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 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
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).
Hybrid Systems

Disturbances ("noise")

Environmental influence

Control

Selection

Active control law

Setpoints

D/A

A/D

Plant

Continuous controllers

Discrete supervisor

Load of continuous computations

Interleaved with discrete decisions

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).
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

1. Decidability of Reachability for a Family of Differential Dynamics
2. Safety of Dynamical Systems under Time Delays
3. Interpolation and Termination in the Context of Program Analysis
4. A Framework for Modelling, Verification and Synthesis of Hybrid Systems
5. Concluding Remarks
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

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

5. Concluding Remarks
   - Summary
Safety Verification Using Reachable Sets

- System is **safe**, if no trajectory enters the unsafe set.
LDSs with Inputs

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

\[
\dot{\xi} = A\xi + u,
\]

where \(\xi(t) \in \mathbb{R}^n\), \(A \in \mathbb{R}^{n \times n}\), and \(u : \mathbb{R} \to \mathbb{R}^n\) is a continuous function vector which is called the *input*. 
Linear dynamical systems (LDSs) with inputs are differential equations of the form

\[ \dot{x} = Ax + u, \]

where \( x(t) \in \mathbb{R}^n \), \( A \in \mathbb{R}^{n \times n} \), and \( u : \mathbb{R} \to \mathbb{R}^n \) is a continuous function vector which is called the input. The forward reachable set:

\[ \text{Post}(X) := \{ y \in \mathbb{R}^n \mid \exists x \exists t : x \in X \land t \geq 0 \land \Phi(x, t) = y \} \]
**LDSs with Inputs**

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

\[ \dot{x} = Ax + u, \]

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

- The forward reachable set:

\[ \text{Post}(X) := \{ y \in \mathbb{R}^n | \exists x \exists t : x \in X \land t \geq 0 \land \Phi(x, t) = y \} \]

- Reachability problem:

\[ \mathcal{F}(X, Y) := Y \cap \text{Post}(X) = \emptyset \]
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 $u$ is a polynomial;

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

3. **A is diagonalizable with purely imaginary eigenvalues**, and each component of $u$ of the form $\sum_{i=1}^{m} c_i \sin(\lambda_i t) + d_i \cos(\lambda_i t)$, where $\lambda_i$s are **rationals** and $c_i$s and $d_i$s are subject to semi-algebraic constraints.
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 \( u \) is of the form \( \sum_{i=1}^{m} c_i e^{\lambda_i t} \), where \( \lambda_i \)'s are **rationals reals** and \( c_i \)'s are subject to semi-algebraic constraints;

\[ \Rightarrow \quad \text{T. Gan, M. Chen, L. Dai, B. Xia, N. Zhan : Decidabil. of the reachabil. for a family of linear vector fields. ATVA’15.} \]

3. \( A \) is **diagonalizable** with **purely imaginary** eigenvalues, and each component of \( u \) of the form \( \sum_{i=1}^{m} c_i \sin(\lambda_i t) + d_i \cos(\lambda_i t) \), where \( \lambda_i \)'s are **rationals reals** and \( c_i \)'s and \( d_i \)'s are subject to semi-algebraic constraints.

\[ \Rightarrow \quad \text{T. Gan, M. Chen, Y. Li, B. Xia, N. Zhan : Computing reachable sets of linear vector fields revisited. ECC’16.} \]
A nonlinear differential dynamic

\[ \dot{\xi} = F(\xi, u) \]

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:

\[
\dot{\xi} = \begin{bmatrix} \dot{\zeta}_1 \\ \dot{\zeta}_2 \\ \vdots \\ \dot{\zeta}_m \end{bmatrix} = \begin{bmatrix} A_1 \zeta_1 + u_1(t) \\ A_2 \zeta_2 + u_2(t, \zeta_1) \\ \vdots \\ A_m \zeta_m + u_m(t, \zeta_1, \ldots, \zeta_{m-1}) \end{bmatrix},
\]

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, \( u_1, \ldots, u_m \) are polynomial-exponential-trigonometric functions.
Solvable Systems

A nonlinear differential dynamic

\[ \dot{\xi} = F(\xi, u) \]

is called \textbf{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:

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

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, \( u_1, \ldots, u_m \) are \textit{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 \( u_j \) is a polynomial;
  2. Each \( A_i \) is diagonalizable with real eigenvalues, and each component of \( u_j \) is of the form \( \sum_{j=1}^{m_i} c_{ij} e^{\lambda_{ij} t} \), where \( \lambda_{ij} \)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 \( u_i \) of the form \( \sum_{j=1}^{m_i} c_{ij} \sin(\lambda_{ij} t) + d_{ij} \cos(\lambda_{ij} t) \), where \( \lambda_{ij} \)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.

Outline

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

5. Concluding Remarks
   - Summary
Advice by a Wise Man

Indecision and delays are the parents of failure.

(George Canning)
Advice by a Wise Man

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?
Advice by a Wise Man

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!
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.
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 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.
Q2: Do Delays Have Observable Effect?

\[
\begin{align*}
\dot{x}(t) &= -x(t) \\
{x(0)} &= 1
\end{align*}
\]

\[
\begin{align*}
\dot{x}(t) &= -x(t-1) \\
x([-1, 0]) &\equiv 1
\end{align*}
\]
Q2: Do Delays Have Observable Effect?  – Yes, they have.

\[
\begin{align*}
\dot{x}(t) &= -x(t) \\
{x(0)} &= 1
\end{align*}
\]
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)
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)
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]
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)

\[
\begin{cases}
\dot{x}(t) = f(x(t), x(t - r_1), \ldots, x(t - r_k)), & t \in [0, \infty) \\
\quad x(t) = x_0 \in \Theta, & t \in [-r_{\text{max}}, 0]
\end{cases}
\]

The unique solution (trajectory): \[\xi_{x_0}(t) : [-r_{\text{max}}, \infty) \mapsto \mathbb{R}^n.\]
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.
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.

Try only if infinite state no longer is scary enough to you!
Method 1: Simulation-Based Verification

Figure – A finite $\epsilon$-cover of the initial set of states.

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

©A. Donzé & O. Maler, 2007
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.

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.

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.

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.

Discrete Dynamics

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

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

\[ \sum_r = \{ RU, UR, LU, UL, RD, DR, LD, DL, \epsilon \}, \]
\[ \sum_k = \{ R, L, U, D \}. \]
A Robot-Escaping Game

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

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

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

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

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

$\Sigma_k = \{R, L, U, D\}$.
A Robot-Escaping Game

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

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

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:
A Robot-Escaping Game

![Diagram of a 4x4 room with a robot and an obstacle, illustrating the game's dynamics.](attachment:image.png)

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

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

**Figure** – A robot escape game in a 4x4 room, with

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

\[ \Sigma_k = \{ R, L, U, D \} \]
A Robot-Escaping Game

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 × 4 room, with
\[ \Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \]
\[ \Sigma_k = \{R, L, U, D\}. \]
A Robot-Escaping Game

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

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.

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

3 steps delay:
A Robot-Escaping Game

**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
\[
\Sigma_r = \{RU, UR, LU, UL, RD, DR, LD, DL, \epsilon\}, \\
\Sigma_k = \{R, L, U, D\}.
\]
A Robot-Escaping Game

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.
Observation: It doesn’t make an observable difference for the joint dynamics whether delay occurs in perception, actuation, or both.
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 with delay can be solved algorithmically. The game graph incurs blow-up by a factor of $|\text{Alphabet(ego)}| \cdot \text{delay}$. 
Safety games w. delay can be solved algorithmically.

Game graph incurs blow-up by factor $|\text{Alphabet(ego)}|^\text{delay}$.
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].
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].
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].
**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].
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:

- Maximum delay 5
- Out of order!

Sample #

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

😊 Observations may arrive out-of-order:

![Diagram showing out-of-order observations](image)

- Maximum delay 5
- Out of order!

- Sample #
  - 56
  - 57
  - 58
  - 61
  - 60
  - 59
  - 62

- Time
  - t
  - 59
  - 60
  - 61
  - 62
  - 63
  - 64
  - 65

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

![Diagram showing effective delay reduction](image)

- Maximum delay 5
- Factual delay 3
- Effective delay 2

More recent state information available earlier

More stochastically expected controllability even better than for strict delay $k$. 

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

© Mingshuai Chen · Institute of Software, CAS
© Modelling · Verification · Synthesis
© RWTH Aachen Univ. · 2018
How about Non-Order-Preserving Delays?

💧 Observations may arrive out-of-order:

![Diagram showing out-of-order observations with maximum delay 5](image)

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

![Diagram showing reduced effective delay with maximum delay 5](image)

💧 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. 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

5. Concluding Remarks
   - Summary
Interpolation over Nonlinear Arithmetic

The cornerstone of ATP, SMT, BMC, etc.

Craig Interpolant

Given $\phi$ and $\psi$ in a theory $T$ s.t. $\phi \land \psi \models_T \bot$, a formula $I$ is a (reverse) interpolant of $\phi$ and $\psi$ if

1. $\models_T I$;
2. $I \land \psi \models_T \bot$; and
3. $\text{var}(I) \subseteq \text{var}(\phi) \cap \text{var}(\psi)$. 

Craig Interpolation
Craig Interpolation

Craig Interpolant

Given $\phi$ and $\psi$ in a theory $\mathcal{T}$ s.t. $\phi \land \psi \models \mathcal{T} \bot$, a formula $I$ is a (reverse) interpolant of $\phi$ and $\psi$ if

1. $\phi \models \mathcal{T} I$;
2. $I \land \psi \models \mathcal{T} \bot$; and
3. $\text{var}(I) \subseteq \text{var}(\phi) \cap \text{var}(\psi)$.
Craig Interpolation

Craig Interpolant

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

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

- **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 $\phi$ and $\psi$ in a theory $T$ s.t. $\phi \land \psi \models_T \bot$, a formula $I$ is a (reverse) interpolant of $\phi$ and $\psi$ if

1. $\phi \models_T I$;
2. $I \land \psi \models_T \bot$; and
3. $\text{var}(I) \subseteq \text{var}(\phi) \cap \text{var}(\psi)$.

- **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:

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;

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.

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);

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 in 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/

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

Our Contributions (Cont’d)

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

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

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 $x$. 
Termination Problem

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

Example (Simple Loops)

\[
\begin{align*}
(x, y) &:= (x_0, y_0); \\
\text{while} \ (x + y = 0) \{ \\
\quad \text{if? then} \ (x, y) &:= (y^2, 2x + y); \\
\quad \text{else} \ (x, y) &:= (2x^2 + y - 1, x + 2y + 1); \\
\} \\
\text{int} \ mccarthy(\text{int} \ n); \\
\text{int} \ c &\leftarrow 1; \\
\text{while} \ (c \neq 0 \land n \neq 91) \{ \\
\quad \text{if} \ (n > 100) \\
\quad \quad \text{then} \ n &:= n - 10; \ c := c - 1; \\
\quad \text{else} \ n &:= n + 11; \ c := c + 1; \\
\} \\
\text{return} \ n;
\end{align*}
\]
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:

   \[
   \begin{align*}
   x & := A_1(x); \\
   \| & x := A_2(x); \\
   \| & \vdots \\
   \| & x := A_{l-1}(x); \\
   \| & x := A_l(x);
   \end{align*}
   \]

   while \( G(x) = 0 \)

Our Contributions

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

   \[
   \begin{align*}
   &\text{while } (G(x) = 0) \quad \begin{cases} 
   x := A_1(x); \\
   \| \\
   x := A_2(x); \\
   \| \\
   \vdots \\
   \| \\
   x := A_{l-1}(x); \\
   \| \\
   x := A_l(x);
   \end{cases}
   \end{align*}
   \]

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

   \[\text{Construct the execution tree symbolically,} \]
   \[\text{Construct the set of } n\text{-nontermination execution paths, each of which forms a descending chain of algebraic sets,} \]
   \[\text{Identify a uniform bound on all these chains using Hilbert's function and Macaulay Theorem,} \]
   \[\text{The set of NTI corresponds exactly to the union of all these algebraic sets in these chains at the bound point.} \]

   \[\text{Generate all invariants of the program, under the template of polynomial equalities of a fixed degree.} \]

Our Contributions

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

   \[
   \begin{align*}
   &x := A_1(x); \\
   \| &x := A_2(x); \\
   \| &\vdots \\
   \| &x := A_{l-1}(x); \\
   \| &x := A_l(x); \\
   \end{align*}
   \]

   while \( G(x) = 0 \)

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

   1. Construct the execution tree symbolically,

Our Contributions

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

\[
\begin{align*}
\forall x & := A_1(x); \\
\| & x := A_2(x); \\
\| & \quad \vdots \\
\| & x := A_{l-1}(x); \\
\| & x := A_l(x);
\end{align*}
\]

while \( G(x) = 0 \)

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,

\[ Y. \text{Li, N. Zhan, H. Lu, G. Wu: } \text{Termination analysis of polynomial programs with equality conditions. } \text{arXiv.} \]
Our Contributions

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

   \[
   \begin{align*}
   &x := A_1(x); \\
   &\|
   &x := A_2(x); \\
   &\|
   &\vdots \\
   &\|
   &x := A_{l-1}(x); \\
   &\|
   &x := A_l(x);
   \end{align*}
   \]

   while \((G(x) = 0)\)

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,

Our Contributions

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

\[
\begin{align*}
\text{while } (G(x) = 0) \\
\{ \\
\quad x &:= A_1(x); \\
\quad \| \quad x &:= A_2(x); \\
\quad \| \quad \vdots \\
\quad \| \quad x &:= A_{l-1}(x); \\
\quad \| \quad x &:= A_l(x); \\
\} 
\end{align*}
\]

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.

Our Contributions

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

   \[
   \begin{align*}
   & x := A_1(x); \\
   \| & x := A_2(x); \\
   \vdots & \\
   \| & x := A_{l-1}(x); \\
   \| & x := A_l(x);
   \end{align*}
   \]

   while \( G(x) = 0 \)

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.

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

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

5. Concluding Remarks
   - Summary
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.


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

\[ P \models PC \parallel PD \]

\[ PC \models v := -2; m := 1250; r := 30; \]
\[ (\langle Sys_1 & f \rangle > 3000) \models CommI; \]
\[ (\langle Sys_2 & f \rangle \leq 3000) \models CommI \] *

\[ PD \models t := 0; g := 1.622; vslw := -2; f_1 := 2027.5; \]
\[ (ch_v?v_1; \; ch_m?m_1; \; f_1 := m_1 \times aIC; \; ch_f!f_1; \]
\[ temp := t; \langle \dot{t} = 1 \& t < temp + 0.128 \rangle ) * \]

\[ aIC \models g - 0.01 \times (f_1 / m_1 - g) - 0.6 \times (v_1 - vslw) \]

\[ Sys_1 \models \dot{m} = -f / 2548, \dot{v} = f / m - 1.622, \dot{r} = v \]

\[ Sys_2 \models \dot{m} = -f / 2842, \dot{v} = f / m - 1.622, \dot{r} = v \]

\[ CommI \models ch_f?f \rightarrow skip \; \parallel \; ch_v?v \rightarrow skip \; \parallel \; ch_m!m \rightarrow skip \]
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

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

5. Concluding Remarks
   - Summary
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?