Modelling · Verification · Synthesis

A Peek into the Blueprint of Hybrid Systems

Mingshuai Chen

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

State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, China

Aachen · October 2018

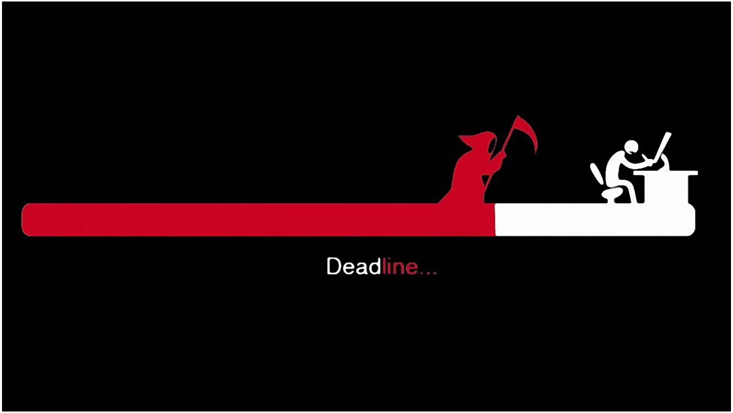
How I Feel Every Time Being Asked to Give a Self-Intro. . . .

About Me ...

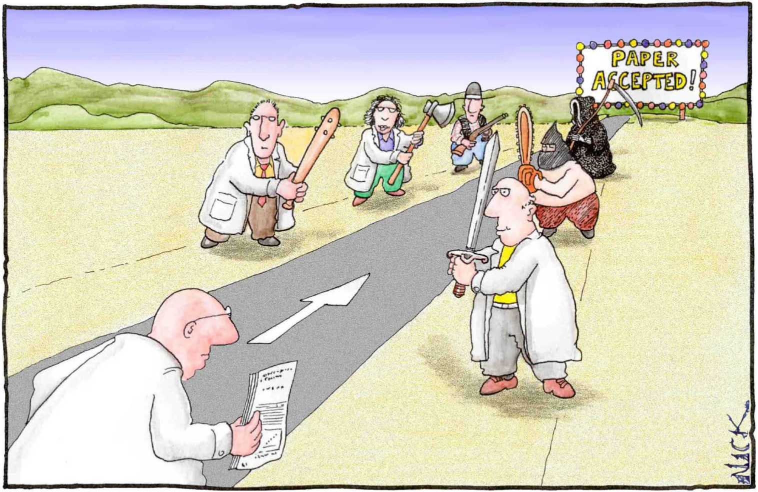
About Me ...

About Me ...

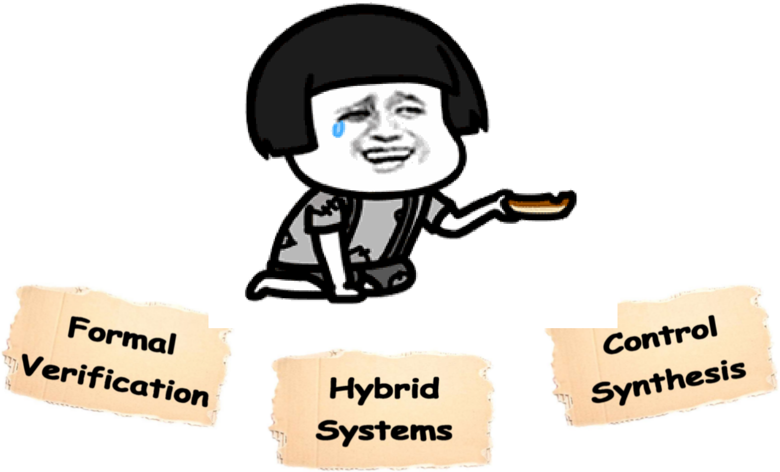
About Me ...



About Me ...

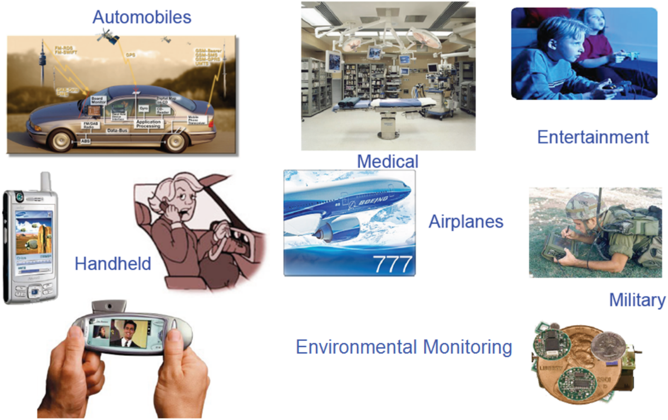


Seeking for a Postdoc Position ...



Hybrid Systems

Hybrid systems exhibit combinations of discrete jumps and continuous evolution, many of which are safety-critical.



Hybrid Behaviours

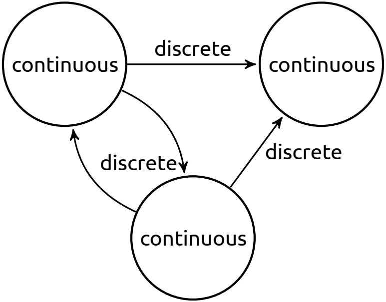


Figure – Macro : switching modes

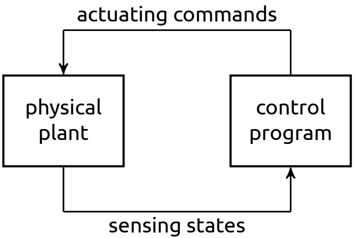
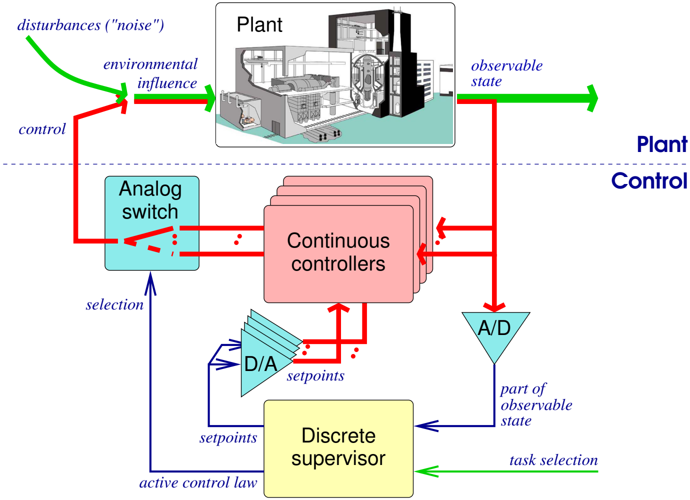
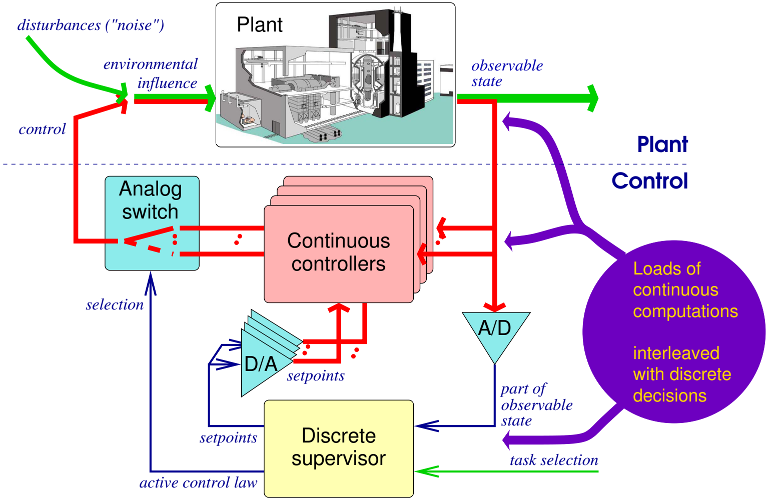


Figure – Micro : closed-loop feedback

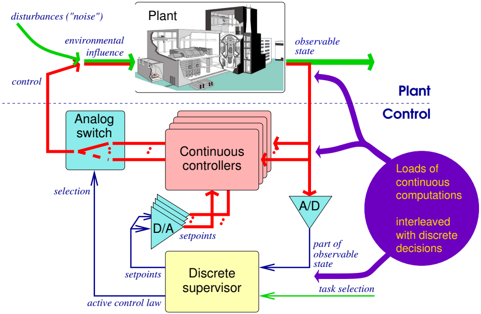
Hybrid Systems



Hybrid Systems



Hybrid Systems



Crucial question :

- How do formal methods guarantee critical properties, e.g., safety, termination, liveness etc. ?

Main answers :

- Theorem proving (automated/interactive deductive-reasoning).
- Model checking (exhaustive state-exploration).
- Synthesis (correct-by-construction).

Outline

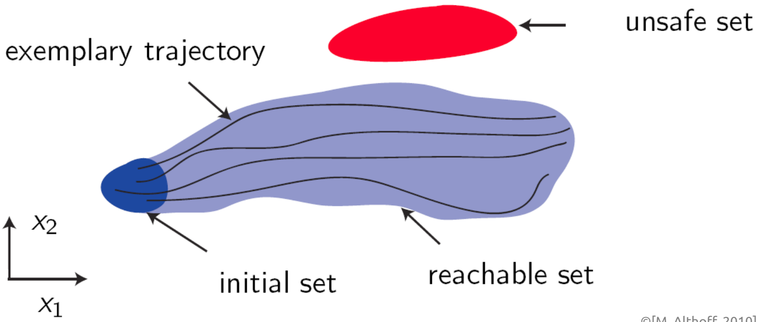
Reachability of Differential Dynamics

The most expressive family whose reachability is decidable

—Joint work with T. Gan, Y. Li, L. Dai, B. Xia and N. Zhan—

Outline

Safety Verification Using Reachable Sets



©[M. Althoff, 2010]

- System is safe, if no trajectory enters the unsafe set.

LDSs with Inputs

- Linear dymamical systems (LDSs) with inputs are differential equations of the form

$$\dot{\xi} = A\xi + \mathbf{u},$$

where $\xi(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$, and $\mathbf{u} : \mathbb{R} \to \mathbb{R}^n$ is a continuous function vector which is called the *input*.

LDSs with Inputs

- Linear dymamical systems (LDSs) with inputs are differential equations of the form

$$\dot{\xi} = A\xi + \mathbf{u},$$

where $\xi(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, and $\mathbf{u} : \mathbb{R} \to \mathbb{R}^n$ is a continuous function vector which is called the *input*.

- The forward reachable set :

$$Post(\mathrm{X}) := \{\mathbf{y} \in \mathbb{R}^n \mid \exists \mathbf{x} \exists t : \mathbf{x} \in \mathrm{X} \wedge t \geq 0 \wedge \Phi(\mathbf{x}, t) = \mathbf{y}\}$$

LDSs with Inputs

- Linear dymamical systems (LDSs) with inputs are differential equations of the form

$$\dot{\xi} = A\xi + \mathbf{u},$$

where $\xi(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, and $\mathbf{u} : \mathbb{R} \to \mathbb{R}^n$ is a continuous function vector which is called the *input*.

- The forward reachable set :

$$Post(\mathrm{X}) := \{\mathbf{y} \in \mathbb{R}^n \mid \exists \mathbf{x} \exists t : \mathbf{x} \in \mathrm{X} \wedge t \geq 0 \wedge \Phi(\mathbf{x}, t) = \mathbf{y}\}$$

- **Reachability problem :**

$$\mathcal{F}(\mathrm{X}, \mathrm{Y}) := \mathrm{Y} \cap Post(\mathrm{X}) = \emptyset \quad ?$$

Decidability Results of the Reachability of LDSs

In [G. Lafferriere *et al.*, J. Symb. Comput., 2001], Lafferriere, Pappas and Yovine proved the decidability of the reachability problems of the following three families of LDSs :

1. A is nilpotent, i.e. $A^n = 0$, and each component of $\mathbf{u}$ is a polynomial ;
2. A is diagonalizable with rational eigenvalues, and each component of $\mathbf{u}$ is of the form $\sum_{i=1}^{m} c_i e^{\lambda_i t}$, where λ_is are rationals and c_is are subject to semi-algebraic constraints ;
3. A is diagonalizable with purely imaginary eigenvalues, and each component of $\mathbf{u}$ of the form $\sum_{i=1}^{m} c_i \sin(\lambda_i t) + d_i \cos(\lambda_i t)$, where λ_is are rationals and c_is and d_is are subject to semi-algebraic constraints.

Our Contributions

We generalize the previous case 2 and case 3 by proving the decidability of the reachability problems where

2. A is diagonalizable with ~~rational~~ real eigenvalues, and each component of $\mathbf{u}$ is of the form $\sum_{i=1}^{m} c_i e^{\lambda_i t}$, where λ_is are ~~rationals~~ reals and c_is are subject to semi-algebraic constraints;

⇒ T. Gan, M. Chen, L. Dai, B. Xia, N. Zhan : *Decidabil. of the reachabil. for a family of linear vector fields.* ATVA'15.

3. A is diagonalizable with purely imaginary eigenvalues, and each component of $\mathbf{u}$ of the form $\sum_{i=1}^{m} c_i \sin(\lambda_i t) + d_i \cos(\lambda_i t)$, where λ_is are ~~rationals~~ reals and c_is and d_is are subject to semi-algebraic constraints.

⇒ T. Gan, M. Chen, Y. Li, B. Xia, N. Zhan : *Computing reachable sets of linear vector fields revisited.* ECC'16.

Solvable Systems

A nonlinear differential dynamic

$$\dot{\xi} = F(\xi, \mathbf{u})$$

is called solvable system (SS) if the variable vector $\xi = (\xi_1, \ldots, \xi_n)$ can be classified into m groups ($m \leq n$) :

$$\zeta_1 = (\xi_{11}, \ldots, \xi_{1n_1}), \ldots, \zeta_m = (\xi_{m1}, \ldots, \xi_{mn_m}),$$

and the dynamical system can be represented as the form :

$$\dot{\xi} = \begin{bmatrix} \dot{\zeta}_1 \\ \dot{\zeta}_2 \\ \vdots \\ \dot{\zeta}_m \end{bmatrix} = \begin{bmatrix} A_1\zeta_1 + \mathbf{u}_1(t) \\ A_2\zeta_2 + \mathbf{u}_2(t, \zeta_1) \\ \vdots \\ A_m\zeta_m + \mathbf{u}_m(t, \zeta_1, \ldots, \zeta_{m-1}) \end{bmatrix},$$

where $0 < n_1 < \ldots < n_m = n$ are integers, $m \in \mathbb{N}$, $A_1, \ldots, A_m$ are real matrices with corresponding dimensions, $\mathbf{u}_1, \ldots, \mathbf{u}_m$ are *polynomial-exponential-trigonometric functions.*

Solvable Systems

A nonlinear differential dynamic

$$\dot{\xi} = F(\xi, \mathbf{u})$$

is called solvable system (SS) if the variable vector $\xi = (\xi_1, \ldots, \xi_n)$ can be classified into m groups ($m \leq n$) :

$$\zeta_1 = (\xi_{11}, \ldots, \xi_{1n_1}), \ldots, \zeta_m = (\xi_{m1}, \ldots, \xi_{mn_m}),$$

and the dynamical system can be represented as the form :

Example (Solvable System)

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} x + e^{-t} \\ 2y + x^2 - e^{-\sqrt{2}t} \\ \sqrt{3}z + xy + 2e^{-t} \end{bmatrix}$$

where $0 < n_1 < \ldots < n_m = n$ are integers, $m \in \mathbb{N}$, $A_1, \ldots, A_m$ are real matrices with corresponding dimensions, $\mathbf{u}_1, \ldots, \mathbf{u}_m$ are *polynomial-exponential-trigonometric functions*.

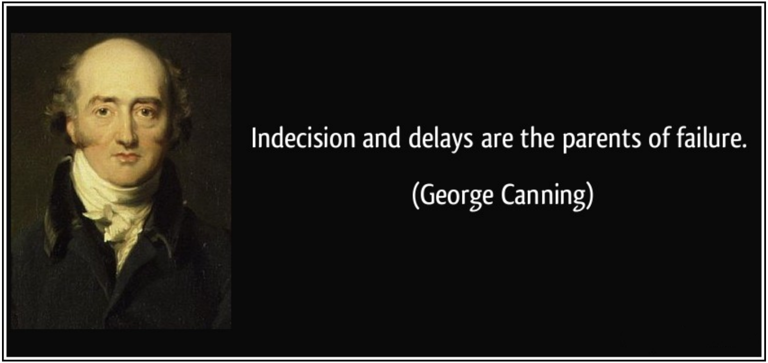
Our Contributions (Cont'd)

- We generalize the decidability of reachability from LDSs to SSs where
  1. $A_1, \ldots, A_m$ are nilpotent, *i.e.* $A_1^{k_1} = 0, \ldots, A_m^{k_m} = 0$, for some $k_1, \ldots, k_m \in \mathbb{N}$, and each component of $\mathbf{u}_i$ is a polynomial;
  2. Each A_i is diagonalizable with real eigenvalues, and each component of $\mathbf{u}_i$ is of the form $\sum_{j=1}^{m_i} c_{ij} e^{\lambda_{ij} t}$, where λ_{ij}s are reals and c_{ij}s are subject to semi-algebraic constraints;
  3. Each A_i is diagonalizable with purely imaginary eigenvalues, whose imaginary parts are reals, and each component of $\mathbf{u}_i$ of the form $\sum_{j=1}^{m_i} c_{ij} \sin(\lambda_{ij} t) + d_{ij} \cos(\lambda_{ij} t)$, where λ_{ij}s are reals and c_{ij}s and d_{ij}s are subject to semi-algebraic constraints.
- We further present a tight abstraction of general solvable dynamical systems, where the system matrix A may have complex eigenvalues.

⇒ T. Gan, M. Chen, Y. Li, B. Xia, N. Zhan : *Reachability analysis for solvable dynamical systems.* IEEE Trans. Automat. Contr. 2017.

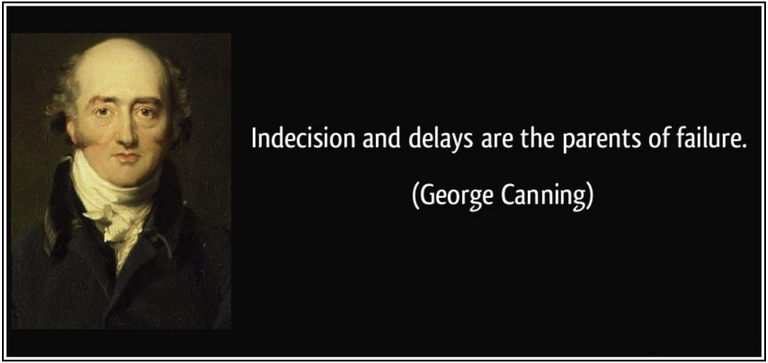
Outline

Advice by a Wise Man



©izQuotes

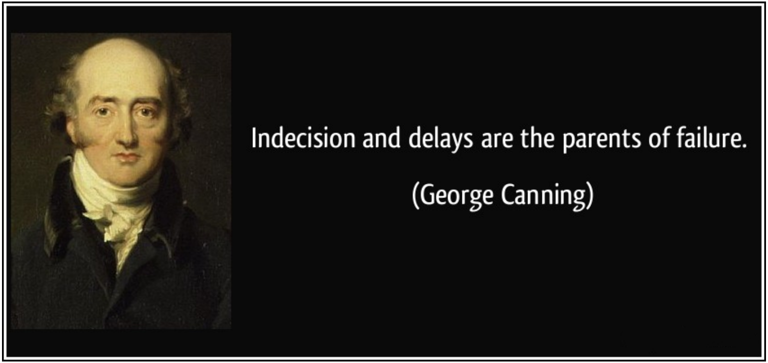
Advice by a Wise Man



©izQuotes

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

Advice by a Wise Man

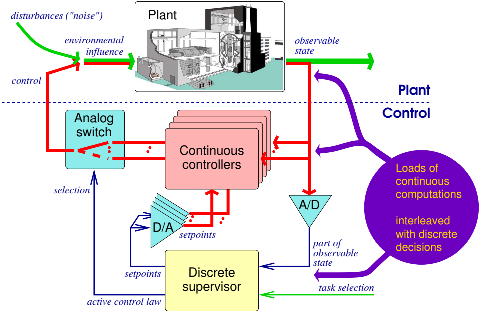


©izQuotes

- 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!

Hybrid Systems



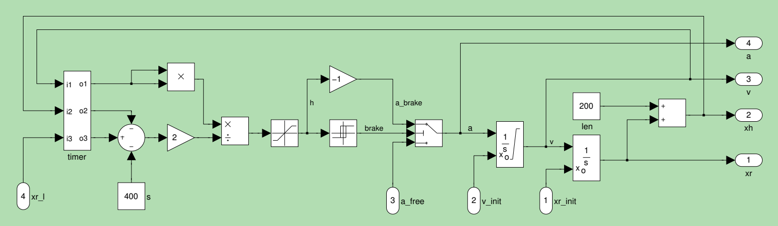
Crucial question :

- How do the controller and the plant interact?

Traditional answer :

- Coupling assumed to be (or at least modelled 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

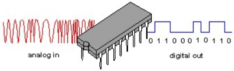
Following the tradition, above (rather typical) Simulink model assumes

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

Central questions :

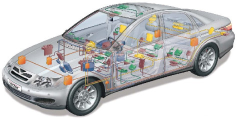
1. Is this realistic?
2. If not, does it have observable effect on control performance?
3. May that effect be detrimental or even harmful?

Q1 : Is Instantaneous Coupling Realistic?



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

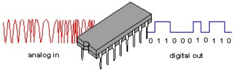
Digital signal processing, especially in complex sensors like CV, needs processing time, adding signal delays.



Networked control introduces communication latency into the feedback control loop.

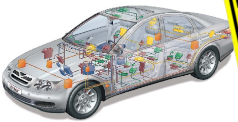
Harvesting, fusing, and forwarding data through sensor networks enlarge the latter by orders of magnitude.

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- l…



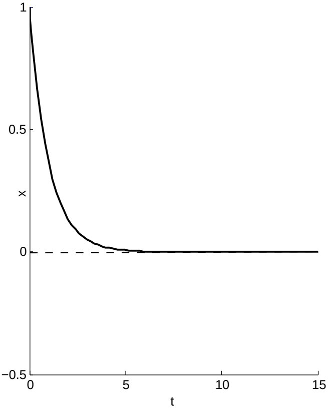
…s communication la- … loop.

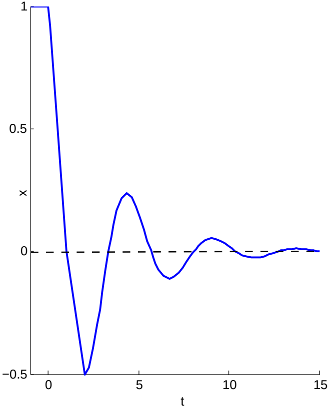
Harvesting, fusing, and forwarding data through sensor networks enlarge the latter by orders of magnitude.

Q2 : Do Delays Have Observable Effect?

$$\begin{cases} \dot{x}(t) = -x(t) \\ x(0) = 1 \end{cases}$$

$$\begin{cases} \dot{x}(t) = -x(t-1) \\ x([-1,0]) \equiv 1 \end{cases}$$

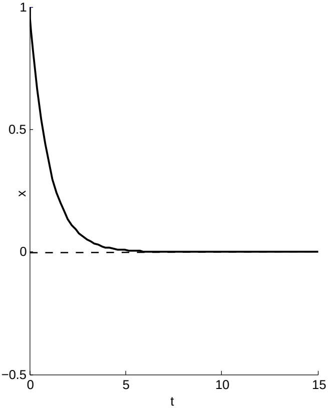


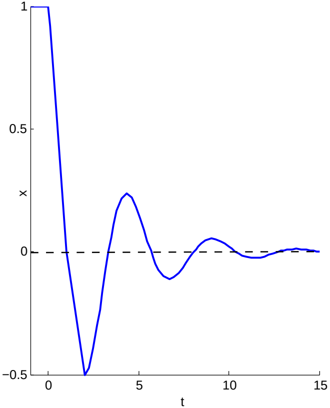


Q2 : Do Delays Have Observable Effect? – Yes, they have.

$$\begin{cases} \dot{x}(t) = -x(t) \\ x(0) = 1 \end{cases}$$

$$\begin{cases} \dot{x}(t) = -x(t-1) \\ x([-1,0]) \equiv 1 \end{cases}$$

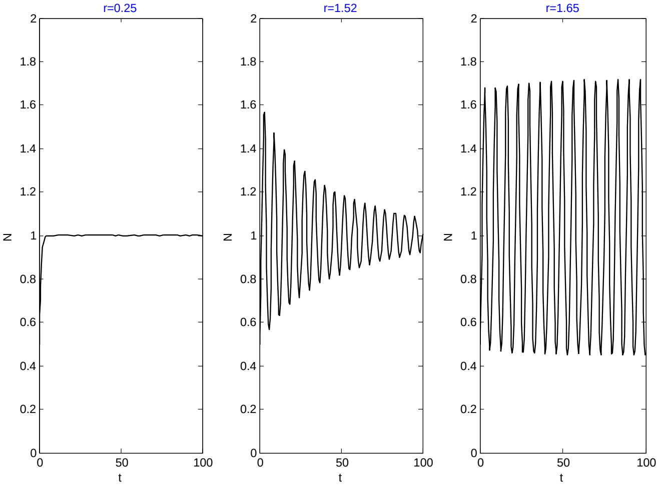




Q3 : May the Effects be Harmful?

- Delayed logistic equation [G. Hutchinson, 1948] :

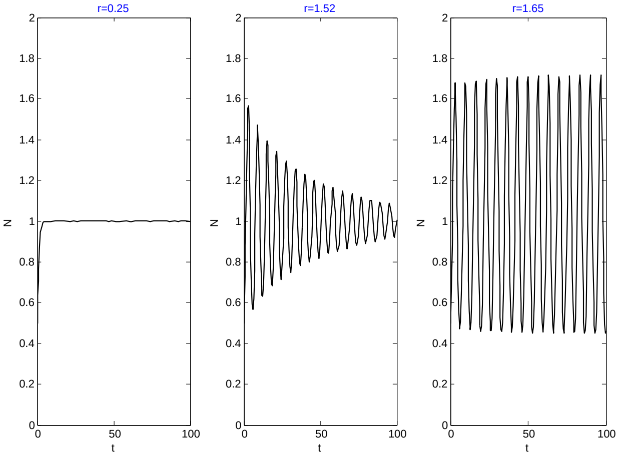
$$\dot{N}(t) = N(t)[1 - N(t - r)]$$



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

- Delayed logistic equation [G. Hutchinson, 1948] :

$$\dot{N}(t) = N(t)[1 - N(t - r)]$$



Consequences

- Delays in feedback control loops are ubiquitous.
- They may well invalidate the safety/stability/...certificates obtained by verifying delay-free abstractions of the feedback control systems.

Automatic verification/synthesis methods addressing feedback delays in hybrid systems should therefore abound!

Consequences

- Delays in feedback control loops are ubiquitous.
- They may well invalidate the safety/stability/...certificates obtained by verifying delay-free abstractions of the feedback control systems.

Automatic verification/synthesis methods addressing feedback delays in hybrid systems should therefore abound!

Surprisingly, they don't :

1. S. Prajna, A. Jadbabaie : *Meth. f. safety verification of time-delay syst.* (CDC'05)
2. L. Zou, M. Fränzle, N. Zhan, P.N. Mosaad : *Autom. verific. of stabil. and safety* (CAV'15)
3. H. Trinh, P.T. Nam, P.N. Pathirana, H.P. Le : *On bwd.s and fwd.s reachable sets bounding for perturbed time-delay systems* (Appl. Math. & Comput. 269, '15)
4. Z. Huang, C. Fan, S. Mitra : *Bounded invariant verification for time-delayed nonlinear networked dynamical systems* (NAHS'16)
5. P.N. Mosaad, M. Fränzle, B. Xue : *Temporal logic verification for DDEs* (ICTAC'16)
6. M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *Validat. simul.-based verific.* (FM'16)
7. B. Xue, P.N. Mosaad, M. Fränzle, M. Chen, Y. Li, N. Zhan : *Safe approx. of reachable sets for DDEs* (FORMATS'17)
8. E. Goubault, S. Putot, L. Sahlman : *Approximating flowpipes for DDEs* (CAV'18)

(plus a handful of related versions)

Solving Delay Differential Equations (DDEs)

A formal model of delayed feedback control

—Joint work with M. Fränzle, Y. Li, P. N. Mosaad, B. Xue and N. Zhan—

Delayed Differential Dynamics (a.k.a., DDEs)

Historical motivation :

> *"Despite [...] very satisfactory state of affairs as far as [ordinary] differential equations are concerned, we are nevertheless forced to turn to the study of more complex equations. Detailed studies of the real world impel us, albeit reluctantly, to take account of the fact that the rate of change of physical systems depends not only on their present state, but also on their past history."*

[Richard Bellman and Kenneth L. Cooke, 1963]

Delayed Differential Dynamics (a.k.a., DDEs)

Historical motivation :

"Despite [...] very satisfactory state of affairs as far as [ordinary] differential equations are concerned, we are nevertheless forced to turn to the study of more complex equations. Detailed studies of the real world impel us, albeit reluctantly, to take account of the fact that the rate of change of physical systems depends not only on their present state, but also on their past history."

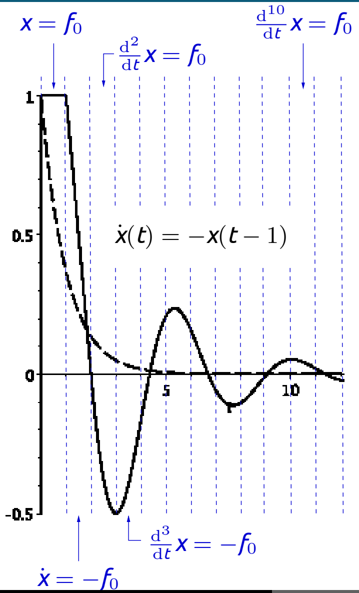
[Richard Bellman and Kenneth L. Cooke, 1963]

Delay Differential Equations (DDEs)

$$\begin{cases} \dot{\mathbf{x}}(t) = \boldsymbol{f}(\mathbf{x}(t), \mathbf{x}(t-r_1), \ldots, \mathbf{x}(t-r_k)), & t \in [0, \infty) \\ \mathbf{x}(t) = \mathbf{x}_0 \in \Theta, \quad t \in [-r_{\max}, 0] \end{cases}$$

The unique *solution* (*trajectory*) : $\xi_{\mathbf{x}_0}(t) : [-r_{\max}, \infty) \mapsto \mathbb{R}^n$.

Why DDEs are Hard(er)

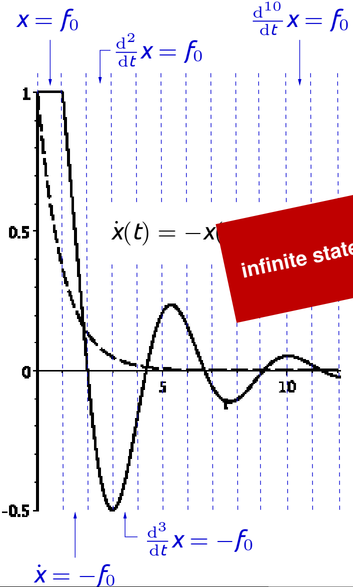


DDEs constitute a model of system dynamics beyond "state snapshots" :

- They feature "functional state" instead of state in the $\mathbb{R}^n$.
- Thus providing rather infallible, infinite-dimensional memory of the past.

N.B. : More complex transformations may be applied to the initial segment f_0 according to the DDE's right-hand side. f_0 will nevertheless hardly ever vanish from the state space.

Why DDEs are Hard(er)



Try only if infinite state no longer is scary enough to you!

DDEs co… model of system …"state snapshots" :

- …functional state"
- …state in the $\mathbb{R}^n$.
- Thus providing rather infallible, infinite-dimensional memory of the past.

N.B. : More complex transformations may be applied to the initial segment f_0 according to the DDE's right-hand side. f_0 will nevertheless hardly ever vanish from the state space.

Method I : Simulation-Based Verification

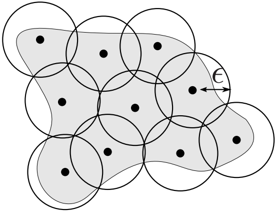


Figure – A finite ϵ-cover of the initial set of states.

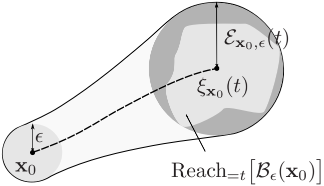
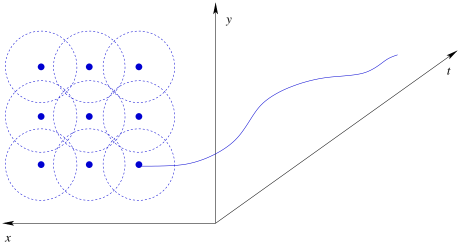


Figure – An Over-approximation of the reachable set by bloating the simulation.

Method I : Simulation-Based Verification

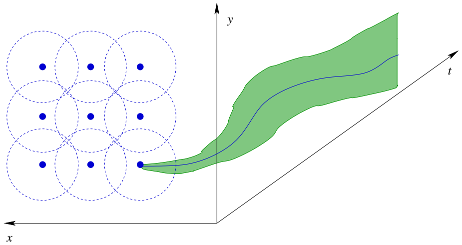
1. Do numerical simulation on a (sufficiently dense) sample of initial states.
2. Add (pessimistic) error analysis and sensitivity analysis.
3. "Bloat" the resulting trajectories accordingly.



⇒ M. Chen, M. Fränzle, Y. Li, P. N. Mosaad, N. Zhan : *Validat. simul.-based verific.*. FM'16.

Method I : Simulation-Based Verification

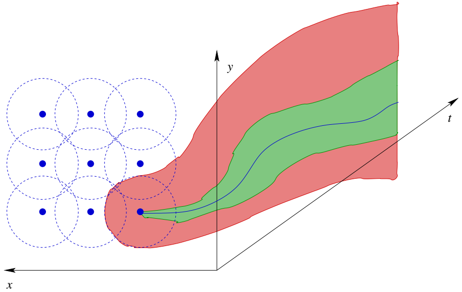
1. Do numerical simulation on a (sufficiently dense) sample of initial states.
2. Add (pessimistic) error analysis and sensitivity analysis.
3. "Bloat" the resulting trajectories accordingly.



⇒ M. Chen, M. Fränzle, Y. Li, P. N. Mosaad, N. Zhan : *Validat. simul.-based verific.*. FM'16.

Method I : Simulation-Based Verification

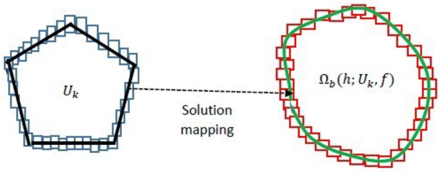
1. Do numerical simulation on a (sufficiently dense) sample of initial states.
2. Add (pessimistic) error analysis and sensitivity analysis.
3. "Bloat" the resulting trajectories accordingly.



⇒ M. Chen, M. Fränzle, Y. Li, P. N. Mosaad, N. Zhan : *Validat. simul.-based verific.*. FM'16.

Method II : Boundary-Based Approximation

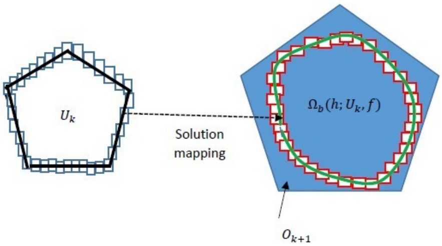
1. Impose a homeomorphism by bounding the time-lag through sensitivity analysis.
2. Compute an enclosure of the reachable set's boundary.
3. Over- (under-)approximate the reachable set by incl. (excl.) the enclosure.



⇒ B. Xue, P. Mosaad, M. Fränzle, M. Chen, Y. Li, N. Zhan : *Safe approx. of reachable sets for DDEs.* FORMATS '17.

Method II : Boundary-Based Approximation

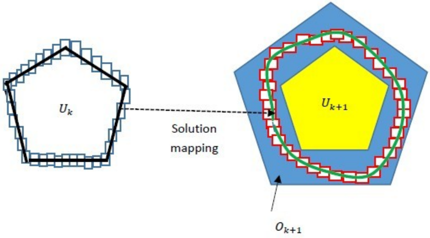
1. Impose a homeomorphism by bounding the time-lag through sensitivity analysis.
2. Compute an enclosure of the reachable set's boundary.
3. Over- (under-)approximate the reachable set by incl. (excl.) the enclosure.



⇒ B. Xue, P. Mosaad, M. Fränzle, M. Chen, Y. Li, N. Zhan : *Safe approx. of reachable sets for DDEs.* FORMATS'17.

Method II : Boundary-Based Approximation

1. Impose a homeomorphism by bounding the time-lag through sensitivity analysis.
2. Compute an enclosure of the reachable set's boundary.
3. Over- (under-)approximate the reachable set by incl. (excl.) the enclosure.



⇒ B. Xue, P. Mosaad, M. Fränzle, M. Chen, Y. Li, N. Zhan : *Safe approx. of reachable sets for DDEs.* FORMATS '17.

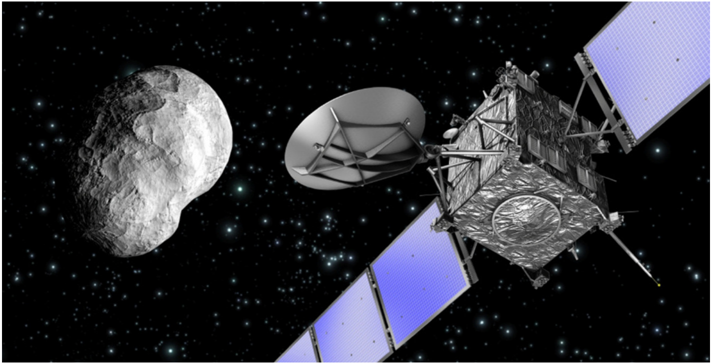
Discrete Safety Games

Staying safe and reaching an objective
when observation & actuation are confined by delays

—Joint work with M. Fränzle, Y. Li, P. N. Mosaad and N. Zhan—

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 Robot-Escaping Game

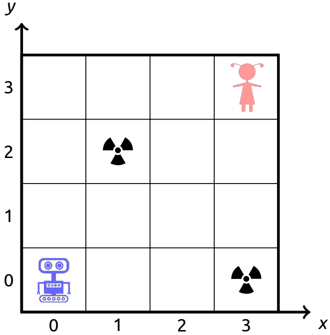


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

A Robot-Escaping Game

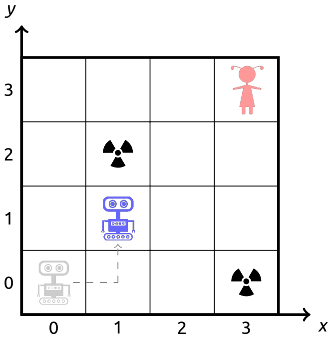


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

A Robot-Escaping Game

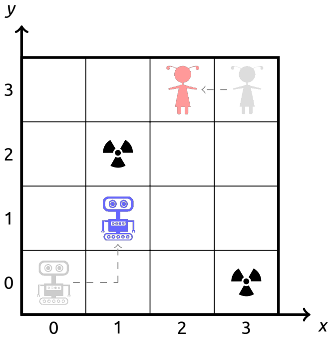


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

A Robot-Escaping Game

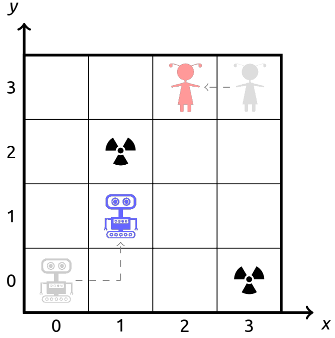


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

No delay :

A Robot-Escaping Game

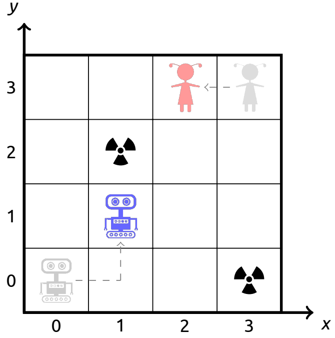


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

No delay :

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

A Robot-Escaping Game

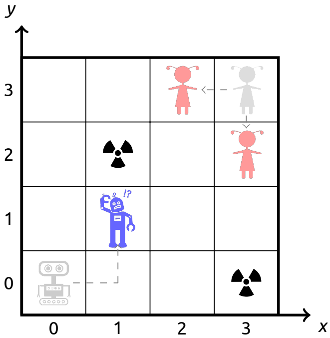


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

No delay :

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

1 step delay :

A Robot-Escaping Game

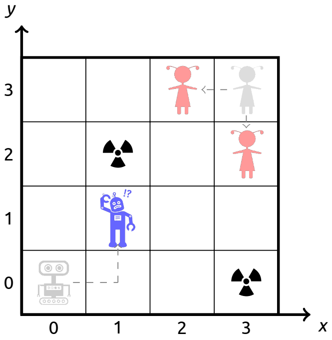


Figure – A robot escape game in a 4 × 4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

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

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

A Robot-Escaping Game

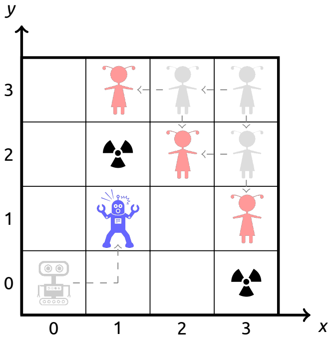


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\texttt{RU}, \texttt{UR}, \texttt{LU}, \texttt{UL}, \texttt{RD}, \texttt{DR}, \texttt{LD}, \texttt{DL}, \epsilon\}$,
$\Sigma_k = \{\texttt{R}, \texttt{L}, \texttt{U}, \texttt{D}\}$.

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

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

2 steps delay :

A Robot-Escaping Game

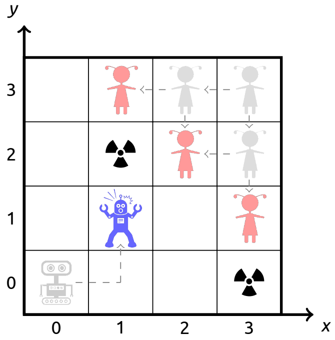


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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.

A Robot-Escaping Game

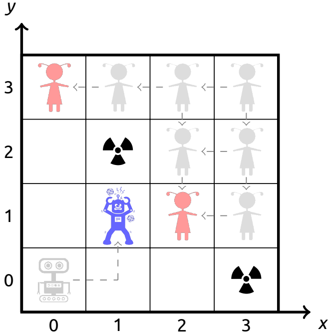


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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 :

A Robot-Escaping Game

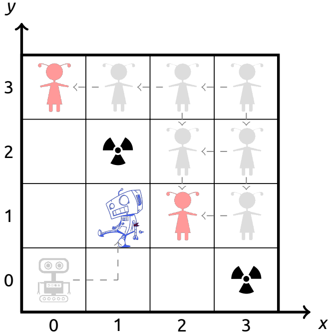


Figure – A robot escape game in a 4×4 room, with $\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$, $\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

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.

A Robot-Escaping Game

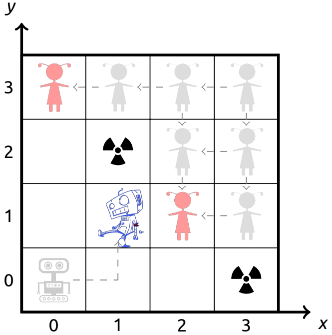


Figure – A robot escape game in a 4×4 room, with
$\Sigma_r = \{\mathtt{RU}, \mathtt{UR}, \mathtt{LU}, \mathtt{UL}, \mathtt{RD}, \mathtt{DR}, \mathtt{LD}, \mathtt{DL}, \epsilon\}$,
$\Sigma_k = \{\mathtt{R}, \mathtt{L}, \mathtt{U}, \mathtt{D}\}$.

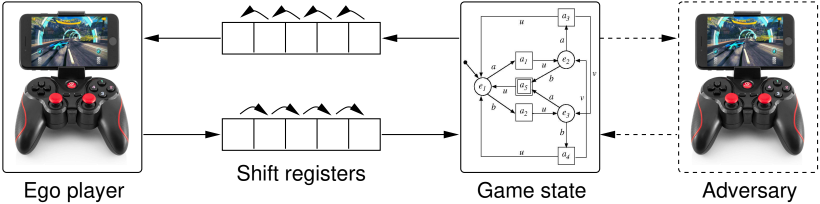
No delay :
Robot always wins by circling around the obstacle at (1,2).

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

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

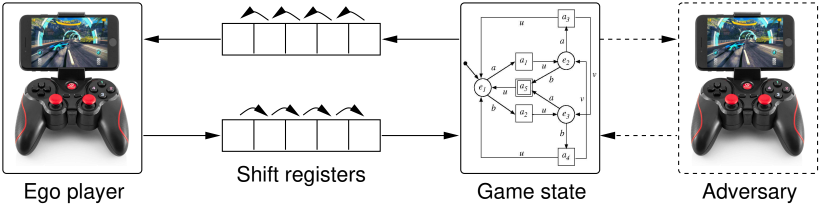
3 steps delay :
Robot is unwinnable (uncontrollable) anymore.

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



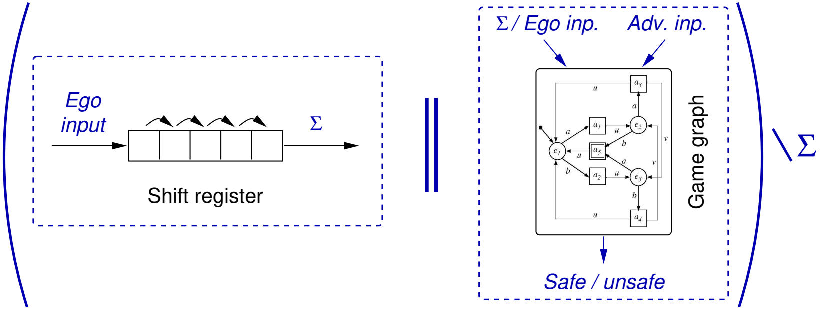
Observation : It doesn't make an observable difference for the joint dynamics whether delay occurs in perception, actuation, or both.

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!*

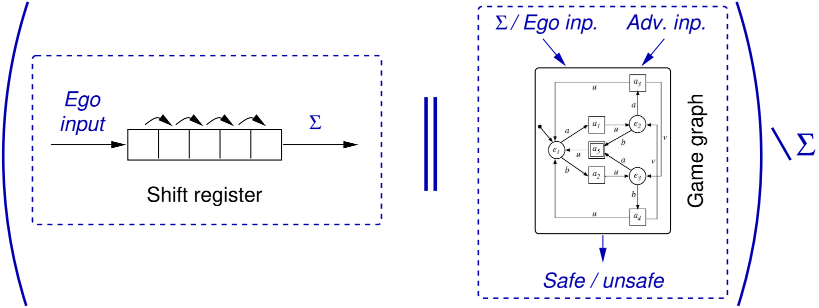
Reduction to Delay-Free Games

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 $|\text{Alphabet(ego)}|^{\text{delay}}$.

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction*. ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

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

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction.* ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

Consequence : 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 :

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction*. ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

Consequence : 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 ;

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction.* ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

Consequence : 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 $k+1$;

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction.* ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

Consequence : 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 ;

⇒ M. Chen, M. Fränzle, Y. Li, P.N. Mosaad, N. Zhan : *What's to come is still unsure : Synthesizing controllers resilient to delayed interaction.* ATVA '18. [Distinguished Paper Award].

Incremental Synthesis

Observation : A winning strategy for delay $k' > k$ can always be utilized for a safe win under delay k.

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

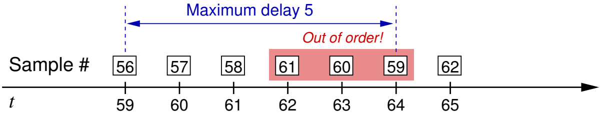
Idea : Incrementally filter out loss states & incrementally synthesize winning strategy for the remaining :

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.* ATVA '18. [Distinguished Paper Award].

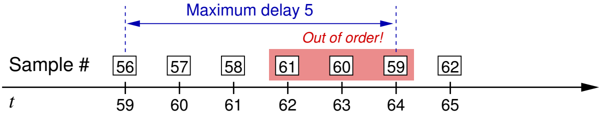
How about Non-Order-Preserving Delays?

☹ Observations may arrive out-of-order :

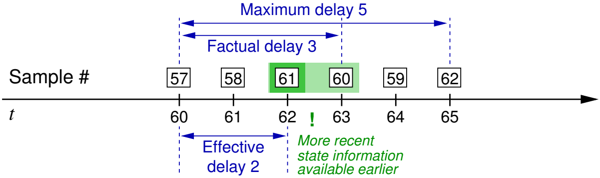


How about Non-Order-Preserving Delays?

☹ Observations may arrive out-of-order:

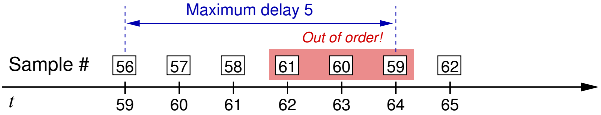


☺ But this may only reduce effective delay, improving controllability:

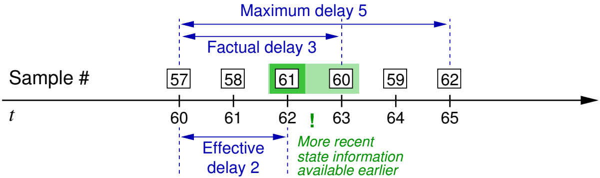


How about Non-Order-Preserving Delays?

- ☹ 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

Interpolation over Nonlinear Arithmetic

The cornerstone of ATP, SMT, BMC, etc.

**—Joint work with T. Gan, L. Dai, B. Xia, N. Zhan,
D. Kapur, J. Wang and J. An—**

Craig Interpolation

Craig Interpolant

Given ϕ and ψ in a theory $\mathcal{T}$ s.t. $\phi \wedge \psi \models_{\mathcal{T}} \bot$, a formula I is a *(reverse) interpolant* of ϕ and ψ if

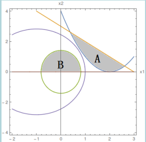
1. $\phi \models_{\mathcal{T}} I$;
2. $I \wedge \psi \models_{\mathcal{T}} \bot$; and
3. $\mathit{var}(I) \subseteq \mathit{var}(\phi) \cap \mathit{var}(\psi)$.

Craig Interpolation

Craig Interpolant

Given ϕ and ψ in a theory $\mathcal{T}$ s.t. $\phi \wedge \psi \models_{\mathcal{T}} \bot$, a formula I is a *(reverse) interpolant* of ϕ and ψ if

1. $\phi \models_{\mathcal{T}} I$;
2. $I \wedge \psi \models_{\mathcal{T}} \bot$; and
3. $var(I) \subseteq var(\phi) \cap var(\psi)$.

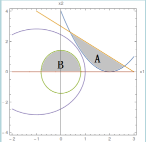


Craig Interpolation

Craig Interpolant

Given ϕ and ψ in a theory $\mathcal{T}$ s.t. $\phi \wedge \psi \models_{\mathcal{T}} \bot$, a formula I is a *(reverse) interpolant* of ϕ and ψ if

1. $\phi \models_{\mathcal{T}} I$;
2. $I \wedge \psi \models_{\mathcal{T}} \bot$; and
3. $var(I) \subseteq var(\phi) \cap var(\psi)$.



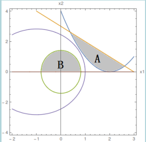
- Nelson-Oppen method in theorem proving : local and modular reasoning;
- SMT : combining different decision procedures to verify programs with complicated data structures;
- Bounded model-checking : generating invariants to verify infinite-state systems due to McMillan;
- ...

Craig Interpolation

Craig Interpolant

Given ϕ and ψ in a theory $\mathcal{T}$ s.t. $\phi \wedge \psi \models_{\mathcal{T}} \bot$, a formula I is a *(reverse) interpolant* of ϕ and ψ if

1. $\phi \models_{\mathcal{T}} I$;
2. $I \wedge \psi \models_{\mathcal{T}} \bot$; and
3. $var(I) \subseteq var(\phi) \cap var(\psi)$.



- Nelson-Oppen method in theorem proving : local and modular reasoning ;
- SMT : combining different decision procedures to verify programs with complicated data structures ;
- Bounded model-checking : generating invariants to verify infinite-state systems due to McMillan ;
- ...
- Little work on synthesizing nonlinear interpolants : [S. Kupferschmid and B. Becker, FORMATS'11].

Our Contributions

1. A complete, polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic (*CQ*) polynomial inequalities over the reals :

⇒ T. Gan, L. Dai, B. Xia, N. Zhan, D. Kapur, M. Chen : *Interpolant synthesis for quadratic polynomial inequalities and combination with EUF*. IJCAR'16.

Our Contributions

1. A complete, polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic (*CQ*) polynomial inequalities over the reals :
   - If $\mathbf{NSOSC}$ holds, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials, where a generalization of Motzkin's transposition theorem applies ;

⇒ T. Gan, L. Dai, B. Xia, N. Zhan, D. Kapur, M. Chen : *Interpolant synthesis for quadratic polynomial inequalities and combination with EUF*. IJCAR'16.

Our Contributions

1. A complete, polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic (*CQ*) polynomial inequalities over the reals :
   - If $\mathbf{NSOSC}$ holds, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials, where a generalization of Motzkin's transposition theorem applies ;
   - If $\mathbf{NSOSC}$ doesn't hold, linear equalities relating variables are deduced, resulting to interpolation subproblems with fewer variables on which the algorithm is recursively applied.

⇒ T. Gan, L. Dai, B. Xia, N. Zhan, D. Kapur, M. Chen : *Interpolant synthesis for quadratic polynomial inequalities and combination with EUF*. IJCAR'16.

Our Contributions

1. A complete, polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic (*CQ*) polynomial inequalities over the reals :
   - If NSOSC holds, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials, where a generalization of Motzkin's transposition theorem applies;
   - If NSOSC doesn't hold, linear equalities relating variables are deduced, resulting to interpolation subproblems with fewer variables on which the algorithm is recursively applied.
2. An algorithm, by partitioning Horn clauses, for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and equality theory over uninterpreted function symbols (*EUF*);

⇒ T. Gan, L. Dai, B. Xia, N. Zhan, D. Kapur, M. Chen : *Interpolant synthesis for quadratic polynomial inequalities and combination with EUF*. IJCAR'16.

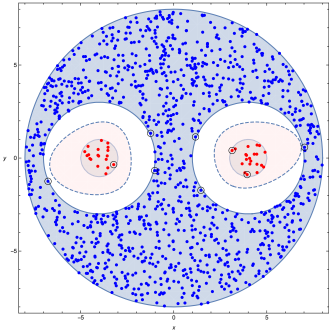
Our Contributions

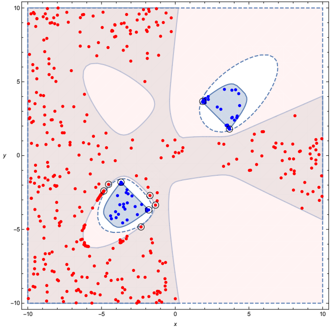
1. A complete, polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic (*CQ*) polynomial inequalities over the reals :
   - If NSOSC holds, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials, where a generalization of Motzkin's transposition theorem applies;
   - If NSOSC doesn't hold, linear equalities relating variables are deduced, resulting to interpolation subproblems with fewer variables on which the algorithm is recursively applied.
2. An algorithm, by partitioning Horn clauses, for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and equality theory over uninterpreted function symbols (*EUF*);
3. Tool NLFIntp : `lcs.ios.ac.cn/~chenms/tools/NLFIntp/`

⇒ T. Gan, L. Dai, B. Xia, N. Zhan, D. Kapur, M. Chen : *Interpolant synthesis for quadratic polynomial inequalities and combination with EUF*. IJCAR'16.

Our Contributions (Cont'd)

We drop the CQ constraint by learning nonlinear interpolants using SVM classification (sampling-guessing-refining) :

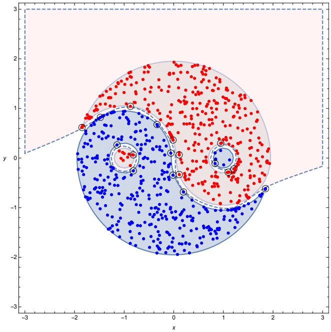


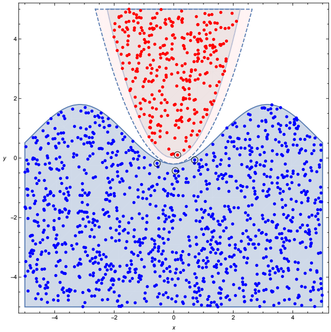


⇒ M. Chen, J. Wang, J. An, D. Kapur, N. Zhan : *NIL : Learning nonlinear interpolants*. Under revision.

Our Contributions (Cont'd)

We drop the CQ constraint by learning nonlinear interpolants using SVM classification (sampling-guessing-refining) :

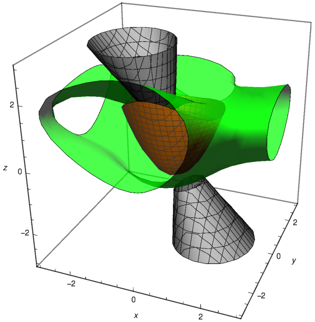




⇒ M. Chen, J. Wang, J. An, D. Kapur, N. Zhan : *NIL : Learning nonlinear interpolants*. Under revision.

Our Contributions (Cont'd)

We drop the CQ constraint by learning nonlinear interpolants using SVM classification (sampling-guessing-refining) :



⇒ M. Chen, J. Wang, J. An, D. Kapur, N. Zhan : *NIL : Learning nonlinear interpolants*. Under revision.

Termination of Polynomial Programs

The largest family whose termination is decidable

—Joint work with Y. Li, N. Zhan, H. Lu and G. Wu—

Program Termination

Termination Problem

Given a program P and an input x, to determine if P terminates with the input x.

Program Termination

Termination Problem

Given a program P and an input x, to determine if P terminates with the input x.

Example (Simple Loops)

$(x, y) := (x_0, y_0);$
while $(x + y = 0)\{$
  if? then $(x, y) := (y^2, 2x + y);$
  else $(x, y) := (2x^2 + y - 1, x + 2y + 1);\}$

int mccarthy(**int** n);
int $c \leftarrow 1;$
while $(c \neq 0 \wedge n \neq 91)\{$
  if $(n > 100)$
   then $n := n - 10; c := c - 1;$
   else $n := n + 11; c := c + 1;\}$
return n;

Positive and Negative Results

- ☹ Termination problem of programs is undecidable in general;
- ☹ Termination problem of general nonlinear programs is undecidable;
- ☹ Termination problem of general linear programs is undecidable;
- ☹ Even, termination problems of subclasses of linear or nonlinear programs are still undecidable.

Positive and Negative Results

- ☹ Termination problem of programs is undecidable in general;
- ☹ Termination problem of general nonlinear programs is undecidable;
- ☹ Termination problem of general linear programs is undecidable;
- ☹ Even, termination problems of subclasses of linear or nonlinear programs are still undecidable.

- ☺ Many sufficient conditions for termination and/or non-termination for linear and nonlinear programs;
- ☺ Termination or non-termination proofs can be synthesized using predicate abstraction for programs with complicated data structures;
- ☺ *Terminator* has been successfully applied in the termination analysis of drivers in Microsoft merchandised software product;
- ☺ The termination problem of some subclasses of linear programs have been proved decidable (e.g., [Tiwari, 2004]).

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while } \ (G(\boldsymbol{x}) = 0) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions.* arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while} \quad (G(\boldsymbol{x}) = 0) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions.* arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while } \left(G(\boldsymbol{x}) = 0\right) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :
   1. Construct the execution tree symbolically,

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions.* arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while}\ \ (G(\boldsymbol{x}) = 0)\quad \left\{\begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array}\right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :
   1. Construct the execution tree symbolically,
   2. Construct the set of n-nontermination execution paths, each of which forms a descending chain of algebraic sets,

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions*. arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while} \quad (G(\boldsymbol{x}) = 0) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :
   1. Construct the execution tree symbolically,
   2. Construct the set of *n*-nontermination execution paths, each of which forms a descending chain of algebraic sets,
   3. Identify a uniform bound on all these chains using Hilbert's function and Macaulay Theorem,

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions.* arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while } \ (G(\boldsymbol{x}) = 0) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :
   1. Construct the execution tree symbolically,
   2. Construct the set of n-nontermination execution paths, each of which forms a descending chain of algebraic sets,
   3. Identify a uniform bound on all these chains using Hilbert's function and Macaulay Theorem,
   4. The set of NTI corresponds exactly to the union of all these algebraic sets in these chains at the bound point.

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions*. arXiv.

Our Contributions

1. A class of nonlinear programs (MPPs) which is expressive enough, yet with a decidable termination problem :

$$\text{while} \ \ (G(\boldsymbol{x}) = 0) \quad \left\{ \begin{array}{ll} & \boldsymbol{x} := \boldsymbol{A}_1(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_2(\boldsymbol{x}); \\ & \vdots \\ \| & \boldsymbol{x} := \boldsymbol{A}_{l-1}(\boldsymbol{x}); \\ \| & \boldsymbol{x} := \boldsymbol{A}_l(\boldsymbol{x}); \end{array} \right\}$$

2. A decision procedure by computing the set of non-termination inputs (NTI) :
   1. Construct the execution tree symbolically,
   2. Construct the set of n-nontermination execution paths, each of which forms a descending chain of algebraic sets,
   3. Identify a uniform bound on all these chains using Hilbert's function and Macaulay Theorem,
   4. The set of NTI corresponds exactly to the union of all these algebraic sets in these chains at the bound point.
3. Generate all invariants of the program, under the template of polynomial equalities of a fixed degree.

⇒ Y. Li, N. Zhan, H. Lu, G. Wu : *Termination analysis of polynomial programs with equality conditions*. arXiv.

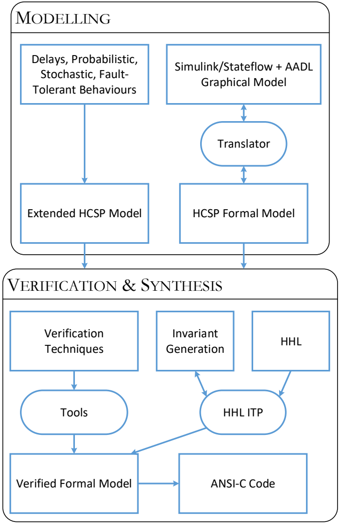
All in a Nutshell

Foundations of formal design of cyber-physical systems

—Joint work further with X. Han, T. Tang, S. Wang, M. Yang, A. P. Ravn, H. Zhao and L. Zou—

Outline

A Framework for Formal Design



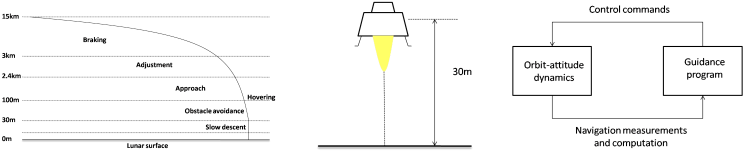
- Hierarchical modelling by Simulink/Stateflow + HCSP;
- Compositional reasoning based on Hybrid Hoare Logic (HHL);
- Substantial verification techniques incorporated :
  - Reachset computation;
  - Verification of delayed systems;
  - Interpolant synthesis;
  - Invariant generation.
- Refinement theory that generates code automatically from verified formal model.

⇛ M. Chen, A. P. Ravn, S. Wang, M. Yang, N. Zhan : *A two-way path between formal and informal design of embedded systems*. UTP'16.

⇛ M. Chen, X. Han, T. Tang, S. Wang, M. Yang, N. Zhan, H. Zhao, L. Zou : *MARS : A toolchain for modelling, analysis and verification of hybrid systems*. ProCoS'17.

Lunar Lander of Chang'e-3

- Mission description :



- Design objectives :
  - ⇒ $|v + 2| \leq 0.05$m/s during the slow descent phase and before touchdown;
  - ⇒ $|v| < 5$m/s at the time of touchdown.

Simulink Models

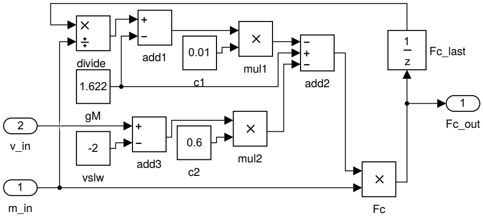


Figure – Simulink diagram of the guidance program for the slow descent phase

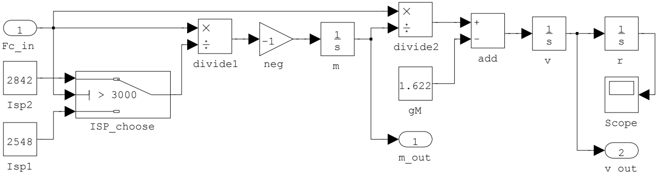


Figure – The Simulink diagram of the continuous dynamics for the slow descent phase

From Simulink to HCSP

$$
\begin{array}{rcl}
P & \widehat{=} & PC \,\|\, PD \\
PC & \widehat{=} & v := -2;\ m := 1250;\ r := 30; \\
 & & (\ \langle Sys_1 \& f > 3000\rangle \unrhd CommI; \\
 & & \langle Sys_2 \& f \le 3000\rangle \unrhd CommI\ )^* \\
PD & \widehat{=} & t := 0;\ g := 1.622;\ vslw := -2;\ f_1 = 2027.5; \\
 & & (\ ch_v?v_1;\ ch_m?m_1;\ f_1 := m_1 * aIC;\ ch_f!f_1; \\
 & & temp := t;\ \langle \dot{t} = 1 \& t < temp + 0.128\rangle\ )^* \\
aIC & \widehat{=} & g - 0.01 * (f_1 / m_1 - g) - 0.6 * (v_1 - vslw) \\
Sys_1 & \widehat{=} & \dot{m} = -f/2548,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
Sys_2 & \widehat{=} & \dot{m} = -f/2842,\ \dot{v} = f/m - 1.622,\ \dot{r} = v \\
CommI & \widehat{=} & ch_f?f \to \text{skip} \,[\!]\, ch_v!v \to \text{skip} \,[\!]\, ch_m!m \to \text{skip}
\end{array}
$$

From HCSP to Simulink

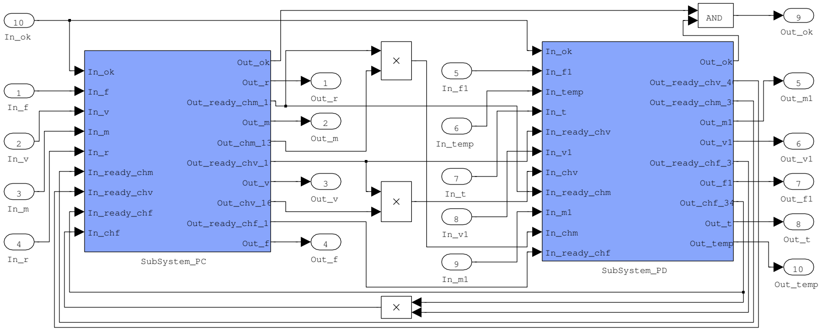


Figure – The top-level view of the translated Simulink model

Simulation Results

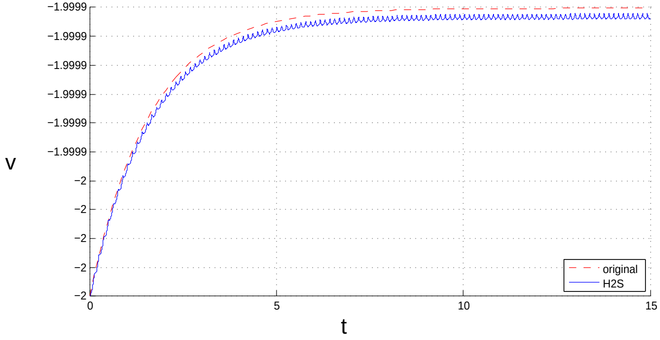


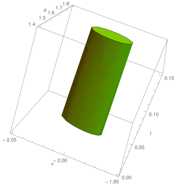
Figure – The evolution of velocity *v* in physical plant *PC*

Verification in HHL Prover

```
lemma cons1: "(t<=0.128) & (t>=0) & Inv |- |v-vlsw|<=0.05"
lemma cons2: "(v=-2) & (m=1250) & (Fc=2027.5)
  & (t=0) |- Inv"
lemma cons3: "(t= 0.128) & Inv
  |- substF([(t,0)], substF([(Fc,
     -0.01*(Fc-1.622*m) - 0.6*(v+2)*m + 1.622*m)],Inv))"
lemma cons4: "exeFlow(''v, m, r, t'',
  ''(Fc/m) - 1.622, -(Fc/2548), v, 1'',t < 0.128,Inv) |- Inv"
lemma cons5: "exeFlow(''v, m, r, t'',
  ''(Fc/m) - 1.622, -(Fc/2842), v, 1'',t < 0.128,Inv) |- Inv"
```

Verification in HHL Prover

```
lemma cons1: "(t<=0.128) & (t>=0) & Inv |- |v-vlsw|<=0.05"
lemma cons2: "(v=-2) & (m=1250) & (Fc=2027.5)
  & (t=0) |- Inv"
lemma cons3: "(t= 0.128) & Inv
  |- substF([(t,0)], substF([(Fc,
     -0.01*(Fc-1.622*m) - 0.6*(v+2)*m + 1.622*m)],Inv))"
lemma cons4: "exeFlow(''v, m, r, t'',
  ''(Fc/m) - 1.622, -(Fc/2548), v, 1'',t < 0.128,Inv) |- Inv"
lemma cons5: "exeFlow(''v, m, r, t'',
  ''(Fc/m) - 1.622, -(Fc/2842), v, 1'',t < 0.128,Inv) |- Inv"
```



Outline

Concluding Remarks

1. Decidability of Reachability for a Family of Differential Dynamics
   - Problem Formulation
   - Extension of the Decidable Fragment

2. Safety of Dynamical Systems under Time Delays
   - Why Time Delays
   - Verifying Delayed Differential Dynamics
   - Synthesizing Controllers Resilient to Delayed Interaction

3. Interpolation and Termination in the Context of Program Analysis
   - Synthesizing Interpolants for Nonlinear Arithmetic
   - Proving Termination of Polynomial Programs

4. A Framework for Modelling, Verification and Synthesis of Hybrid Systems
   - Overview of the Framework for Formal Design
   - Case Study on the Control Program of a Lunar Lander

Thank You — Q & A ?