M}}(u) \not\models \varphi$

Simulation

$\mathcal{M}(u) \models \varphi$

$\tilde{\mathcal{M}}(u) \models \varphi$

$\tilde{\mathcal{M}} \approx \mathcal{M}$

[Peled+, FORTE’99]

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$?

≈ $\tilde{M}$

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

$\tilde{M}(u) \not\models \varphi$

Yes

No. cex $u$

$M(u) \not\models \varphi$

$\tilde{M} \approx M$

Validate $\tilde{M}$

Yes

No

$\not\exists$ cex $u$

$M(u) \models \varphi$

Yes

$\not\exists$ cex $u$

$M(u) \not\models \varphi$

Yes

No

$\not\exists$ cex $u$

$M(u) \models \varphi$

Yes

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. $\tilde{M}$ from CPS $M$

Model Check $\tilde{M} \models \varphi$?

Approximate $M$ by autom. $\tilde{M}$

Exploit $\tilde{M}$

Yes

Yes

$\tilde{M} \approx M$

Validate $\tilde{M}$

$\tilde{M}(u) \not\models \varphi$

$M(u) \not\models \varphi$

No.

No.

No. cex $u$

No

Yes

Yes

$\tilde{M}(u) \not\models \varphi$

$M(u) \not\models \varphi$

$\tilde{M}(u) \models \varphi$

$M(u) \models \varphi$

$\tilde{M} \models \varphi$

$M \models \varphi$

[Peled+, FORTE’99]

M. Waga (NII)
Black-Box Checking (BBC)

Learn Autom. \( \tilde{M} \) from CPS \( M \)

Model Check \( \tilde{M} \Vdash \varphi ? \)

- Yes
  - \( \tilde{M}(u) \Vdash \varphi \)
    - Validate \( \tilde{M} \)
  - \( \tilde{M} \approx M \)
    - \( \checkmark + \tilde{M} \)
  - Difficult
    - No
- \( M(u) \not\Vdash \varphi \)
  - No. cex \( u \)
  - \( \checkmark + \tilde{M} \)
  - Approximate \( M \) by autom. \( \tilde{M} \)
  - Exploit \( \tilde{M} \)

[Peled+, FORTE’99]

M. Waga (NII)
Outline

• Introduction

• Preliminary: Black-box checking
  • Automata learning, model checking, & equiv. test

• Main Contribution:
  Robustness-guided equivalence test
  • idea: Optimization + robustness

• Experiments
Robustness-Guided Equivalence Test

Contributed

General population $U$

$U := U'$

$\forall u \in U . \tilde{M}(u) = M(u) ?$

Yes

General next population $U'$

Selection: $u' \in U'$ s.t.

$\rho(M(u'), \varphi)$ is small

No

Return $u$
Robustness-Guided Equivalence Test

[Contribution]

Gen. population $U$

$U := U'$

$\forall u \in U. \, \tilde{M}(u) = M(u)$?

Yes

Gen. next population $U'$

Selection: $u' \in U'$ s.t.

$\rho(\tilde{M}(u'), \varphi)$ is small

No

Return $u$
Robustness-Guided Equivalence Test

[Contribution]

Gen. population $U$

$U := U'$

$\forall u \in U . \tilde{M}(u) = M(u)$ ?

Yes

Gen. next population $U'$

Selection: $u' \in U'$ s.t.

$\rho(M(u'), \phi)$ is small

No

Return $u$
Robustness-Guided Equivalence Test

[Contribution]

Gen. population $U$

$U := U'$

$\forall u \in U . \tilde{M}(u) = M(u)$?

Yes

Gen. next population $U'$

Selection: $u' \in U'$ s.t. $\rho(M(u'), \varphi)$ is small

No

Return $u$

GA: Crossover & Mutation
HC: Neighbors

Only Continuous

M. Waga (NII)
Why $\rho(\mathcal{M}(u'), \varphi)$ as Fitness?

**Assumption:** $\mathcal{M} \not\models \varphi$

i.e. $\exists u. \rho(\mathcal{M}(u), \varphi) \leq 0$

**Fact:** $\tilde{\mathcal{M}} \models \varphi$ i.e.

$\forall u. \rho(\tilde{\mathcal{M}}(u), \varphi) \geq 0$

by model checking

**Heuristic:** Find $u'$ s.t.

$\tilde{\mathcal{M}}(u') \neq \mathcal{M}(u')$ in

$\rho(\tilde{\mathcal{M}}(u'), \varphi)$ is small
Why $\rho(\mathcal{M}(u'), \varphi)$ as Fitness?

**Assumption:** $\mathcal{M} \not\models \varphi$

i.e. $\exists u. \rho(\mathcal{M}(u), \varphi) \leq 0$

**Fact:** $\tilde{\mathcal{M}} \models \varphi$ i.e.

$\forall u. \rho(\mathcal{M}(u), \varphi) \geq 0$

by model checking

**Heuristic:** Find $u'$ s.t.

$\tilde{\mathcal{M}}(u') \neq \mathcal{M}(u')$ in

$\rho(\mathcal{M}(u'), \varphi)$ is small

$\mathcal{M}$ and $\tilde{\mathcal{M}}$ deviate
How About Direct Comparison?

- Direct comparison is also available e.g. [Abbas+, MEMOCODE’14]

- **Potential Issue:** more local maxima

---

<table>
<thead>
<tr>
<th>Cont. vs Cont.</th>
<th>Cont. vs Disc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td><strong>Output</strong></td>
</tr>
<tr>
<td>$\mathcal{M}(u)$</td>
<td>$\mathcal{M}(u)$</td>
</tr>
<tr>
<td>$\mathcal{M}'(u)$</td>
<td>$\mathcal{M}'(u)$</td>
</tr>
<tr>
<td><strong>Input</strong></td>
<td><strong>Input</strong></td>
</tr>
</tbody>
</table>

M. Waga (NII)
Outline

• Introduction

• Preliminary: Black-box checking
  • Automata learning, model checking, & equiv. test

• Main Contribution:
  Robustness-guided equivalence test
  • **idea:** Optimization + robustness

• Experiments
Research Questions

**RQ1:** Does RobBBC falsify as many spec. as baseline?

**RQ2:** Which BBC is the best: Random, GA, and HC?

**RQ3:** Scalable wrt # of spec?
Experiment Setting: Algorithms

- Robustness-guided BBC with
  - Genetic Algorithm (RobBBC(GA))
  - Hill Climbing (RobBBC(HC))

- BBC (equivalence test by random search)
- Breach: one of the SoTA falsification tools
- Pure random search
  - show the difficulty of the specifications
Experiment Setting: Benchmark

**Model**: Automatic Transmission [Hoxha+, ARCH’14-15]

**Spec**: Set of STL formulas

- Similar specifications with different thresholds
- Motivation: hard to decide the threshold

<table>
<thead>
<tr>
<th>STL template</th>
<th>parameter valuations</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi_1$ $\square (v &lt; p)$</td>
<td>$p \in {100, 102.5, 105, 107.5, 110, 112.5, 115, 117.5, 120}$</td>
<td>9</td>
</tr>
<tr>
<td>$\varphi_2$ $\square (g = 3 \Rightarrow v &gt; p)$</td>
<td>$p \in {20, 22.5, 25, 27.5, 30}$</td>
<td>5</td>
</tr>
<tr>
<td>$\varphi_3$ $\Diamond_{[p_1,p_2]} (v &lt; p_3 \lor v &gt; p_4)$</td>
<td>$(p_1, p_2) \in {(5, 20), (5, 25), (15, 30), (10, 30)}$</td>
<td>8</td>
</tr>
<tr>
<td>$\varphi_4$ $\square_{[0,26]} (v &lt; p_1) \lor \square_{[28,28]} (v &gt; p_2)$</td>
<td>$(p_3, p_4) \in {(50, 60), (53, 57)}$</td>
<td>9</td>
</tr>
<tr>
<td>$\varphi_5$ $\square (\omega &lt; p_1 \lor \Diamond (\omega &gt; p_2))$</td>
<td>$p_1 \in {90, 100, 110}$, $p_2 \in {55, 65, 75}$</td>
<td>6</td>
</tr>
<tr>
<td>$\varphi_6$ $\square (v &lt; p_1 \Rightarrow \square_{[0,p_2]} (v &lt; p_3))$</td>
<td>$p_1 \in {4000, 4700}$, $p_2 \in {600, 1000, 1500}$</td>
<td>36</td>
</tr>
<tr>
<td>$\varphi_7$ $\square ((g \neq p_1) \land \Diamond (g = p_1)) \Rightarrow \square_{[0,p_2]} (g = p_1)$</td>
<td>$p_1 \in {30, 40, 50}$, $p_2 \in {6, 8, 10}$, $p_3 \in {60, 70, 80, 90}$</td>
<td>12</td>
</tr>
</tbody>
</table>

M. Waga (NII)
• RobBBC(HC) and Breach: Often fails
• Random BBC: sometimes fails
• RobBBC(GA): the largest for all specifications
# of Falsified Properties

# of Properties

- RobBBC(HC) and Breach: Often fails
- Random BBC: sometimes fails
- RobBBC(GA): the largest for all specifications

RQ1: Yes, RobBBC can falsify as many as Breach
• RobBBC(HC) and Breach: Often fails
• Random BBC: sometimes fails
• RobBBC(GA): the largest for all specifications

RQ1: Yes, RobBBC can falsify as many as Breach
RQ2: GA falsified more than Random and HC...
Time to Falsify by the Fastest Method

Time to Falsify

- RobBBC(GA) tends not be the fastest when others are available → not the best for easy spec.
• RobBBC(GA) tends not to be the fastest when others are available → not the best for easy spec.

RQ2: GA falsified more than Random HC
For easy spec. Random BBC can be an option
Scalability wrt # of Spec. in $\varphi_6$

RQ3: BBC is more scalable than Breach wrt # of Spec.
Conclusion

• Introduced robustness-guided BBC
  • Equivalence test using robustness
• Experimental evaluation
  • It outperforms Breach
Conclusion

- Introduced robustness-guided BBC
- Equivalence test using robustness
- Experimental evaluation
- It outperforms Breach

But this is the beginning of the story…
Direct Reuse of \( \tilde{M} \) Does Not Work

Model checking generates **false** counter examples

<table>
<thead>
<tr>
<th>STL formula ( \varphi )</th>
<th># of ( \varphi \not\models \tilde{M} )</th>
<th># of ( \varphi \not\models M )</th>
<th>Average of ( \llbracket \varphi \rrbracket )</th>
<th>std. dev. of ( \llbracket \varphi \rrbracket )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Box (v &lt; 90) )</td>
<td>10</td>
<td>5</td>
<td>1.10</td>
<td>1.94</td>
</tr>
<tr>
<td>( \Box [0,26] (v &lt; 90) \lor \Box [28,28] (v &gt; 40) )</td>
<td>4</td>
<td>0</td>
<td>4.19</td>
<td>0.00</td>
</tr>
<tr>
<td>( \Box [0,26] (v &lt; 90) \lor \Box [28,28] (v &gt; 50) )</td>
<td>10</td>
<td>0</td>
<td>3.80</td>
<td>0.60</td>
</tr>
<tr>
<td>( \Box [0,26] (v &lt; 90) \lor \Box [28,28] (v &gt; 60) )</td>
<td>10</td>
<td>0</td>
<td>3.24</td>
<td>0.76</td>
</tr>
</tbody>
</table>

But robustness is low
Future Work

• How to (re-)use the generated Mealy machine?

• Black-box checking for multiple systems

  • Adaptive model checking?

• Try more benchmarks to figure out further issues
Appendix
Experiment Setting (Detail)

• Executed each same experiment setting 10 times
  • We used the average
• Amazon EC2 c4.large (2 vCPUs and 3.75 GiB RAM)