What's to Come is Still Unsure
Synthesizing Controllers Resilient to Delayed Interaction

Mingshuai Chen\textsuperscript{1}, Martin Fränzle\textsuperscript{2}, Yangjia Li\textsuperscript{3,1}, Peter N. Mosaad\textsuperscript{2}, Naijun Zhan\textsuperscript{1}

\texttt{chenms@ios.ac.cn} \hspace{1em} \texttt{lcs.ios.ac.cn/~chenms/}

\textsuperscript{1} State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, China
\textsuperscript{2} Dpt. of Computing Science, Carl v. Ossietzky Universität Oldenburg, Germany
\textsuperscript{3} University of Tartu, Estonia

Los Angeles, October 2018

\textsuperscript{*}. William Shakespeare, Twelfth Night/What You Will, Act 2, Scene 3.
Staying Safe
When Observation & Actuation Suffer from Serious Delays

- You could move slowly. *(Well, can you?)*
- You could trust autonomy.
- Or you have to anticipate and issue actions early.
A Pearl of Wisdom

Indecision and delays are the parents of failure.

(George Canning)
Indecision and delays are the parents of failure.

(George Canning)

- Only relevant to ordinary people's life?
- Or to scientists, in particular comp. sci. and control folks, too?
A Pearl of Wisdom

Indecision and delays are the parents of failure.

(George Canning)

- Only relevant to ordinary people's life?
- Or to scientists, in particular comp. sci. and control folks, too?

Remember that Canning briefly controlled Great Britain!
1. Why Time Delays
2. Safety Games under Delay
3. Synthesizing Controllers Resilient to Delayed Interaction
4. Experimental Evaluation
5. Concluding Remarks
Outline

1. **Why Time Delays**
   - Motivation

2. **Safety Games under Delay**
   - Delayed observation and actuation
   - Reducibility to standard safety games

3. **Synthesizing Controllers Resilient to Delayed Interaction**
   - Incremental handling of order-preserving delays
   - Out-of-order delivery

4. **Experimental Evaluation**
   - Performance

5. **Concluding Remarks**
   - Summary
**Hybrid Systems**

- **Plant Control**
  - **Analog switch**
  - **Continuous controllers**
  - **Discrete supervisor**

- **Disturbances** ("noise")
- **Environmental influence**
- **Control**
- **Selection**
- **Setpoints**
- **Active control law**
- **Part of observable state**
- **Task selection**

** Crucial question:** How do the controller and the plant interact?

**Traditional answer:** Coupling assumed to be (or at least modeled as) delay-free.
Hybrid Systems

Crucial question: How do the controller and the plant interact?

Traditional answer: Coupling assumed to be (or at least modeled as) delay-free.

Mode dynamics is covered by the conjunction of the individual ODEs.

Switching between modes is an immediate reaction to environmental conditions.
Hybrid Systems

Crucial question:
- How do the controller and the plant interact?

Traditional answer:
- Coupling assumed to be (or at least modeled as) delay-free.
  - Mode dynamics is covered by the conjunction of the individual ODEs.
  - Switching btw. modes is an immediate reaction to environmental conditions.
Why Time Delays

Delay Games

Incremental Synthesis

Experimental Evaluation

Concluding Remarks

**Instantaneous Coupling**

Following the tradition, above (rather typical) Simulink model assumes

- delay-free coupling between all components,
- instantaneous feed-through within all functional blocks.

**Central questions:**

1. Is this realistic?
2. If not, does it have observable effect on control performance?
3. May that effect be detrimental or even harmful?
Q1 : Is Instantaneous Coupling Realistic?
Q1: Is Instantaneous Coupling Realistic?

We are no better:
As soon as computer scientists enter the scene, serious delays are ahead...
Q1: Is Instantaneous Coupling Realistic?

Digital control needs **A/D and D/A conversion**, which induces latency in signal forwarding.

Digital signal processing, especially in complex sensors like CV, needs **processing time**, adding signal delays.

**Networked control** introduces communication latency into the feedback control loop.

Harvesting, fusing, and forwarding data through **sensor networks** enlarge the latter by orders of magnitude.
Q1: Is Instantaneous Coupling Realistic? -- No.

Digital control needs A/D and D/A conversion, which induces latency in signal forwarding.

Digital signal processing, especially in complex sensors like CV, requires processing time, adding signal delays.

Networked control introduces communication latency into the feedback control loop.

Harvesting, fusing, and forwarding data through sensor networks enlarge the latter by orders of magnitude.
**Q1a : Resultant Forms of Delay**

**Delayed reaction**: Reaction to a stimulus is not immediate.
- Easy to model in timed automata, hybrid automata, … :

\[
\begin{array}{c}
\text{a / } x:=0 \\
x < 4 \\
x > 3 / b
\end{array}
\]

- Thus amenable to the pertinent analysis tools.

⇒ Not of interest today.
Motivation

Q1a : Resultant Forms of Delay

**Delayed reaction**: Reaction to a stimulus is not immediate.
- Easy to model in timed automata, hybrid automata, …:
  
  ![Transition Diagram]

  a / x := 0 \[\rightarrow\] x < 4 \[\rightarrow\] x > 3 / b

  Thus amenable to the pertinent analysis tools.
  
  ⇒ Not of interest today.

**Network delay**: Information of different age coexists and is queuing in the network when piped towards target.
- End-to-end latency may exceed sampling intervals etc. by orders of magnitude
- Not (continuous-time pipelined delay) or not efficiently (discrete-time pipelined delay) expressible in our std. models.

  ⇒ Our theme today: discrete-time pipelined delay.

[M. Chen, M. Fränzle *et al.*. ATVA’18],
[M. Zimmermann. LICS’18, GandALF’17], [F. Klein & M. Zimmermann. ICALP’15, CSL’15].
Q2 : Do Delays Have Observable Effect?

Figure: A robot escape game in a $4 \times 4$ room, with

\[
\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\},
\]
\[
\Sigma_k = \{R, L, U, D\}.
\]
Q2: Do Delays Have Observable Effect?

Figure: A robot escape game in a $4 \times 4$ room, with
\[
\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\},
\Sigma_k = \{R, L, U, D\}.
\]
Q2 : Do Delays Have Observable Effect?

Figure : A robot escape game in a $4 \times 4$ room, with

$\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\},$

$\Sigma_k = \{R, L, U, D\}.$
Q2: Do Delays Have Observable Effect?

Figure: A robot escape game in a 4x4 room, with
\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\} \]
\[ \Sigma_k = \{R, L, U, D\} \]
Q2: Do Delays Have Observable Effect?

Figure: A robot escape game in a $4 \times 4$ room, with

\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]
\[ \Sigma_k = \{R, L, U, D\}. \]

No delay:
Robot always wins by circling around the obstacle at (1,2).

Motivation
Q2: Do Delays Have Observable Effect?
Q2: Do Delays Have Observable Effect?

**Figure**: A robot escape game in a 4×4 room, with

\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]

\[ \Sigma_k = \{R, L, U, D\}. \]

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Q2: Do Delays Have Observable Effect?

Figure: A robot escape game in a 4x4 room, with
\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]
\[ \Sigma_k = \{R, L, U, D\}. \]

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.
Q2: Do Delays Have Observable Effect?

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:

Figure: A robot escape game in a 4 x 4 room, with
\[ \Sigma_r = \{ RU, UR, LU, UL, RD, DR, LD, DL, \epsilon \}, \]
\[ \Sigma_k = \{ R, L, U, D \}. \]
Q2: Do Delays Have Observable Effect?

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:
Robot still wins, yet extra memory is needed.

Figure: A robot escape game in a 4×4 room, with
\[ \Sigma_r = \{ \text{RU, UR, LU, UL, RD, DR, LD, DL, } \epsilon \} \],
\[ \Sigma_k = \{ \text{R, L, U, D} \} \].
Q2 : Do Delays Have Observable Effect?

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:
Robot still wins, yet extra memory is needed.

3 steps delay:

Figure: A robot escape game in a 4 x 4 room, with
\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]
\[ \Sigma_k = \{R, L, U, D\}. \]
Q2 : Do Delays Have Observable Effect?

Figure: A robot escape game in a $4 \times 4$ room, with

$\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon \}$,
$\Sigma_k = \{R, L, U, D \}$.

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:
Robot still wins, yet extra memory is needed.

3 steps delay:
Robot is unwinnable (uncontrollable) anymore.
Q2: Do Delays Have Observable Effect?  -- Yes, they have.

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:
Robot still wins, yet extra memory is needed.

3 steps delay:
Robot is unwinnable (uncontrollable) anymore.

Figure: A robot escape game in a $4 \times 4$ room, with
\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]
\[ \Sigma_k = \{R, L, U, D\}. \]
Q3: May the Effects be Harmful? -- Yes, delays may well annihilate control performance.

No delay:
Robot always wins by circling around the obstacle at (1,2).

1 step delay:
Robot wins by 1-step pre-decision.

2 steps delay:
Robot still wins, yet extra memory is needed.

3 steps delay:
Robot is unwinnable (uncontrollable) anymore.

Figure: A robot escape game in a 4 × 4 room, with
\[
\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \\
\Sigma_k = \{R, L, U, D\}.
\]
Outline

1. Why Time Delays
   - Motivation

2. Safety Games under Delay
   - Delayed observation and actuation
   - Reducibility to standard safety games

3. Synthesizing Controllers Resilient to Delayed Interaction
   - Incremental handling of order-preserving delays
   - Out-of-order delivery

4. Experimental Evaluation
   - Performance

5. Concluding Remarks
   - Summary
A Trivial Safety Game

Goal: Avoid $a_5$ by appropriate actions of player $e$.
A Trivial Safety Game

**Goal:** Avoid $a_5$ by appropriate actions of player $e$.

**Strategy:** May always play "$a$" except in $e_3$:

- $e_1, e_2 \leftrightarrow a$
- $e_3 \leftrightarrow b$
Observation: It doesn't make an observable difference for the joint dynamics whether delay occurs in perception, actuation, or both.
Playing Safety Game Subject to Discrete Delay

**Observation**: It doesn't make an observable difference for the joint dynamics whether delay occurs in perception, actuation, or both.

**Consequence**: There is an obvious reduction to a safety game of perfect information.

---

1. **In fact, two different ones**: To mimic opacity of the shift registers, delay has to be moved to actuation/sensing for ego/adversary, resp. *The two thus play different games!*
Reduction to Delay-Free Games
from Ego-Player Perspective

Safety games w. delay can be solved algorithmically.

Game graph incurs blow-up by factor $|\text{Alphabet(ego)}|^{\text{delay}}$. 
Safety games w. delay can be solved algorithmically.

Game graph incurs blow-up by factor $|\text{Alphabet(ego)}|^\text{delay}$. 
The Simple Safety Game
...but with Delay

No delay:
\( e_1, e_2 \mapsto a \)
\( e_3 \mapsto b \)

1 step delay: Strategy?
\( a_1, a_4 \mapsto a \)
\( a_2, a_3 \mapsto b \)
The Simple Safety Game

...but with Delay

No delay:

- $e_1, e_2 \mapsto a$
- $e_3 \mapsto b$

1 step delay: Strategy?

- $a_1, a_4 \mapsto a$
- $a_2, a_3 \mapsto b$

2 steps delay: Strategy?

- $e_1 \mapsto \begin{cases} a & \text{if 2 steps back an "a" was issued,} \\ b & \text{if 2 steps back a "b" was issued.} \end{cases}$
- $e_2 \mapsto b$
- $e_3 \mapsto a$

Need memory!
Outline

1. Why Time Delays
   - Motivation

2. Safety Games under Delay
   - Delayed observation and actuation
   - Reducibility to standard safety games

3. Synthesizing Controllers Resilient to Delayed Interaction
   - Incremental handling of order-preserving delays
   - Out-of-order delivery

4. Experimental Evaluation
   - Performance

5. Concluding Remarks
   - Summary
**Observation**: A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay $k$.

**Consequence**: That a position is winning for delay $k$ is a necessary condition for it being winning under delay $k' > k$. 
Observation: A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay $k$.

Consequence: That a position is winning for delay $k$ is a necessary condition for it being winning under delay $k' > k$.

Idea: Incrementally filter out loss states & incrementally synthesize winning strategy for the remaining:

1. Synthesize winning strategy for underlying delay-free safety game.
2. For each winning state, lift strategy from delay $k$ to $k + 1$.
3. Remove states where this does not succeed.
4. Repeat from 2 until either delay-resilience suffices (winning) or initial state turns lossy (losing).

Incremental Synthesis of Delay-Tolerant Strategies

1. Generate a *maximally permissive* strategy for delay $k = 0$. 
Incremental Synthesis of Delay-Tolerant Strategies

1. Generate a maximally permissive strategy for delay $k = 0$.

2. Advance to delay $k + 1$:
   - If $k$ odd: For each (ego-)winning adversarial state define strategy as
     
     ```latex
     \text{after playing } \sigma_1, \ldots, \sigma_{(k-1)/2},
     \text{play } \{a, c, e\}
     ```

     - and eliminate any dead ends by bwd. traversal.

   - If $k$ even: For each winning ego state define strategy as
     
     ```latex
     \text{after playing } \sigma_1, \ldots, \sigma_{(k-1)/2},
     \text{play } \{a, c, e\} \cap \{b, c, e, f\} = \{c, e\}
     ```

     - and eliminate any dead ends by bwd. traversal.
**Incremental Synthesis of Delay-Tolerant Strategies**

1. Generate a *maximally permissive* strategy for delay $k = 0$.

2. Advance to delay $k + 1$:

   - **If $k$ odd**: For each (ego-)winning adversarial state define strategy as

   - After playing $\sigma_1, \ldots, \sigma_{(k-1)/2}$, play $\{a, c, e\}$

   - After playing $\sigma_1, \ldots, \sigma_{(k-1)/2}$, play $\{b, c, e, f\}$

   ... and eliminate any dead ends by bwd. traversal.

   - **If $k$ even**: For each winning ego state define strategy as

   - Play $\sigma'_1, \ldots, \sigma'_{k/2}$

   - Play $a, \sigma_1, \ldots, \sigma_{k/2}$ or play $c, \sigma'_1, \ldots, \sigma'_{k/2}$

---

Mingshuai Chen  Institute of Software, CAS  Synthesizing Controllers Resilient to Delay  Los Angeles, ATVA 2018  20 / 25
**Incremental Synthesis of Delay-Tolerant Strategies**

1. Generate a *maximally permissive* strategy for delay $k = 0$.

2. Advance to delay $k + 1$:
   - **If $k$ odd:** For each (ego-)winning adversarial state define strategy as
     - after playing $\sigma_1, \ldots, \sigma_{(k-1)/2}$, play $\{a, c, e\}$
     - after playing $\sigma_1, \ldots, \sigma_{(k-1)/2}$, play $\{b, c, e, f\}$

     ![Diagram](image)

3. Repeat from 2 until either delay-resilience suffices or initial state turns lossy.
Observations may arrive out-of-order:

- Maximum delay 5
- Out of order!

Sample #

<table>
<thead>
<tr>
<th>t</th>
<th>59</th>
<th>60</th>
<th>61</th>
<th>62</th>
<th>63</th>
<th>64</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>57</td>
<td>58</td>
<td>61</td>
<td>60</td>
<td>59</td>
<td>62</td>
<td></td>
</tr>
</tbody>
</table>
How About Non-Order-Preserving Delays?

- Observations may arrive out-of-order:

  ![Diagram](image-url)

  - Maximum delay 5
  - Out of order!

- But this may only reduce effective delay, improving controllability:

  ![Diagram](image-url)

  - Maximum delay 5
  - Factual delay 3
  - Effective delay 2
  - More recent state information available earlier

Regarding qualitative controllability, the worst-case of out-of-order delivery is equivalent to order-preserving delay $k$. Stochastically expected controllability even better than for strict delay $k$. 

Mingshuai Chen
Institute of Software, CAS

Synthesizing Controllers Resilient to Delay

Los Angeles, ATVA 2018 21 / 25
How About Non-Order-Preserving Delays?

- Observations may arrive out-of-order:

- But this may only reduce effective delay, improving controllability:

- W.r.t. qualitative controllability, the worst-case of out-of-order delivery is equivalent to order-preserving delay $k$.

- Stochastically expected controllability even better than for strict delay $k$. 
Outline

1. Why Time Delays
   - Motivation

2. Safety Games under Delay
   - Delayed observation and actuation
   - Reducibility to standard safety games

3. Synthesizing Controllers Resilient to Delayed Interaction
   - Incremental handling of order-preserving delays
   - Out-of-order delivery

4. Experimental Evaluation
   - Performance

5. Concluding Remarks
   - Summary
### Incremental vs. Reduction-Based

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Reduction + Explicit-State Synthesis</th>
<th>Incremental Explicit-State Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\delta_{\text{max}}$ $\delta = 0$ $\delta = 1$ $\delta = 2$ $\delta = 3$ $\delta = 4$</td>
<td>$\delta_{\text{max}}$ $\delta = 0$ $\delta = 1$ $\delta = 2$ $\delta = 3$ $\delta = 4$ $%$</td>
</tr>
<tr>
<td>Exmp.trv1</td>
<td>≥ 20 0.00 0.01 0.02 0.02</td>
<td>≥ 30 0.00 0.00 0.01 0.01</td>
</tr>
<tr>
<td>Exmp.trv2</td>
<td>= 2 0.00 0.01 0.01 0.02</td>
<td>= 2 0.08 0.13 0.22 0.25</td>
</tr>
<tr>
<td>Escp.4×4</td>
<td>≥ 20 0.08 11.66 11.73 1059.23</td>
<td>≥ 2 0.08 0.13 0.22 0.25</td>
</tr>
<tr>
<td>Escp.4×5</td>
<td>= 2 0.18 34.09 33.80 3084.58</td>
<td>= 2 0.18 0.27 0.46 0.63</td>
</tr>
<tr>
<td>Escp.5×5</td>
<td>≥ 2 0.46 96.24 97.10</td>
<td>= 2 0.46 0.68 1.16 1.71</td>
</tr>
<tr>
<td>Escp.5×6</td>
<td>≥ 2 1.01 217.63 216.83</td>
<td>= 2 1.00 1.42 2.40 4.30</td>
</tr>
<tr>
<td>Escp.6×6</td>
<td>≥ 2 2.13 516.92 511.41</td>
<td>= 2 2.06 2.90 5.12 10.30</td>
</tr>
<tr>
<td>Escp.7×7</td>
<td>≥ 2 7.81 2167.86 2183.01</td>
<td>= 2 7.71 10.67 19.04 52.47</td>
</tr>
<tr>
<td>Escp.7×8</td>
<td>≥ 0 13.07</td>
<td>= 2 13.44 18.25 32.69 108.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Reduction + Yosys + SafetySynth (symbolic)</th>
<th>Incremental Synthesis (explicit-state implementation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\delta_{\text{max}}$ $\delta = 0$ $\delta = 1$ $\delta = 2$ $\delta = 3$ $\delta = 4$</td>
<td>$\delta = 0$ $\delta = 1$ $\delta = 2$ $\delta = 3$ $\delta = 4$ $\delta = 5$ $\delta = 6$ $%$</td>
</tr>
<tr>
<td>Stub.4×4</td>
<td>= 2 1.07 1.24 1.24 1.80</td>
<td>0.04 0.07 0.12 0.18</td>
</tr>
<tr>
<td>Stub.4×5</td>
<td>= 2 1.16 1.49 1.49 2.83</td>
<td>0.08 0.14 0.25 0.44</td>
</tr>
<tr>
<td>Stub.5×5</td>
<td>= 2 1.19 2.61 2.50 13.67</td>
<td>0.21 0.37 0.63 1.17</td>
</tr>
<tr>
<td>Stub.5×6</td>
<td>= 2 1.18 2.60 2.59 23.30</td>
<td>0.42 0.69 1.20 2.49</td>
</tr>
<tr>
<td>Stub.6×6</td>
<td>= 4 1.17 2.76 2.74 19.96 19.69 655.24</td>
<td>0.93 1.47 2.60 5.79 7.54 7.60</td>
</tr>
<tr>
<td>Stub.7×7</td>
<td>= 4 1.23 2.50 2.48 24.57 23.01 2224.62</td>
<td>3.60 5.52 10.08 22.75 31.18 32.98</td>
</tr>
</tbody>
</table>

**Table**: Benchmark results in relation to reduction-based approaches (time in seconds)
## Incremental vs. Reduction-Based

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Reduction + Explicit-State Synthesis</th>
<th>Incremental Explicit-State Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td>**</td>
<td>S</td>
</tr>
<tr>
<td>Exmp.trv1</td>
<td>≥ 22 0.00 0.00 0.01 0.02 0.02</td>
<td>≥ 30 0.00 0.00 0.00 0.01 0.01</td>
</tr>
<tr>
<td>Exmp.trv2</td>
<td>2 0.00 0.01 0.01 0.02 –</td>
<td>2 0.00 0.00 0.00 0.01 –</td>
</tr>
<tr>
<td>Escp.4×4</td>
<td>2 0.08 11.66 11.73 1059.23 –</td>
<td>2 0.08 0.13 0.22 0.25 –</td>
</tr>
<tr>
<td>Escp.4×5</td>
<td>2 0.18 34.09 33.80 3084.58 –</td>
<td>2 0.18 0.27 0.46 0.63 –</td>
</tr>
<tr>
<td>Escp.5×5</td>
<td>≥ 2 0.46 96.24 97.10 ? ?</td>
<td>= 2 0.46 0.68 1.16 1.71 –</td>
</tr>
<tr>
<td>Escp.5×6</td>
<td>≥ 2 1.01 217.63 216.83 ? ?</td>
<td>≥ 2 1.00 1.42 2.40 4.30 –</td>
</tr>
<tr>
<td>Escp.6×6</td>
<td>≥ 2 2.13 516.92 511.41 ? ?</td>
<td>≥ 2 2.06 2.90 5.12 10.30 –</td>
</tr>
<tr>
<td>Escp.7×7</td>
<td>≥ 2 7.81 2167.86 2183.01 ? ?</td>
<td>≥ 2 7.71 10.67 19.04 52.47 –</td>
</tr>
</tbody>
</table>

### Concluding Remarks

Relating our symbolic model of the escape games to reduction-based approaches, we observe that incremental explicit-state synthesis outperforms reduction-based approaches. For some larger room sizes and small delays, the robot under delays is controllable, whereas for larger delays, the controller has no winning strategy. Our incremental algorithm always wins within the given CPU time bound, except for some larger delays.

**Remark 1.** The performance of the current explicit-state implementation of Algorithm 1 is compared with that of SafetySynth, the winner in the safety synthesis track of the 3rd and 4th Reactive Synthesis Competition. As these are conveyed in AIGER format only and its input is an extension of the AIGER format known from hardware model-checking, use of non-symbolic encodings is desirable. For larger delays, we used a slight modification of the escape game forbidding the kid to take moves to the right or up, increasing the controllability for the robot.

**Remark 2.** It is evident that our incremental algorithm always wins, despite its sequential safety synthesis track of the 3rd and 4th Reactive Synthesis Competition. As these are conveyed in AIGER format only and its input is an extension of the AIGER format known from hardware model-checking, use of non-symbolic encodings is desirable. For larger delays, we used a slight modification of the escape game forbidding the kid to take moves to the right or up, increasing the controllability for the robot.

### Table: Benchmark results in relation to reduction-based approaches (time in seconds)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Reduction + Yosys + SafetySynth (symbolic)</th>
<th>Incremental Synthesis (explicit-state implementation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stub.4×4</td>
<td>δ = 0 1.07 1.24 1.24 1.80 – – –</td>
<td>δ = 0 0.04 0.07 0.12 0.18 – – –</td>
</tr>
<tr>
<td>Stub.4×5</td>
<td>δ = 0 1.16 1.49 1.49 2.83 – – –</td>
<td>δ = 0 0.08 0.14 0.25 0.44 – – –</td>
</tr>
<tr>
<td>Stub.5×5</td>
<td>δ = 0 1.19 2.61 2.50 13.67 – – –</td>
<td>δ = 0 0.21 0.37 0.63 1.17 – – –</td>
</tr>
<tr>
<td>Stub.5×6</td>
<td>δ = 0 1.18 2.60 2.59 23.30 – – –</td>
<td>δ = 0 0.42 0.69 1.20 2.49 – – –</td>
</tr>
<tr>
<td>Stub.6×6</td>
<td>δ = 0 1.17 2.76 2.74 19.96 19.69 655.24 –</td>
<td>δ = 0 0.93 1.47 2.60 5.79 7.54 7.60 –</td>
</tr>
<tr>
<td>Stub.7×7</td>
<td>δ = 0 1.23 2.50 2.48 24.57 23.01 2224.62 –</td>
<td>δ = 0 3.60 5.52 10.08 22.75 31.18 32.98 –</td>
</tr>
</tbody>
</table>

**Table:** Benchmark results in relation to reduction-based approaches (time in seconds)
Outline

1. Why Time Delays
   - Motivation

2. Safety Games under Delay
   - Delayed observation and actuation
   - Reducibility to standard safety games

3. Synthesizing Controllers Resilient to Delayed Interaction
   - Incremental handling of order-preserving delays
   - Out-of-order delivery

4. Experimental Evaluation
   - Performance

5. Concluding Remarks
   - Summary
Concluding Remarks

Problem: We face
- increasingly wide-spread use of networked distributed sensing and control,
- substantial delays thus impacting controllability and control performance,
- naïve reduction to delay-free settings, yet with an exponential blow-up.

Status: We present
- insufficiency of memoryless control strategies for discrete safety games under delay,
- incremental algorithm for efficient delay-tolerant control synthesis,
- the practically relevant case of non-order-preserving delays.

Future Work: We plan to
- integrate stochastic models of message delays into safety synthesis processes,
- let synthesis constructively leverage the advantages of (partial) control on out-of-order delivery,
- extend to hybrid setting combining delayed continuous and delayed discrete reactive behavior.