Rigorous Continuous Evolution of Uncertain Systems*

Luca Geretti[1], Sanja Zivanovic Gonzalez[2], Pieter Collins[3], Davide Bresolin[4], Tiziano Villa[1]

[1]University of Verona, Italy [2]Barry University, Miami (FL), USA

[3]University of Maastricht, The Netherlands [4]University of Padova, Italy

(in)Formal Methods Meeting, Udine, July 5th 2019

*To appear in NSV19

Outline

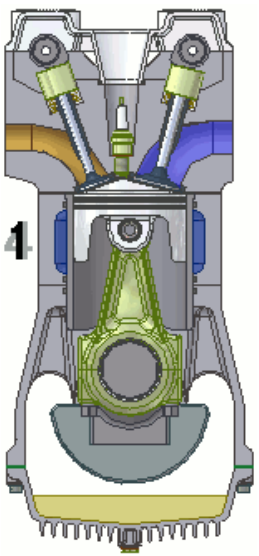
Hybrid systems

Many real systems have a double nature:

- they evolve in a continuous fashion
- they are controlled by a discrete system

They are called hybrid systems and interact with the physical world via sensors and actuators, usually with feedback loops where physics affects computation and vice versa

Example: 4-strokes engine



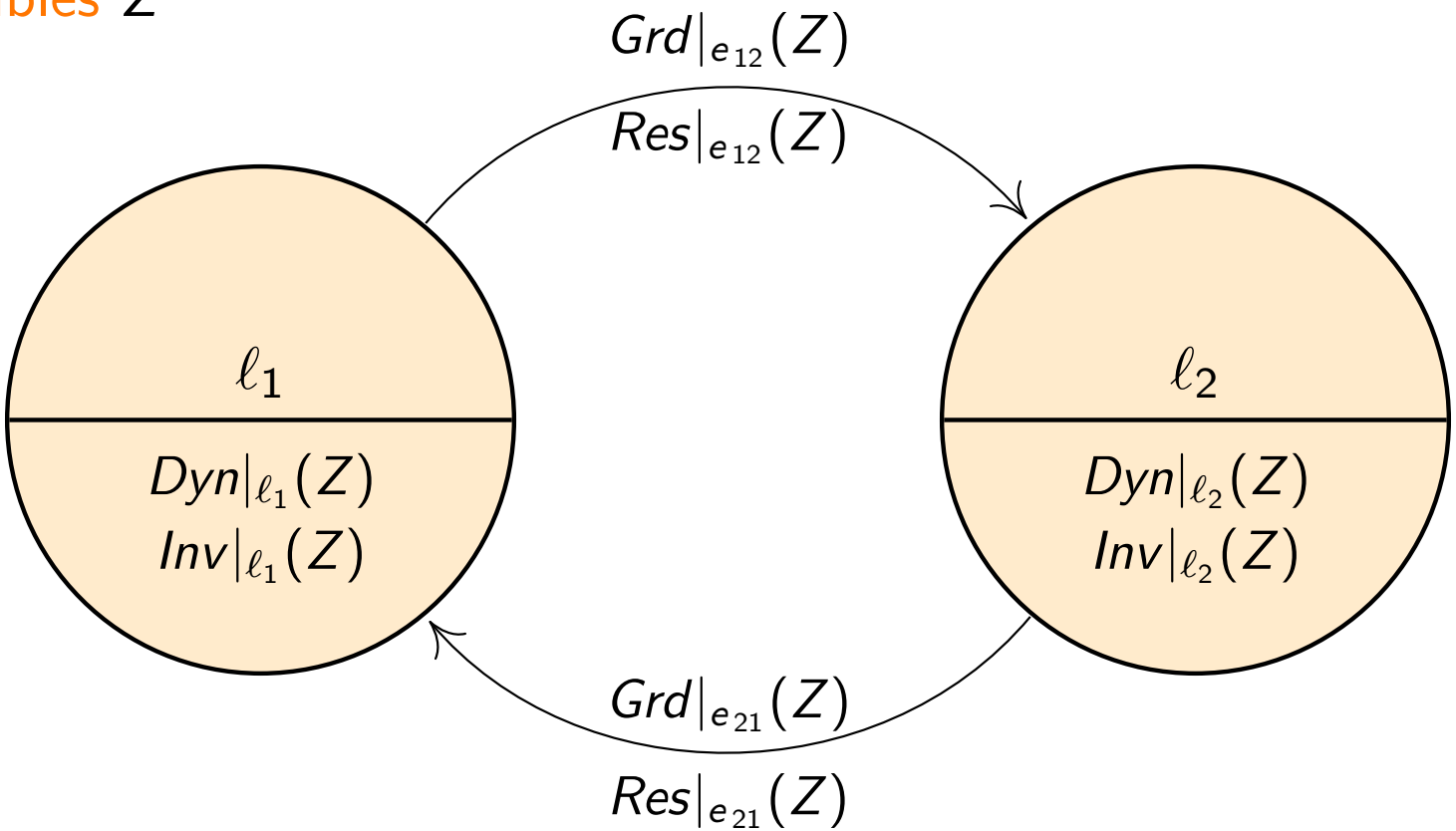
- Intake stroke: air and vaporized fuel are drawn in
- Compression stroke: fuel vapor and air are compressed and ignited
- Combustion stroke: fuel combusts and piston is pushed downwards
- Exhaust/Emission stroke: exhaust is driven out
- During 1st, 2nd and 4th stroke the piston is relying on the power and momentum generated by the pistons of the other cylinders

During the 4 strokes pression, temperature, . . . , vary continuously

Hybrid automata

The intuition

A hybrid automaton H is a finite-state automaton with continuous variables Z



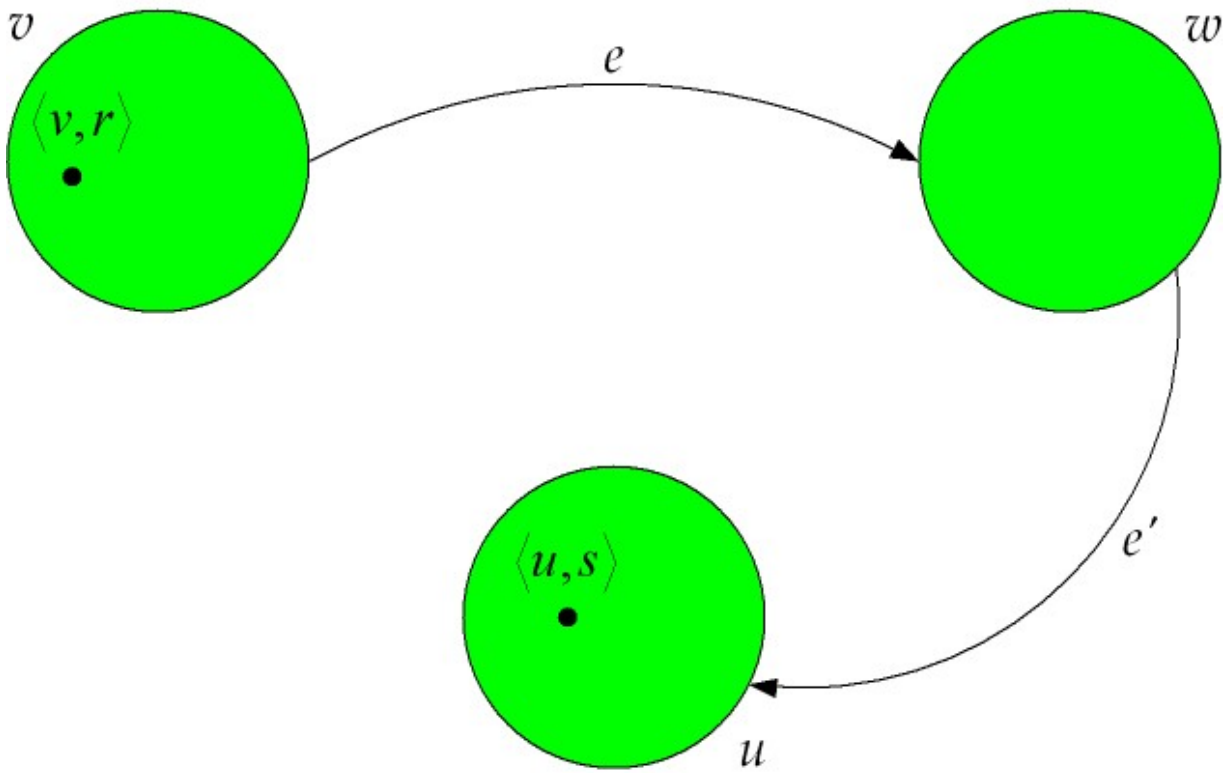
A state is a couple $\langle \ell, r \rangle$ where r is a valuation for Z

Hybrid automata

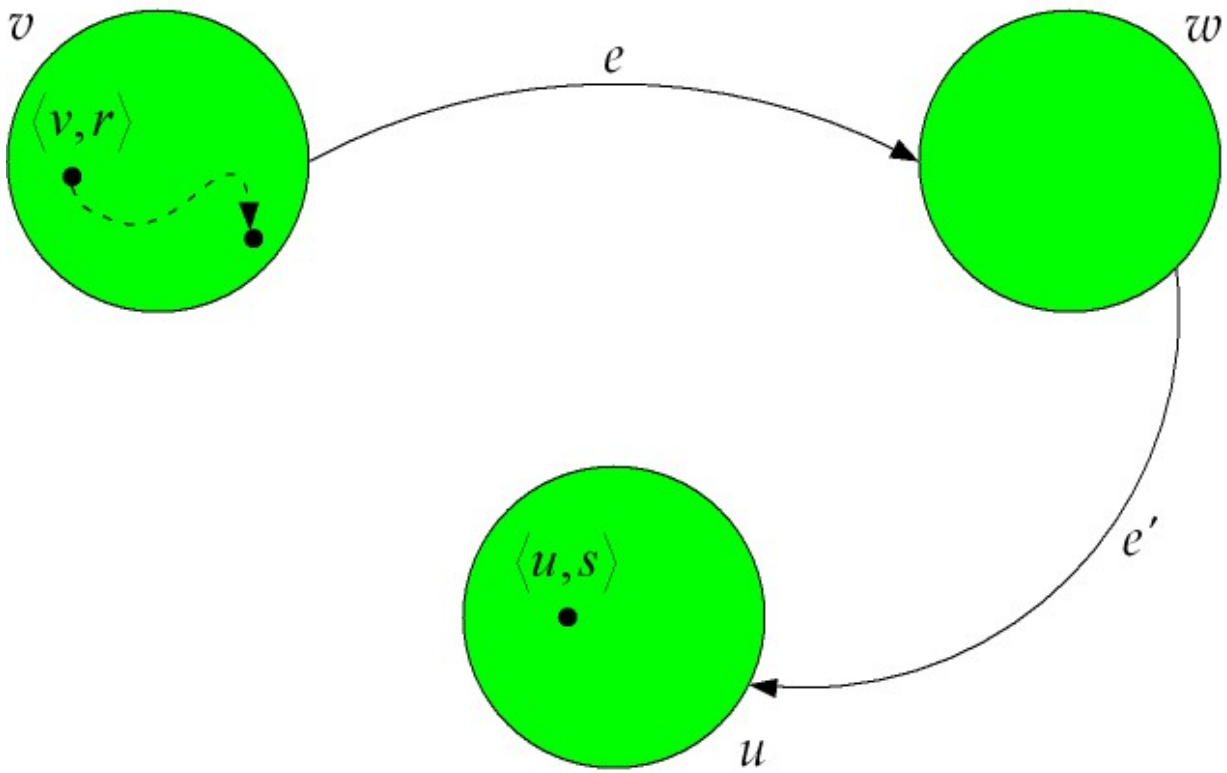
Functional representation

- dynamics $Dyn|_\ell$: evolution of the variables in location ℓ
- invariant $Inv|_\ell$: conditions under which continuous evolution is allowed in location ℓ
- guard $Gua|_e$: conditions under which discrete evolution is allowed according to event e
- reset $Res|_e$: transformation of the continuous state after event e

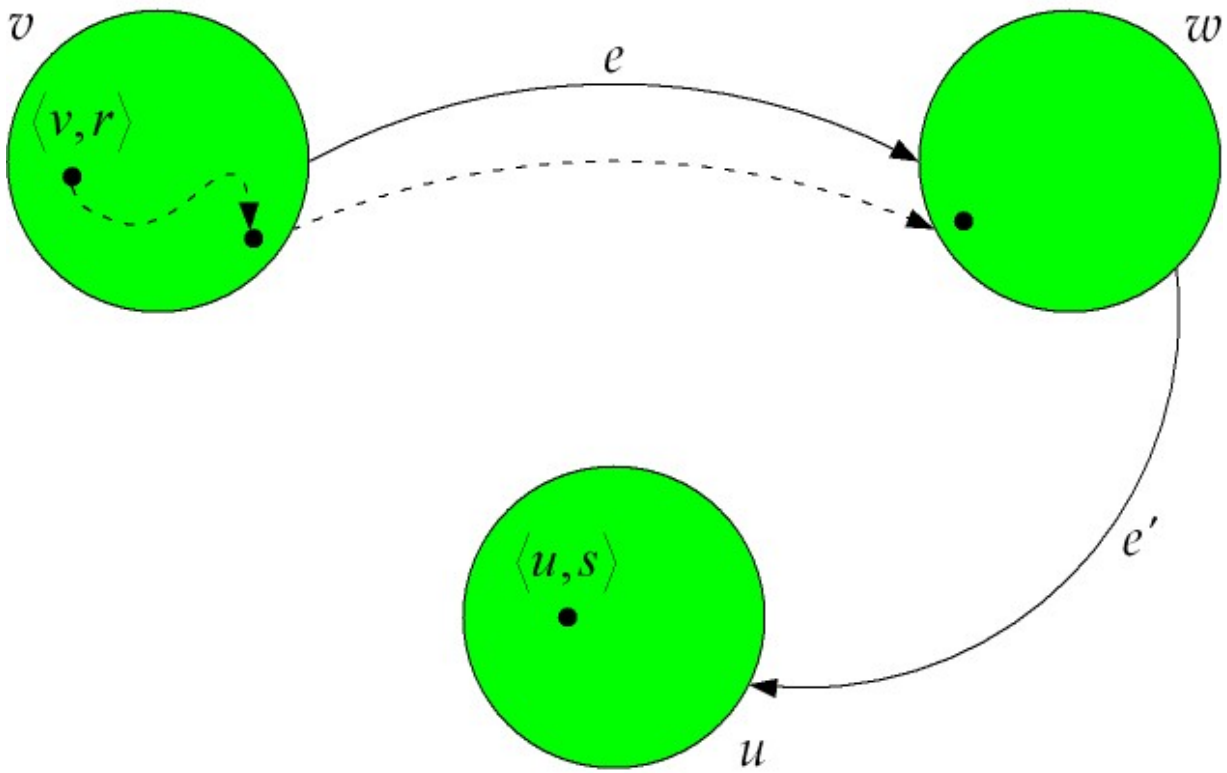
Reachability



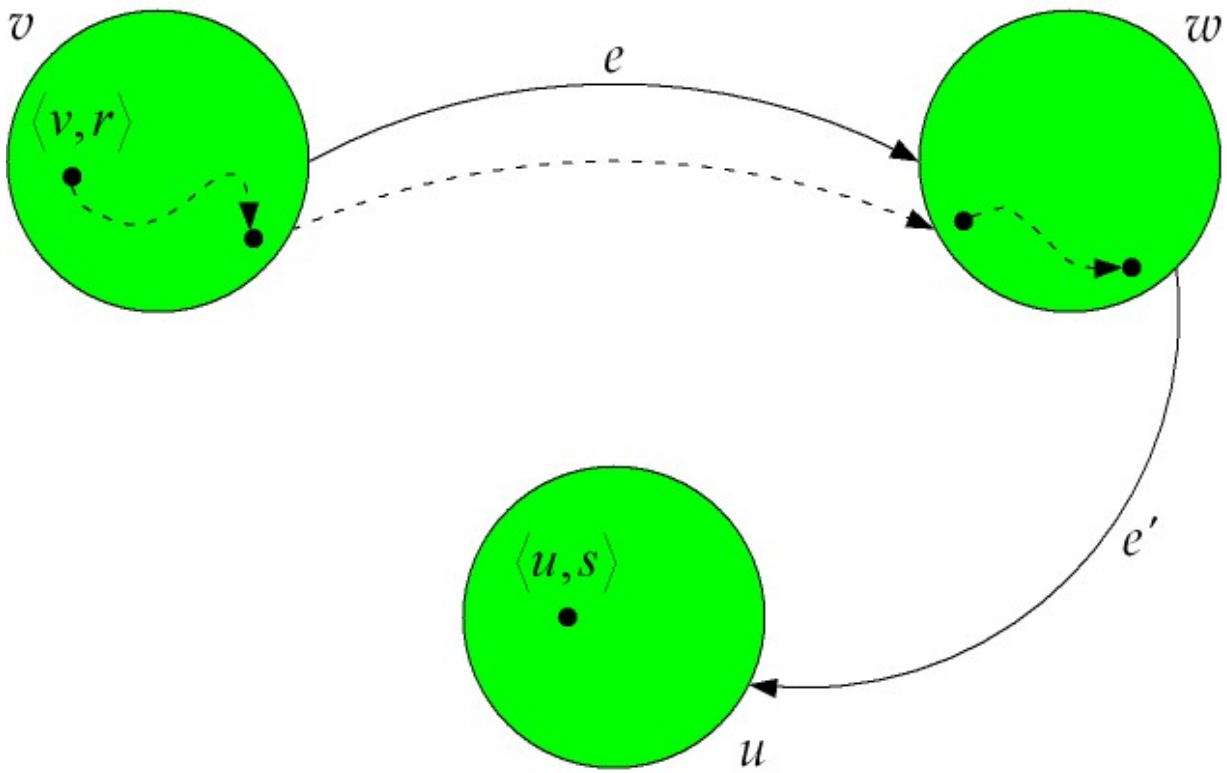
Reachability



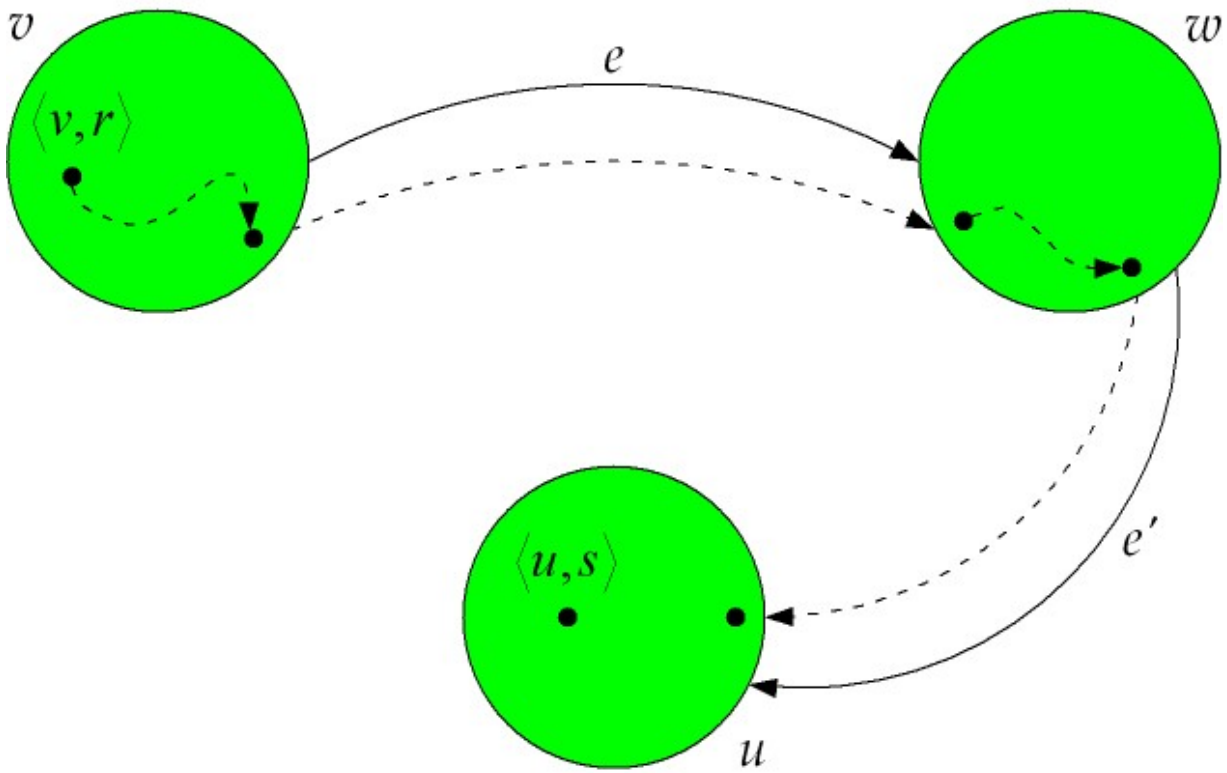
Reachability



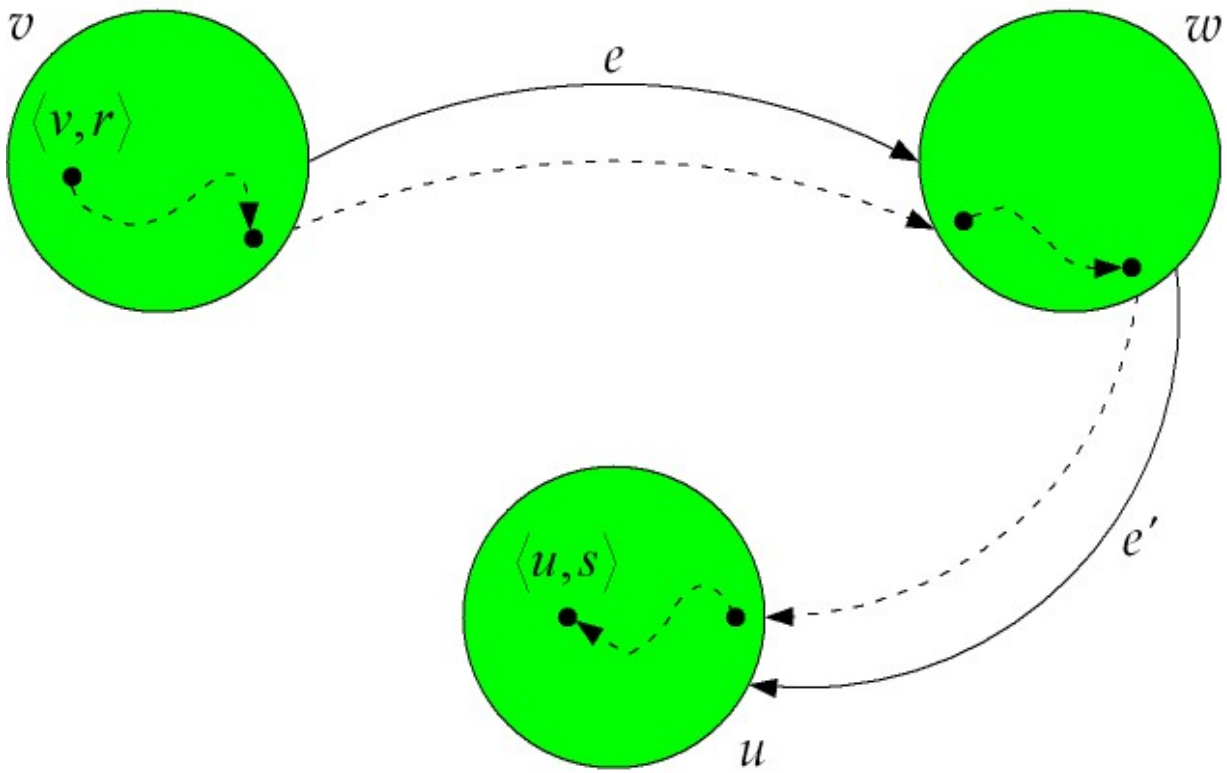
Reachability



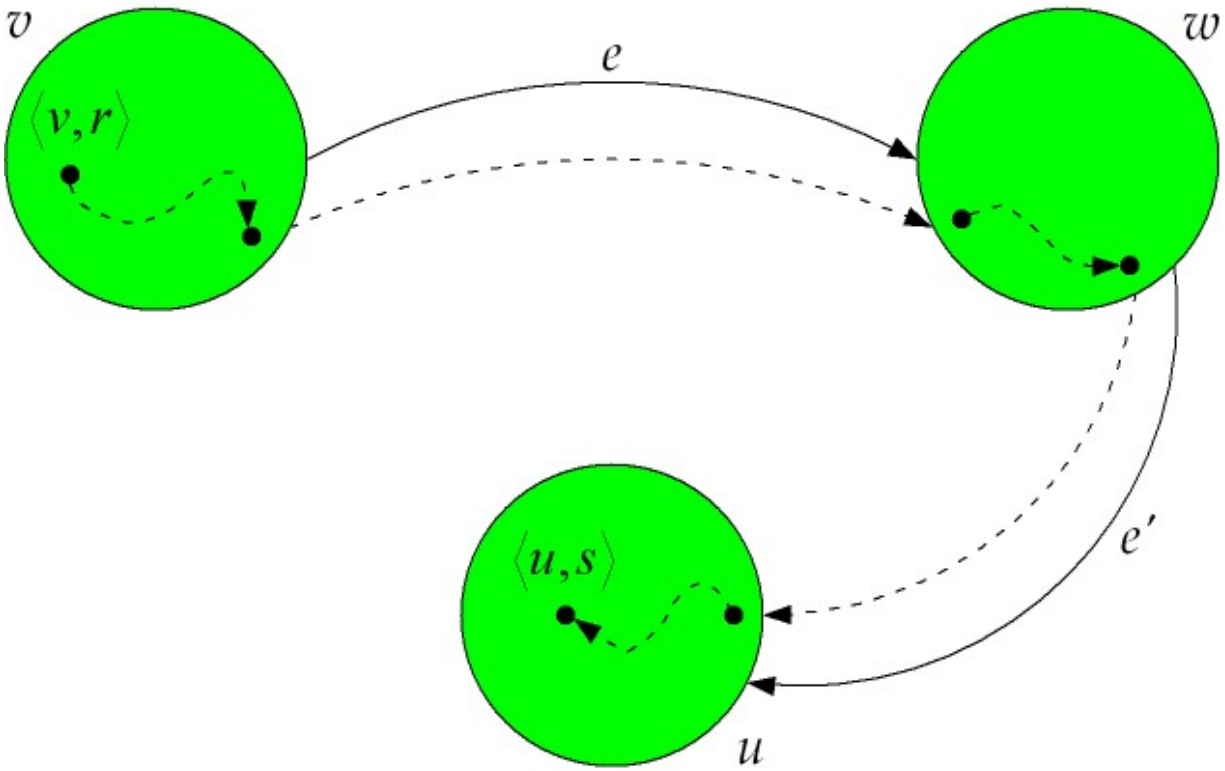
Reachability



Reachability



Reachability



Let $s, r \in \mathbb{R}^k$. Can we reach $\langle u, s \rangle$ from $\langle v, r \rangle$?

Reachability and Verification

Reachability can be used to verify safety properties

φ is always true in H

if and only if

all states reachable from the initial states of H

are included in the safe set $Sat(\varphi)$.

Reachability and Verification

Reachability can be used to verify safety properties

φ is always true in H

if and only if

all states reachable from the initial states of H

are included in the safe set $Sat(\varphi)$.

Question

Is the reachability problem for Hybrid Automata decidable?

Reachability and Verification

Reachability can be used to verify safety properties

φ is always true in H
if and only if
all states reachable from the initial states of H
are included in the safe set $Sat(\varphi)$.

Question

Is the reachability problem for Hybrid Automata decidable?

Answer

No (Alur et al. 1995).

Many different approaches

Different approaches have been developed to solve the verification problem, at least in some cases:

1. Restrict to special classes of Hybrid Automata that admit exact verification
   - UPPAAL, HYTECH
2. Compute approximations of the reachable set
   - PHAVER, SPACEEX, COHO, FLOW*, HYPRO/HYDRA, SAPO, ARIADNE
3. Compute discrete abstractions of the system
   - HSOLVER, HYBRIDSAL, HYCOMP, DREACH/DREAL
4. Simulate the system to find counterexample
   - BREACH, S-TALIRO, C2E2, HYLAA
5. Use automated theorem proving techinques
   - KEYMAERA X, ISABELLE/HOL

What is Ariadne?

- A software framework for rigorous numerical computations on functions and sets.
- The result of a collaboration between several universities, mainly Maastricht University and the University of Verona.
- Specialised on reachability analysis of continuous and hybrid systems.
- Implemented as a C++ library, with an optional Python interface.
- Released as an open source distribution:
  `http://www.ariadne-cps.org`

The main motivations for Ariadne

1. To be formally sound in all numeric, logic and geometric operations
   - Most tools lack a formal framework or settle for inaccuracies in order to increase their efficiency

The main motivations for Ariadne

1. To be formally sound in all numeric, logic and geometric operations
   - Most tools lack a formal framework or settle for inaccuracies in order to increase their efficiency
2. To properly deal with nonlinear functions in the dynamics and the discrete transitions
   - Many tools are limited to the analysis of affine systems

The main motivations for Ariadne

1. To be formally sound in all numeric, logic and geometric operations
   - Most tools lack a formal framework or settle for inaccuracies in order to increase their efficiency
2. To properly deal with nonlinear functions in the dynamics and the discrete transitions
   - Many tools are limited to the analysis of affine systems
3. To use distinct semantics allowing to compute set approximations from above or from below, which in turn enable both verification and falsification of properties
   - Most tools have only verification or only falsification capabilities

Outline

Decomposition of nonlinear systems

Motivation

Validated reachability for nonlinear systems can be effectively computed for a small number of coupled variables.

Decomposition of nonlinear systems

Motivation

Validated reachability for nonlinear systems can be effectively computed for a small number of coupled variables.

Approach

Instead of analysing a full system, we decompose it into subsystems where we replace the coupling variables with proper time-varying inputs.

- We settle on partial information over the dynamics of some variables
- Equivalently, inputs represent noise sources with bounded values

Contract-based Design

A formal approach used to handle complex systems:

- The system is specified as a set of components
- Each component is annotated with assumptions and guarantees that represent a contract
  - An assumption can be seen as a set of conditions on the inputs to the component (i.e., the set of allowed inputs)
  - A guarantee can be seen as a set of required outputs by the component (i.e., the set of expected outputs)
- Contracts can be composed/decomposed to handle complex hierarchical systems
  - E.g., a contract on the full system can be decomposed into a set of simpler contracts for each component that, if satisfied separately, guarantee that the original contract is satisfied
- Also referred to as assume-guarantee reasoning

Continuous vs hybrid decomposition

Continuous case

We decompose the dynamics on a specific location of the composed hybrid system.

- We can handle a high-dimensional system in a flexible way, where optimality depends on the decomposition criterion.

Continuous vs hybrid decomposition

Continuous case

We decompose the dynamics on a specific location of the composed hybrid system.

- We can handle a high-dimensional system in a flexible way, where optimality depends on the decomposition criterion.

Hybrid case

We directly decompose the hybrid system, according to the defined components.

- We can use a contract-based approach, to implement assume-guarantee verification of the components of interest.

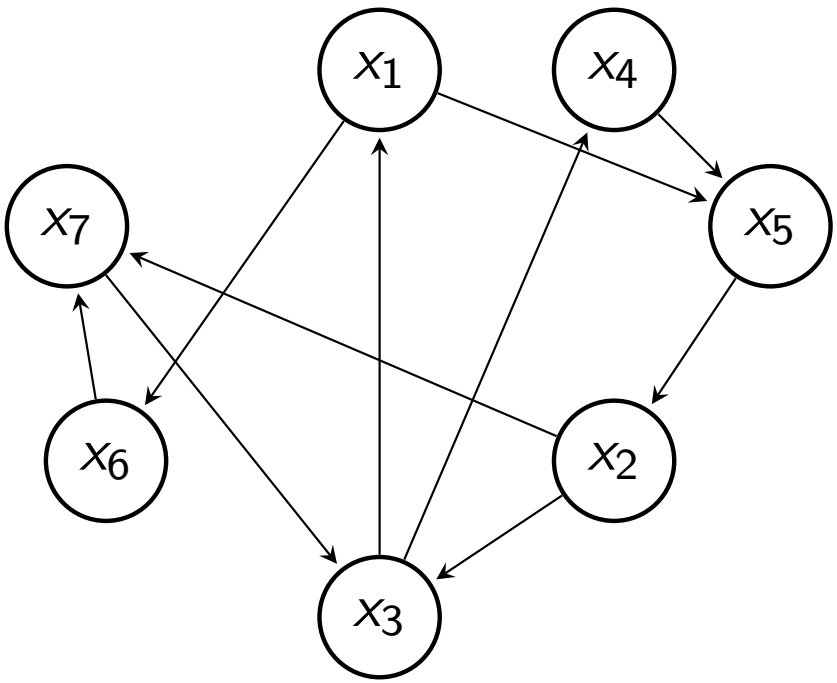
Decomposition in the continuous space

System definition

Source: *M. T. Laub and W. F. Loomis*. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. Molecular Biology of the Cell, 9:3521–3532, 1998.

$$\begin{cases} \dot{x}_1 = 1.4x_3 - 0.9x_1 \\ \dot{x}_2 = 2.5x_5 - 1.5x_2 \\ \dot{x}_3 = 0.6x_7 - 0.8x_2x_3 \\ \dot{x}_4 = 2 - 1.3x_3x_4 \\ \dot{x}_5 = 0.7x_1 - x_4x_5 \\ \dot{x}_6 = 0.3x_1 - 3.1x_6 \\ \dot{x}_7 = 1.8x_6 - 1.5x_2x_7 \end{cases}$$

Dynamics



Static dependency graph

Decomposition in the continuous space

A partially distributed approach

On each continuous integration step k:

1. Compute the bounding box of the flow B_k
   - Gives the bounds for an input that replaces a variable
   - May not converge

Decomposition in the continuous space

A partially distributed approach

On each continuous integration step k:

1. Compute the bounding box of the flow B_k
   - Gives the bounds for an input that replaces a variable
   - May not converge
2. Choose a decomposition of the variables into Q disjoint sets that minimises a chosen cost function
   - Static decomposition, valid $\forall k$: minimise the number of arcs removed from the dependency graph
   - Dynamic decomposition: weight each arc with the impact on the dynamics in terms of range width (using B_k), minimize the sum of the weights

Decomposition in the continuous space

A partially distributed approach

On each continuous integration step k:

1. Compute the bounding box of the flow B_k
   - Gives the bounds for an input that replaces a variable
   - May not converge
2. Choose a decomposition of the variables into Q disjoint sets that minimises a chosen cost function
   - Static decomposition, valid $\forall k$: minimise the number of arcs removed from the dependency graph
   - Dynamic decomposition: weight each arc with the impact on the dynamics in terms of range width (using B_k), minimize the sum of the weights
3. For each set of variables:
   - 3.1 Replace the external variables with inputs
   - 3.2 Compute the flow function

Decomposition in the continuous space

A partially distributed approach

On each continuous integration step k:

1. Compute the bounding box of the flow B_k
   - Gives the bounds for an input that replaces a variable
   - May not converge
2. Choose a decomposition of the variables into Q disjoint sets that minimises a chosen cost function
   - Static decomposition, valid $\forall k$: minimise the number of arcs removed from the dependency graph
   - Dynamic decomposition: weight each arc with the impact on the dynamics in terms of range width (using B_k), minimize the sum of the weights
3. For each set of variables:
   - 3.1 Replace the external variables with inputs
   - 3.2 Compute the flow function
4. Combine the flow functions

Decomposition in the continuous space

A partially distributed approach

On each continuous integration step k:

1. Compute the bounding box of the flow B_k
   - Gives the bounds for an input that replaces a variable
   - May not converge
2. Choose a decomposition of the variables into Q disjoint sets that minimises a chosen cost function
   - Static decomposition, valid $\forall k$: minimise the number of arcs removed from the dependency graph
   - Dynamic decomposition: weight each arc with the impact on the dynamics in terms of range width (using B_k), minimize the sum of the weights
3. For each set of variables:
   - 3.1 Replace the external variables with inputs
   - 3.2 Compute the flow function
4. Combine the flow functions
5. Evaluate the set at the end of the continuous step

Decomposition in the continuous space

For our system

Static decomposition suggests $\{x_1, x_4, x_5\}$ and $\{x_2, x_3, x_6, x_7\}$ sets, yielding

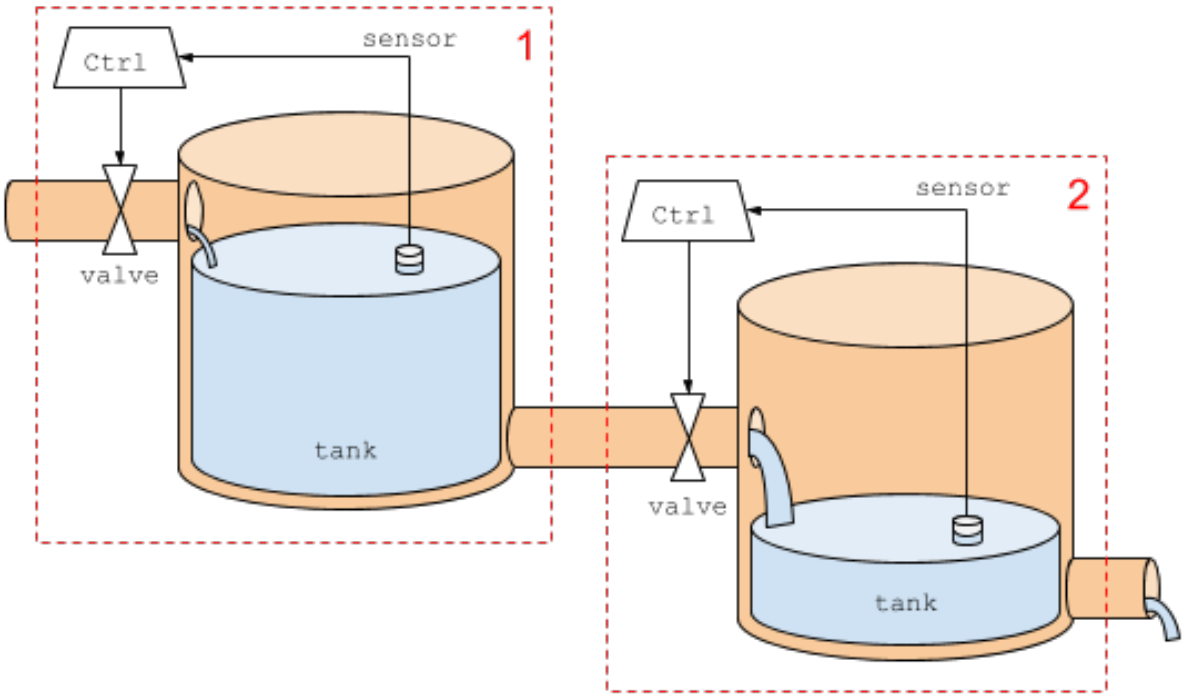
$$C_1 : \begin{cases} \dot{x}_1 = 1.4v_3 - 0.9x_1 \\ \dot{x}_4 = 2 - 1.3v_3x_4 \\ \dot{x}_5 = 0.7x_1 - x_4x_5 \end{cases} \quad C_2 : \begin{cases} \dot{x}_2 = 2.5v_5 - 1.5x_2 \\ \dot{x}_3 = 0.6x_7 - 0.8x_2x_3 \\ \dot{x}_6 = 0.3v_1 - 3.1x_6 \\ \dot{x}_7 = 1.8x_6 - 1.5x_2x_7 \end{cases}$$

Hence we have $\{n = 3, m = 1\}$ and $\{n = 4, m = 2\}$ components.

The resulting solution will be coarser but faster to calculate.

Decomposition in the hybrid space

An open loop system



Objective

Keep the water level x in each tank within a required range, using a valve aperture control u.

a. The first component C_1 can be analyzed in isolation

b. The second component C_2 can be analyzed in isolation if we replace its dependency from C_1 with a time-varying input.

Decomposition in the hybrid space

Model details

- Each controller 1,2 switches between several operating modes/locations ℓ_1,ℓ_2 as a function of the water level.
- For the two components and a given composed discrete location $\ell_1 \oplus \ell_2$, the dynamics are:

$$\begin{aligned} \dot{x}_1 &= \alpha_1 u_1 - \beta_1 \sqrt{x_1} & \qquad \dot{x}_2 &= \alpha_2 u_2 - \beta_2 \sqrt{x_2} \\ \dot{u}_1 &= \psi_{1,\ell_1}(x_1) & \qquad \dot{u}_2 &= \psi_{2,\ell_2}(x_2, x_1) \end{aligned}$$

  where the constants α, β depend on the tank and pipe sections, and ψ depends on the specific controller (e.g., hysteretic, proportional) and the location/mode.

- To remove the dependency from x_1, we introduce an input v that overapproximates the behavior of C_1 in respect to x_1.

Decomposition in the hybrid space

Analyzing the decoupled component

1. First, we analyze C_1 separately, obtaining the reachable set Re_1.
2. Then, we analyze the decoupled C_2 on each location ℓ_2:

$$\begin{aligned} \dot{x}_2 &= \alpha_2 u_2 - \beta_2 \sqrt{x_2} \\ \dot{u}_2 &= \psi_{2,\ell_2}(x_2, v) \end{aligned}$$

where we set $v \in Re_1 \downarrow_{x_1}$ as an input.

By decoupling we discarded all discrete synchronisation information between C_1 and C_2: hence we need to restrict v to the whole Re_1.

Decomposition in the hybrid space

General issues

Reachable sets might be significantly large

We need an appropriate representation for a reachable set to minimise the corresponding ranges of the introduced inputs.

Decomposition in the hybrid space

General issues

Reachable sets might be significantly large

We need an appropriate representation for a reachable set to minimise the corresponding ranges of the introduced inputs.

Closed loop systems require initial assumptions on inputs

This translates into successive refinements of the inputs until convergence, if numerically possible, is obtained.

Outline

From differential equations to differential inclusions

We use differential inclusions(DI) to extend ARIADNE's framework to time-varying inputs.

The dynamics of the continuous state $x \in \mathbb{R}^n$ in a DI is expressed as

$$\dot{x} \in F(x, V)$$

where $V \subset \mathbb{R}^m$ is the input set that represent the sources of uncertainty in the dynamics.

The above inclusion is equivalent to

$$\dot{x}(t) = F(x(t), v(t))$$

for some unspecified input trajectory $v(t)$ bounded by V

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.
- In order for the approximation to be tight, we perform it at each integration step k.
  - We call the resulting approximate solution $y(t_k)$.

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.
- In order for the approximation to be tight, we perform it at each integration step k.
  - We call the resulting approximate solution $y(t_k)$.
- On each integration step k:
  1. We identify proper $w_{i,k}$, $\forall i$ and compute $y(t_k)$ from $x(t_{k-1})$.

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.
- In order for the approximation to be tight, we perform it at each integration step k.
  - We call the resulting approximate solution $y(t_k)$.
- On each integration step k:
  1. We identify proper $w_{i,k}$, $\forall i$ and compute $y(t_k)$ from $x(t_{k-1})$.
  2. We compute an approximation error ε_k such that $|x(t_k) - y(t_k)| \leq \varepsilon_k$.

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.
- In order for the approximation to be tight, we perform it at each integration step k.
  - We call the resulting approximate solution $y(t_k)$.
- On each integration step k:
  1. We identify proper $w_{i,k}$, $\forall i$ and compute $y(t_k)$ from $x(t_{k-1})$.
  2. We compute an approximation error ε_k such that $|x(t_k) - y(t_k)| \leq \varepsilon_k$.
  3. We over-approximate the exact solution with $\tilde{x}(t_k) = y(t_k) \pm \varepsilon_k$.

The general approach

- To compute the solution of $\dot{x}(t) = F(x(t), v(t))$, we approximate each v_i with a function we call w_i, $i = 1, \ldots, m$.
- In order for the approximation to be tight, we perform it at each integration step k.
  - We call the resulting approximate solution $y(t_k)$.
- On each integration step k:
  1. We identify proper $w_{i,k}$, $\forall i$ and compute $y(t_k)$ from $x(t_{k-1})$.
  2. We compute an approximation error ε_k such that $|x(t_k) - y(t_k)| \leq \varepsilon_k$.
  3. We over-approximate the exact solution with $\tilde{x}(t_k) = y(t_k) \pm \varepsilon_k$.
  4. We use $\tilde{x}(t_k)$ in place of $x(t_k)$ on the $(k+1)$-th step.

Restrictions currently used in ARIADNE

In order to identify tight analytical expressions for ε, some restrictions are currently in place:

Restrictions currently used in ARIADNE

In order to identify tight analytical expressions for ε, some restrictions are currently in place:

- We focus on input-affine dynamics of the form

$$\dot{x}(t) = f(x(t)) + \sum_{i=1}^{m} g_i(x(t))v_i(t)$$

  with $\{f, g_i\}$ non-linear.
  - This represents a limitation in terms of ability to substitute variables with inputs.

Restrictions currently used in ARIADNE

In order to identify tight analytical expressions for ε, some restrictions are currently in place:

- We focus on input-affine dynamics of the form

$$\dot{x}(t) = f(x(t)) + \sum_{i=1}^{m} g_i(x(t))v_i(t)$$

  with $\{f, g_i\}$ non-linear.
  - This represents a limitation in terms of ability to substitute variables with inputs.

- We settle on a box $V = \prod_i [-V_i, V_i]$, with $|v_i(t)| \leq V_i$
  - This is quite coarse when input values represent sets, as usually required by hybrid decomposition.

The auxiliary functions $w_{i,k}$

The auxiliary system we construct relies on replacing v_i at each step k with a function $w_{i,k}$ in a number of parameters $a_{i,k}$ bounded by V_i. We currently can use the following auxiliary functions:

0) Zero: $w_{i,k}(t) = 0$,

1) Constant: $w_{i,k}(t) = a_{i,k}^{(0)}$,

2a) Affine: $w_{i,k}(t) = a_{i,k}^{(0)} + a_{i,k}^{(1)}\,(t - t_{k+1/2})/h_k$,

2b) Sinusoidal: $w_{i,k}(t) = a_{i,k}^{(0)} + a_{i,k}^{(1)} \sin\left(\gamma(t - t_{k+1/2})/h_k\right)$,

2c) Piecewise: $w_{i,k}(t) = \begin{cases} a_{i,k}^{(0)} \text{ if } t \in [t_k, t_{k+1/2}) \\ a_{i,k}^{(1)} \text{ if } t \in [t_{k+1/2}, t_{k+1}), \end{cases}$

where $h_k = t_{k+1} - t_k$ is the integration step size and $\gamma \approx 4.163152$.

The approximation error ε_k

- In the most general case ε_k is a function of the C^2 norms of f, g_i on the domain D_k, and of the step size h_k;
- The error formula depends on the auxiliary functions chosen: the more the parameters used, the tighter the result;
- For two-parameter auxiliary functions, it is specialised for the cases of additive inputs or single input.

The approximation error ε_k

- In the most general case ε_k is a function of the C^2 norms of f, g_i on the domain D_k, and of the step size h_k;
- The error formula depends on the auxiliary functions chosen: the more the parameters used, the tighter the result;
- For two-parameter auxiliary functions, it is specialised for the cases of additive inputs or single input.

Order of the error based on the number of parameters:

- 0: $O(h_k)$,
- 1: $O(h_k^2)$,
- 2: $\begin{cases} O(h_k^2) + O(h_k^3) & \text{but can get to } O(h_k^3), \\ O(h_k^3) & \text{if additive inputs or single input.} \end{cases}$

The algorithm for a single continuous step

Let $S_k = \left\{ q_k(p) \pm e_k \mid p \in [-1, 1]^{P_k} \right\}$ be an over-approximation of the reachable set at time t_k.

1. Choose auxiliary functions $w_{i,k}(t, a_{i,k})$;
2. Compute the flow $\tilde{\phi}_k(x_k, a_k)$ of

$$\dot{x}(t) = f(x(t)) + \sum_{i=1}^{m} g_i(x(t)) w_{i,k}(t, a_{i,k})$$

   for $t \in [t_k, t_{k+1}]$, $x_k = x(t_k) \in S_k$;
3. Add the uniform error bound ε_k;
4. Compute the set S_{k+1} which over-approximates the reachable set at time t_k.

Implementation aspects

Reconditioning the set

At each step the set S_k will increase its number of parameters

$\rightarrow$ For efficiency purposes, we need to periodically simplify the set by reducing such number at the cost of accuracy.

Implementation aspects

Reconditioning the set

At each step the set S_k will increase its number of parameters

$\rightarrow$ For efficiency purposes, we need to periodically simplify the set by reducing such number at the cost of accuracy.

Choosing the auxiliary function dynamically

Finite accuracy in the representation of a set has an additional effect on the approximation quality

- Using a high-order $w_{i,k}$, while yielding a lower ε, may not result in the tightest S_{k+1}

$\rightarrow$ Instead of using a fixed $w_{i,k}$, we dynamically evaluate each auxiliary function and choose the one giving the tightest set.

Outline

Three different tools

- ARIADNE
- FLOW*
  - developed at UC Boulder and at RWTH Aachen University
  - non-linear ODEs (polynomial dynamics inside modes, polyhedral guards on discrete transitions)
  - state space is represented by Taylor models
  - results are guaranteed to be numerically sound
- CORA
  - developed at TU München
  - a collection of MATLAB classes for the formal verification of cyber-physical systems
  - results are not numerically sound

A benchmark of 10 continuous systems

Name	Alias	Inputs	Vars	Avg. Order	Additive?	h	T_e
Higgins-Sel'kov	HS	2	3	3	N	1/50	10
Chemical Reactor	CR	4	3	2	N	1/16	10
Lotka-Volterra	LV	2	2	2	N	1/50	10
Jet Engine	JE	2	2	2	Y	1/50	5
PI Controller	PI	2	1	2	Y	1/32	5
Jerk Eq. 21	J21	3	1	5/3	N	1/16	10
Lorenz Attractor	LA	3	1	5/3	N	1/256	1
Rössler Attractor	RA	3	1	5/3	Y	1/128	12
Jerk Eq. 16	J16	3	1	4/3	Y	1/16	10
DC-DC Converter	DC	2	2	1	N	1/10	5

Comparison for the same execution time

setup			system									
noise	tool		HS	CR	LV	JE	PI	J21	LA	RA	J16	DC
$\times\frac{1}{4}$	ARIADNE	Σ_V	**109.1**	**1573**	**69.31**	**29.21**	**12.82**	**36.31**	**33.48**	**385.6**	**58.56**	4.877
	CORA	Σ_V	49.42	4656	N/A	27.20	13.00	8.753	12.23	464.8	51.47	7.655
	FLOW*	Σ_V	64.92	643.3	2.161	26.55	11.10	N/A	14.26	133.5	56.25	**7.725**
$\times\frac{1}{2}$	ARIADNE	Σ_V	**76.77**	**943.3**	**32.70**	21.85	**8.849**	**30.47**	**17.64**	**221.5**	39.67	3.816
	CORA	Σ_V	43.53	2684	N/A	22.84	9.360	12.96	11.82	270.6	40.70	3.820
	FLOW*	Σ_V	54.76	384.5	N/A	**22.65**	7.994	N/A	11.56	94.44	**42.42**	**3.860**
$\times 1$	ARIADNE	Σ_V	**48.91**	**502.4**	**14.54**	15.47	5.492	**23.10**	9.070	**113.8**	23.77	1.906
	CORA	Σ_V	33.15	1364	N/A	16.50	6.000	16.73	9.900	153.8	27.62	1.902
	FLOW*	Σ_V	40.50	185.5	N/A	**16.49**	**5.690**	8.974	**9.416**	73.96	**27.90**	**1.924**
$\times 2$	ARIADNE	Σ_V	**23.36**	**217.6**	**5.947**	9.368	3.085	**15.41**	4.574	**58.85**	13.11	0.944
	CORA	Σ_V	20.07	612.4	N/A	10.33	3.507	15.24	6.284	79.61	15.85	0.944
	FLOW*	Σ_V	21.13	69.44	N/A	**10.34**	**3.316**	10.95	**6.045**	53.06	**16.18**	**0.955**
$\times 4$	ARIADNE	Σ_V	**11.49**	**60.87**	**1.165**	4.953	1.664	8.807	2.255	29.12	6.570	0.464
	CORA	Σ_V	1.086	214.1	N/A	5.537	1.909	9.864	3.201	37.97	6.772	0.465
	FLOW*	Σ_V	1.725	N/A	N/A	**6.421**	**1.836**	**9.133**	**3.286**	**32.23**	**8.452**	**0.471**

Outline

Extensions

More complex representation of the input set

Especially for hybrid decomposition, it is necessary to improve on the box representation that is currently used.

- Zonotope representation and nonlinear constraints
- Abstraction of components providing the inputs

Extensions

More complex representation of the input set

Especially for hybrid decomposition, it is necessary to improve on the box representation that is currently used.

- Zonotope representation and nonlinear constraints
- Abstraction of components providing the inputs

Nonlinearity in the inputs

For decomposition in general, we need to allow any variable replacement. An affine relation is not sufficient in many cases.

- Linearisation in respect to the inputs

A selection of publications

Differential inclusions

Geretti, L.; Zivanovic Gonzalez, S.; Collins, P.; Bresolin, D.; Villa, T. "Rigorous Continuous Evolution of Uncertain Systems", to appear on 12th International Workshop on Numerical Software Verification 2019 (collocated with CAV), July 13-14 2019, New York, NY, USA.

Collins, P.; Zivanovic Gonzalez, S. "Numerical solutions to noisy systems", 49th IEEE Conference on Decision and Control (CDC), pg. 798-803, Dec 2010

Assume-guarantee verification

Benvenuti, L; Bresolin, D.; Collins, P.; Ferrari, A.; Geretti, L.; Villa, T. "Assume-guarantee verification of nonlinear hybrid systems with Ariadne", International Journal of Robust and Nonlinear Control, Volume 24, Issue 4, Mar. 2014, pg. 699-724, ISSN: 1049-8923, DOI: 10.1002/RNC.2914

Contract-based design and survey of the state of the art

Nuzzo, P.; Sangiovanni-Vincentelli, A.L.; "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems", Proceedings of the IEEE, Volume 103, Issue 11, 2015, pg. 2104-2132, DOI: 10.1109/JPROC.2015.2453253