What's to Come is Still Unsure *

Synthesizing Controllers Resilient to Delayed Interaction

Mingshuai Chen[1], Martin Fränzle[2], Yangjia Li[3,1], Peter N. Mosaad[2], Naijun Zhan[1]

chenms@ios.ac.cn lcs.ios.ac.cn/~chenms/

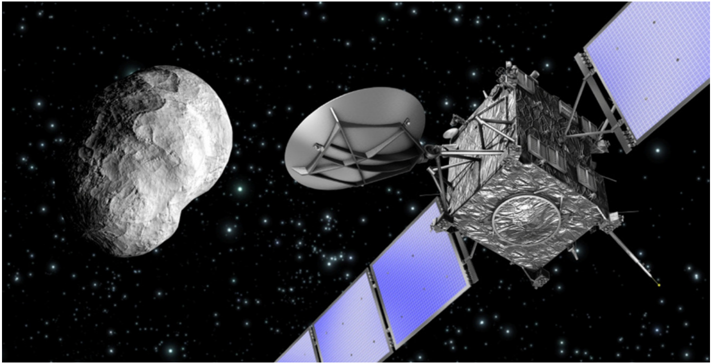
[1] State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[2] Dpt. of Computing Science, Carl v. Ossietzky Universität Oldenburg, Germany
[3] University of Tartu, Estonia

Los Angeles, October 2018

*. William Shakespeare, Twelfth Night/What You Will, Act 2, Scene 3.

Staying Safe

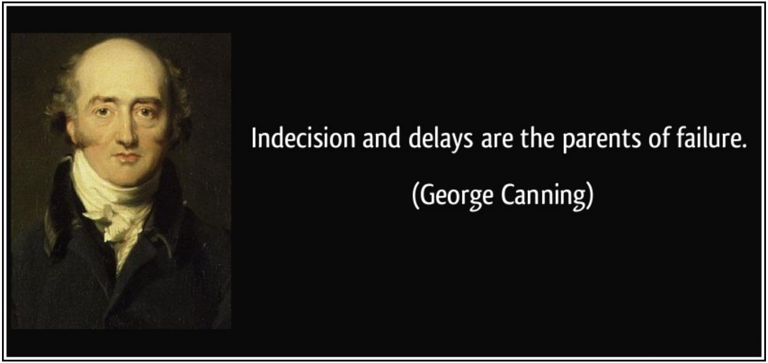
When Observation & Actuation Suffer from Serious Delays



©ESA

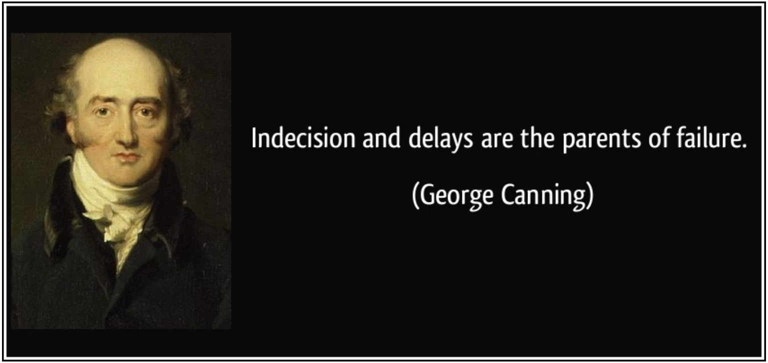
- You could move slowly. (Well, can you ?)
- You could trust autonomy.
- Or you have to anticipate and issue actions early.

A Pearl of Wisdom



©izQuotes

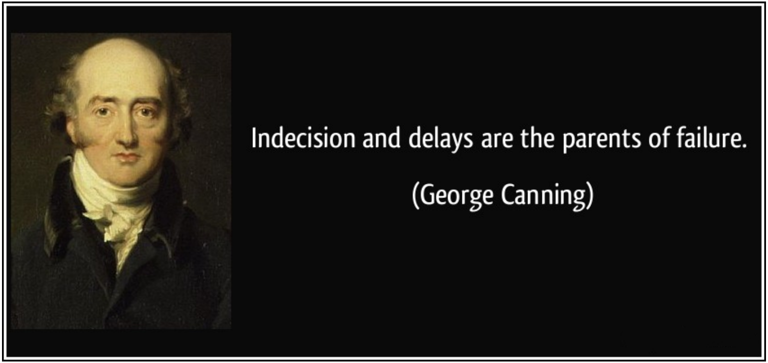
A Pearl of Wisdom



©izQuotes

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

A Pearl of Wisdom



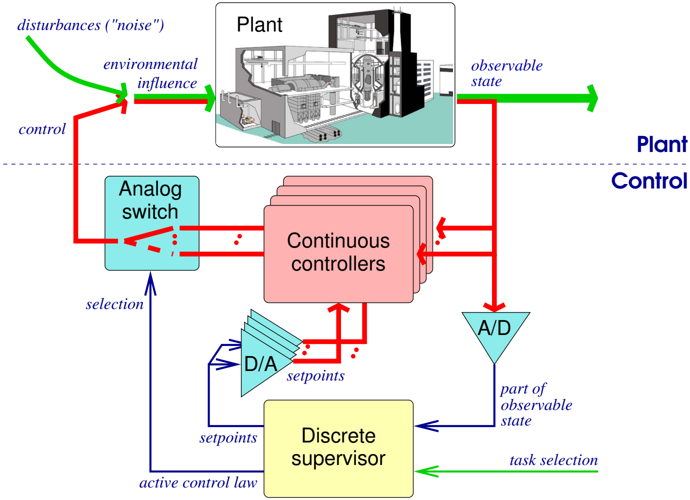
- Only relevant to ordinary people's life ?
- Or to scientists, in particular comp. sci. and control folks, too ?

Remember that Canning briefly controlled Great Britain !

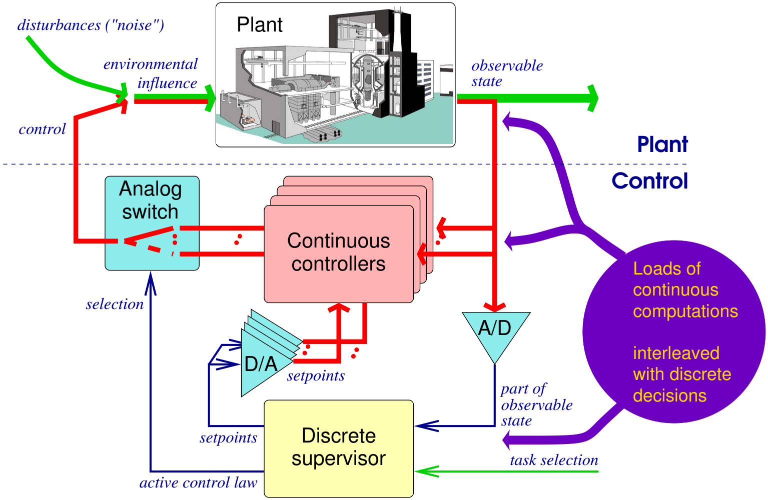
Outline

Outline

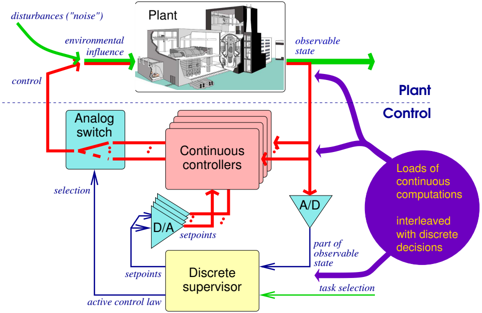
Hybrid Systems



Hybrid Systems



Hybrid Systems



Crucial question :

- How do the controller and the plant interact ?

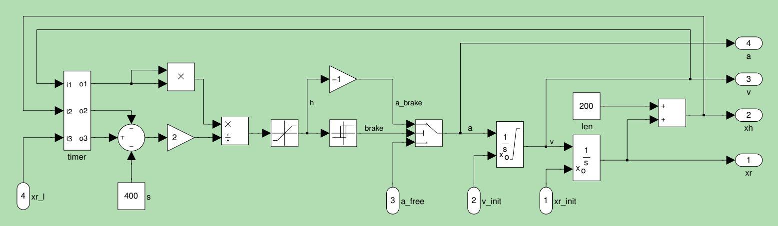
Traditional answer :

- 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.

Instantaneous Coupling



©ETCS-3

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

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

Central questions :

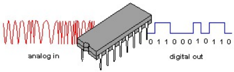
1. Is this realistic ?
2. If not, does it have observable effect on control performance ?
3. May that effect be detrimental or even harmful ?

Q1 : Is Instantaneous Coupling Realistic ?

Q1 : Is Instantaneous Coupling Realistic ?

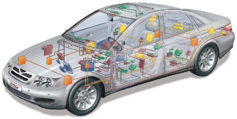
We are no better :
As soon as computer scientists enter the scene, serious delays are ahead…

Q1 : Is Instantaneous Coupling Realistic ?



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

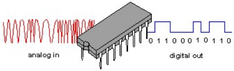
Digital signal processing, especially in complex sensors like CV, needs processing time, adding signal delays.



Networked control introduces communication latency into the feedback control loop.

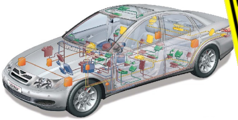
Harvesting, fusing, and forwarding data through sensor networks enlarge the latter by orders of magnitude.

Q1 : Is Instantaneous Coupling Realistic ? -- No.



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

Digital signal processin… specially in complex sensors like CV… g time, adding signal de-



… s communication la-… 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, … :

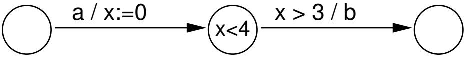
- Thus amenable to the pertinent analysis tools.

⇒ Not of interest today.

Q1a : Resultant Forms of Delay

Delayed reaction : Reaction to a stimulus is not immediate.

- Easy to model in timed automata, hybrid automata, … :



- 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 ?

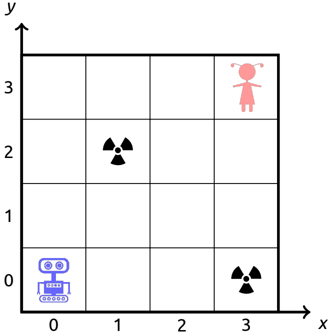


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

Q2 : Do Delays Have Observable Effect ?

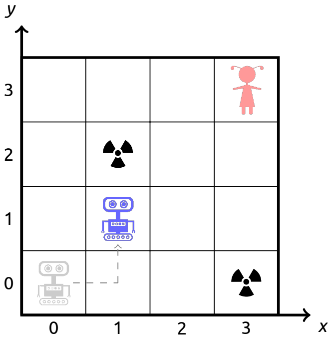


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

Q2 : Do Delays Have Observable Effect ?

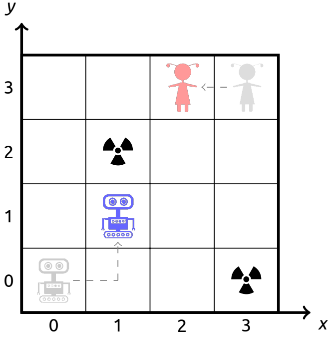


Figure : A robot escape game in a 4 × 4 room, with
$\Sigma_r = \{$RU, UR, LU, UL, RD, DR, LD, DL, $\epsilon\}$,
$\Sigma_k = \{$R, L, U, D$\}$.

Q2 : Do Delays Have Observable Effect ?

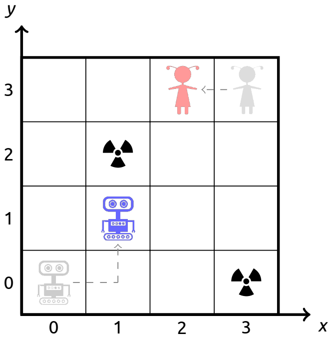


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

No delay :

Q2 : Do Delays Have Observable Effect ?

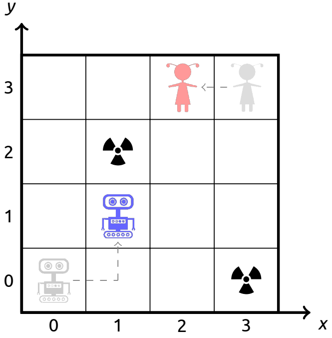


Figure : A robot escape game in a 4 $\times$ 4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

No delay :

Robot always wins by circling around the obstacle at (1,2).

Q2 : Do Delays Have Observable Effect ?

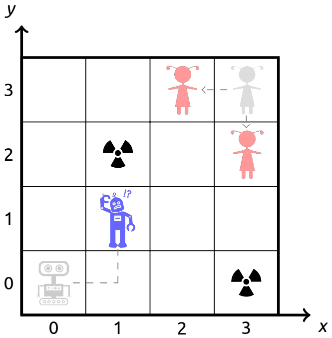


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

No delay :

Robot always wins by circling around the obstacle at (1,2).

1 step delay :

Q2 : Do Delays Have Observable Effect ?

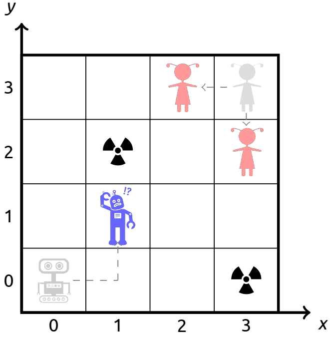


Figure : A robot escape game in a 4 × 4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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 ?

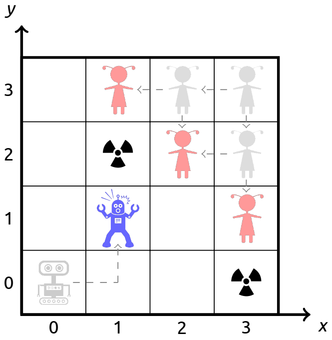


Figure : A robot escape game in a 4×4 room, with $\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$, $\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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 :

Q2 : Do Delays Have Observable Effect ?

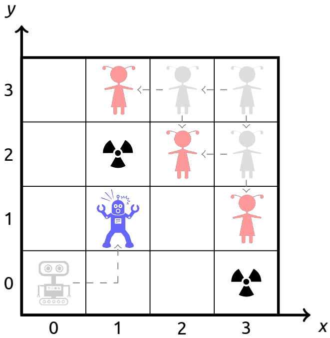


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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.

Q2 : Do Delays Have Observable Effect ?

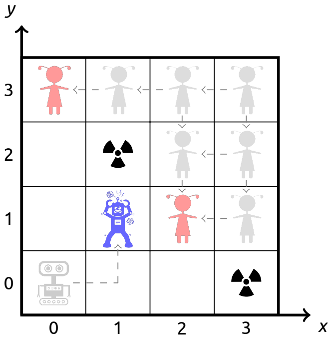


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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 :

Q2 : Do Delays Have Observable Effect ?

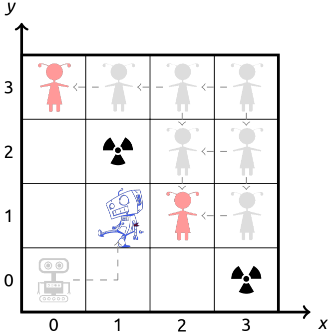


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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.

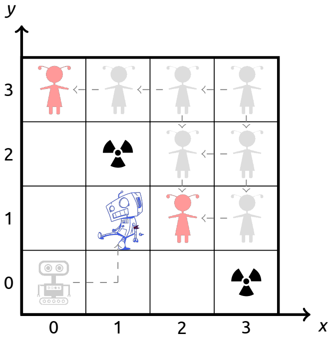


Figure : A robot escape game in a 4×4 room, with $\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$, $\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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.

Q3 : May the Effects be Harmful ? -- Yes, delays may well annihilate control performance.

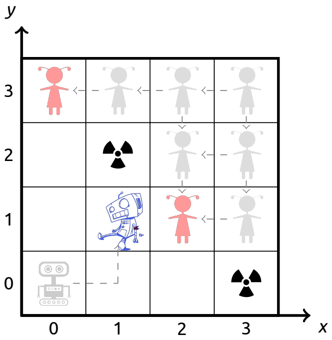


Figure : A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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

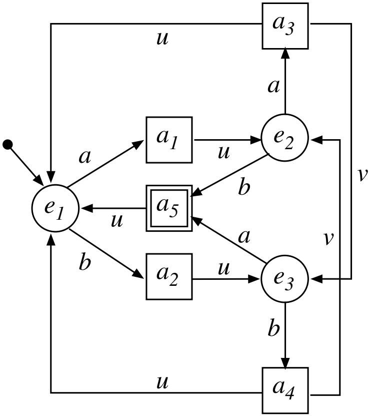
1 step delay :
Robot wins by 1-step pre-decision.

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

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

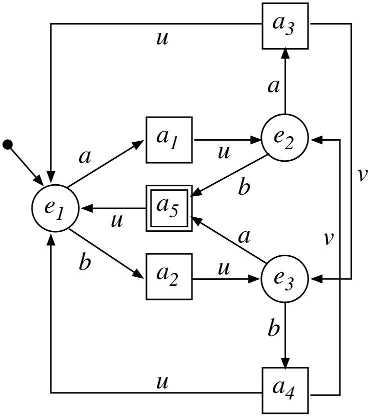
Outline

A Trivial Safety Game



Goal : Avoid $\boxed{a_5}$ by appropriate actions of player *e*.

A Trivial Safety Game

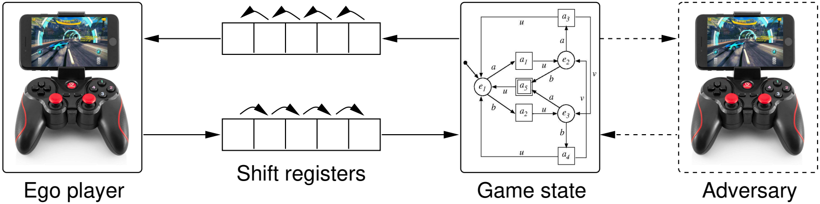


Goal : Avoid $\boxed{a_5}$ by appropriate actions of player e.

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

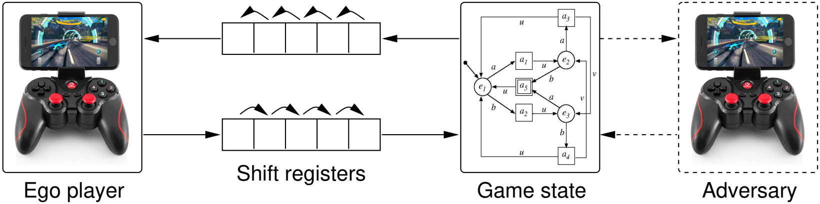
$$\begin{aligned} e_1, e_2 &\mapsto a \\ e_3 &\mapsto b \end{aligned}$$

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.

Playing Safety Game Subject to Discrete Delay



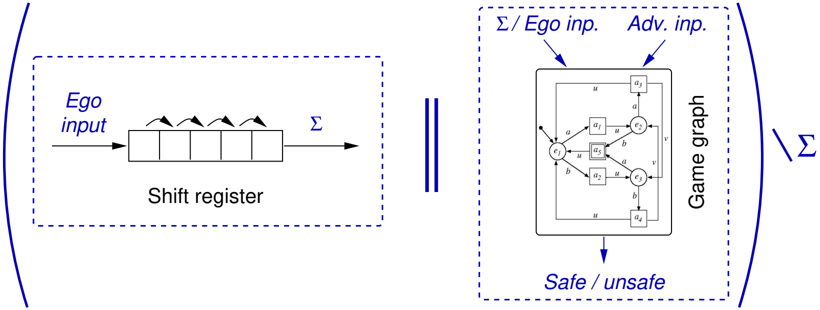
Observation : It doesn't make an observable difference for the joint dynamics whether delay occurs in perception, actuation, or both.

Consequence : There is an[1] 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 !*

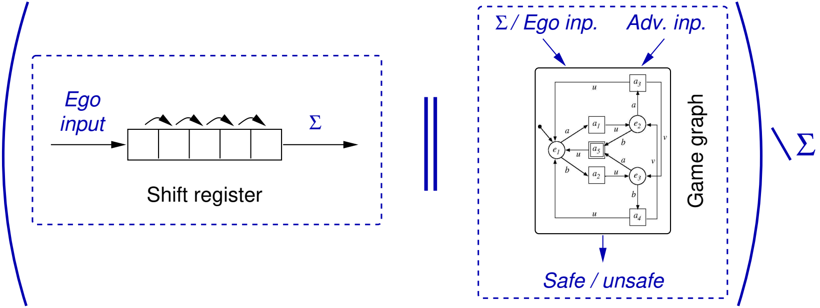
Reduction to Delay-Free Games

from Ego-Player Perspective



Reduction to Delay-Free Games

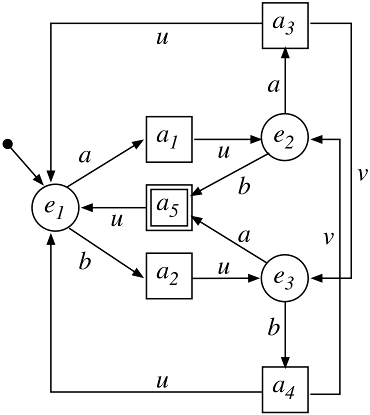
from Ego-Player Perspective



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

The Simple Safety Game

...but with Delay



No delay :

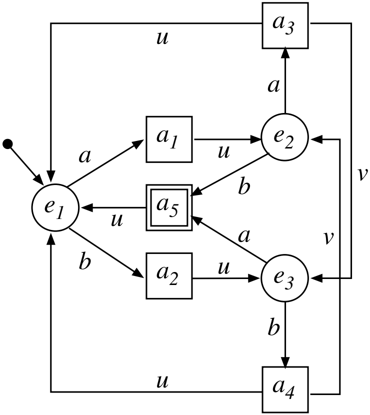
$e_1, e_2 \mapsto a$
$e_3 \mapsto b$

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 ?

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

Need memory !

Outline

Incremental Synthesis in a Nutshell

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 $k' > k$.

Incremental Synthesis in a Nutshell

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).

- M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction*. To appear in Proc. of ATVA 2018.

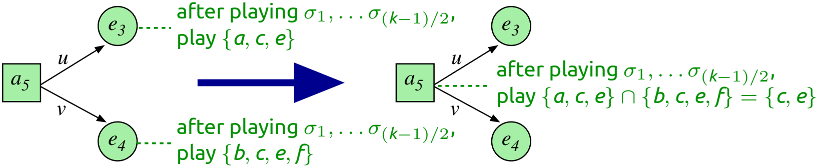
Incremental Synthesis of Delay-Tolerant Strategies

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

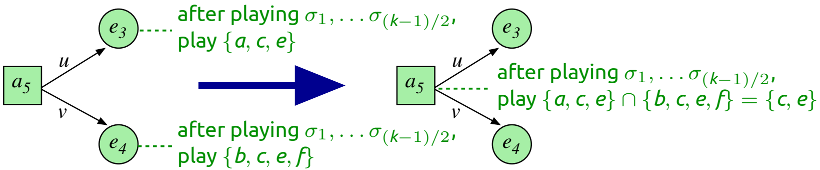


… and eliminate any dead ends by bwd. traversal.

Incremental Synthesis of Delay-Tolerant Strategies

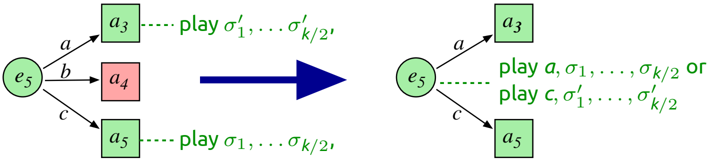
1. Generate a *maximally permissive* strategy for delay $k = 0$.
2. Advance to delay $k + 1$:

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



… and eliminate any dead ends by bwd. traversal.

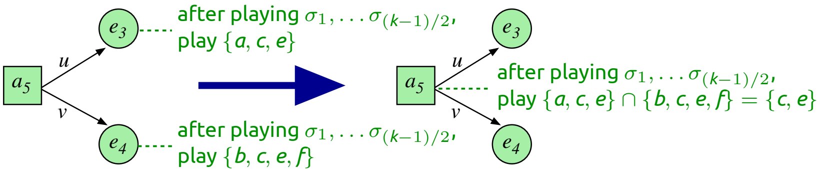
If k even: For each winning ego state define strategy as



Incremental Synthesis of Delay-Tolerant Strategies

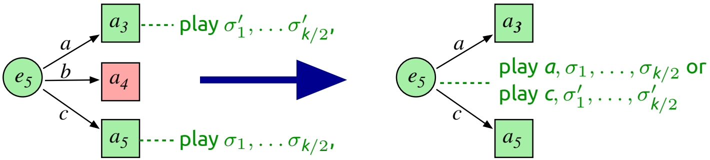
1. Generate a *maximally permissive* strategy for delay $k = 0$.
2. Advance to delay $k + 1$:

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



   … and eliminate any dead ends by bwd. traversal.

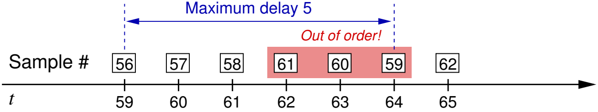
   If k even: For each winning ego state define strategy as



3. Repeat from 2 until either delay-resilience suffices or initial state turns lossy.

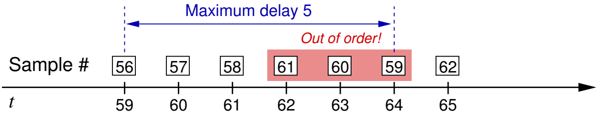
How About Non-Order-Preserving Delays ?

☹ Observations may arrive out-of-order :

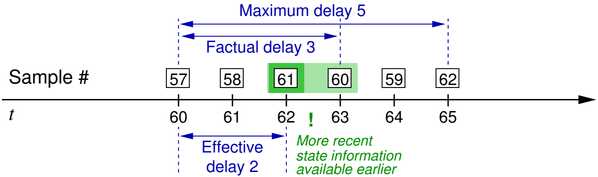


How About Non-Order-Preserving Delays ?

☹ Observations may arrive out-of-order :

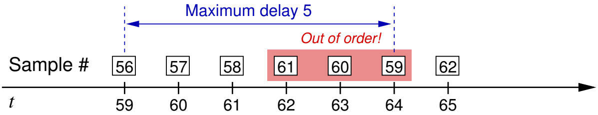


☺ But this may only reduce effective delay, improving controllability :

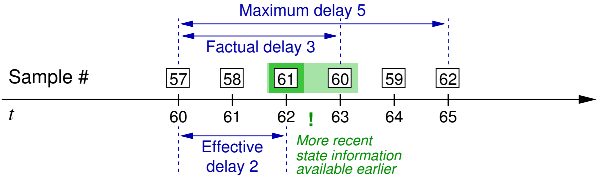


How About Non-Order-Preserving Delays ?

☹ 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

Incremental vs. Reduction-Based

Benchmark				Reduction + Explicit-State Synthesis						Incremental Explicit-State Synthesis						
name	$\lvert S\rvert$	$\lvert\rightarrow\rvert$	$\lvert\mathcal{U}\rvert$	$\delta_{\max}$	$\delta=0$	$\delta=1$	$\delta=2$	$\delta=3$	$\delta=4$	$\delta_{\max}$	$\delta=0$	$\delta=1$	$\delta=2$	$\delta=3$	$\delta=4$	%
Exmp.trv1	14	20	4	≥ 22	0.00	0.00	0.01	0.02	0.02	$\boldsymbol{\geq 30}$	0.00	0.00	**0.00**	**0.01**	**0.01**	
Exmp.trv2	14	22	4	$=2$	0.00	0.01	0.01	0.02	–	$=2$	0.00	**0.00**	**0.00**	**0.01**	–	81.97
Escp.4×4	224	738	16	$=2$	0.08	11.66	11.73	1059.23	–	$=2$	0.08	**0.13**	**0.22**	**0.25**	–	99.02
Escp.4×5	360	1326	20	$=2$	0.18	34.09	33.80	3084.58	–	$=2$	0.18	**0.27**	**0.46**	**0.63**	–	99.02
Escp.5×5	598	2301	26	≥ 2	0.46	96.24	97.10	?	?	$\boldsymbol{=2}$	0.46	**0.68**	**1.16**	**1.71**	–	98.98
Escp.5×6	840	3516	30	≥ 2	1.01	217.63	216.83	?	?	$\boldsymbol{=2}$	**1.00**	**1.42**	**2.40**	**4.30**	–	99.00
Escp.6×6	1224	5424	36	≥ 2	2.13	516.92	511.41	?	?	$\boldsymbol{=2}$	**2.06**	**2.90**	**5.12**	**10.30**	–	98.97
Escp.7×7	2350	11097	50	≥ 2	7.81	2167.86	2183.01	?	?	$\boldsymbol{=2}$	**7.71**	**10.67**	**19.04**	**52.47**	–	98.99
Escp.7×8	3024	14820	56	≥ 0	**13.07**	?	?	?	?	$\boldsymbol{=2}$	13.44	**18.25**	**32.69**	**108.60**	–	99.01

Benchmark		Reduction + Yosys + SafetySynth (symbolic)							Incremental Synthesis (explicit-state implementation)							
name	$\delta_{\max}$	$\delta=0$	$\delta=1$	$\delta=2$	$\delta=3$	$\delta=4$	$\delta=5$	$\delta=6$	$\delta=0$	$\delta=1$	$\delta=2$	$\delta=3$	$\delta=4$	$\delta=5$	$\delta=6$	%
Stub.4×4	$=2$	1.07	1.24	1.24	1.80	–	–	–	**0.04**	**0.07**	**0.12**	**0.18**	–	–	–	98.98
Stub.4×5	$=2$	1.16	1.49	1.49	2.83	–	–	–	**0.08**	**0.14**	**0.25**	**0.44**	–	–	–	98.97
Stub.5×5	$=2$	1.19	2.61	2.50	13.67	–	–	–	**0.21**	**0.37**	**0.63**	**1.17**	–	–	–	98.97
Stub.5×6	$=2$	1.18	2.60	2.59	23.30	–	–	–	**0.42**	**0.69**	**1.20**	**2.49**	–	–	–	98.96
Stub.6×6	$=4$	1.17	2.76	2.74	19.96	19.69	655.24	–	**0.93**	**1.47**	**2.60**	**5.79**	**7.54**	**7.60**	–	99.89
Stub.7×7	$=4$	**1.23**	**2.50**	**2.48**	24.57	**23.01**	2224.62	–	3.60	5.52	10.08	**22.75**	31.18	**32.98**	–	99.88

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

Incremental vs. Reduction-Based

Benchmark				Reduction + Explicit-State Synthesis						Incremental Explicit-State Synthesis						
name	$\lvert S \rvert$	$\lvert \rightarrow \rvert$	$\lvert \mathcal{U} \rvert$	$\delta_{\max}$	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta_{\max}$	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	%
`Exmp.trv1`	14	20	4	≥ 22	0.00	0.00	0.01	0.02	0.02	$\boldsymbol{\geq 30}$	0.00	0.00	**0.00**	**0.01**	**0.01**	
`Exmp.trv2`	14	22	4	$= 2$	0.00	0.01	0.01	0.02	–	$= 2$	0.00	**0.00**	**0.00**	**0.01**	–	81.97
`Escp.4×4`	224	738	16	$= 2$	0.08	11.66	11.73	1059.23	–	$= 2$	0.08	**0.13**	**0.22**	**0.25**	–	99.02
`Escp.4×5`	360	1326	20	$= 2$	0.18	34.09	33.80	3084.58	–	$= 2$	0.18	**0.27**	**0.46**	**0.63**	–	99.02
`Escp.5×5`	598	2301	26	≥ 2	0.46	96.24	97.10	?	?	$\boldsymbol{= 2}$	0.46	**0.68**	**1.16**	**1.71**	–	98.98
`Escp.5×6`	840	3516	30	≥ 2	1.01	217.63	216.83	?	?	$\boldsymbol{= 2}$	**1.00**	**1.42**	**2.40**	**4.30**	–	99.00
`Escp.6×6`	1224	5424	36	≥ 2	2.13	516.92	511.41	?	?	$\boldsymbol{= 2}$	**2.06**	**2.90**	**5.12**	**10.30**	–	98.97
`Escp.7×7`	2350	11097	50	≥ 2	7.81	2167.86	2183.01	?	?	$\boldsymbol{= 2}$	**7.71**	**10.67**	**19.04**	**52.47**	–	98.99
`Escp.7×8`	3024	14820	56	≥ 0	**13.07**	?	?	?	?	$\boldsymbol{= 2}$	13.44	**18.25**	**32.69**	**108.60**	–	99.01

Benchmark		Reduction + Yosys + SafetySynth (symbolic)							Incremental Synthesis (explicit-state implementation)							
name	$\delta_{\max}$	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	%
`Stub.4×4`	$= 2$	1.07	1.24	1.24	1.80	–	–	–	**0.04**	**0.07**	**0.12**	**0.18**	–	–	–	98.98
`Stub.4×5`	$= 2$	1.16	1.49	1.49	2.83	–	–	–	**0.08**	**0.14**	**0.25**	**0.44**	–	–	–	98.97
`Stub.5×5`	$= 2$	1.19	2.61	2.50	13.67	–	–	–	**0.21**	**0.37**	**0.63**	**1.17**	–	–	–	98.97
`Stub.5×6`	$= 2$	1.18	2.60	2.59	23.30	–	–	–	**0.42**	**0.69**	**1.20**	**2.49**	–	–	–	98.96
`Stub.6×6`	$= 4$	1.17	2.76	2.74	19.96	19.69	655.24	–	**0.93**	**1.47**	**2.60**	**5.79**	**7.54**	**7.60**	–	99.89
`Stub.7×7`	$= 4$	**1.23**	**2.50**	**2.48**	24.57	**23.01**	2224.62	–	3.60	5.52	10.08	**22.75**	31.18	**32.98**	–	99.88

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

Outline

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.