Formal verification of nonlinear hybrid systems using Ariadne

Tiziano Villa and Luca Geretti

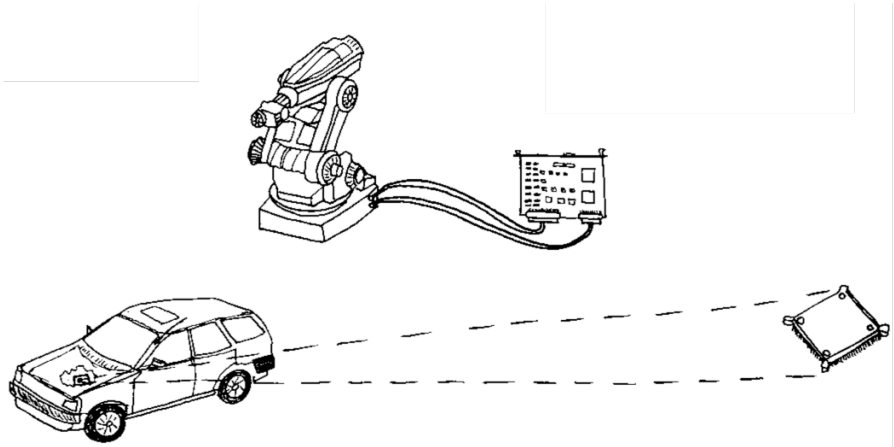
University of Verona

September 8, 2017

Hybrid systems and nonlinearity

Why consider systems as hybrid?

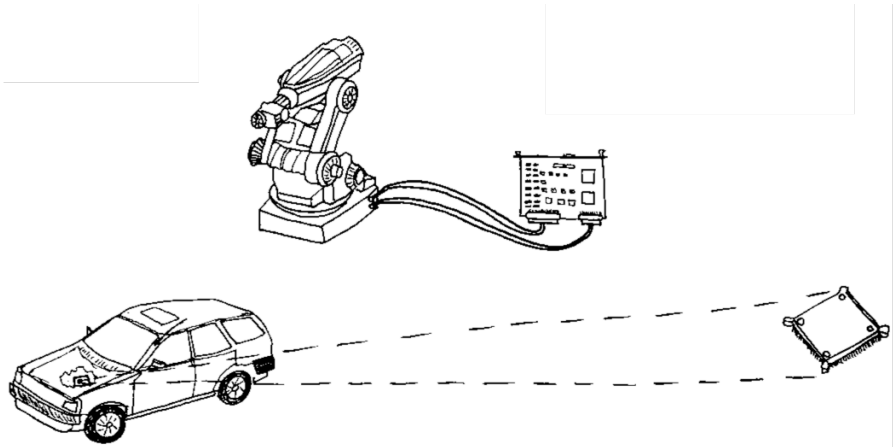
Real systems composed of a controller with switching *discrete states* and an environment with evolving *continuous variables*.



Hybrid systems and nonlinearity

Why consider systems as hybrid?

Real systems composed of a controller with switching *discrete states* and an environment with evolving *continuous variables*.



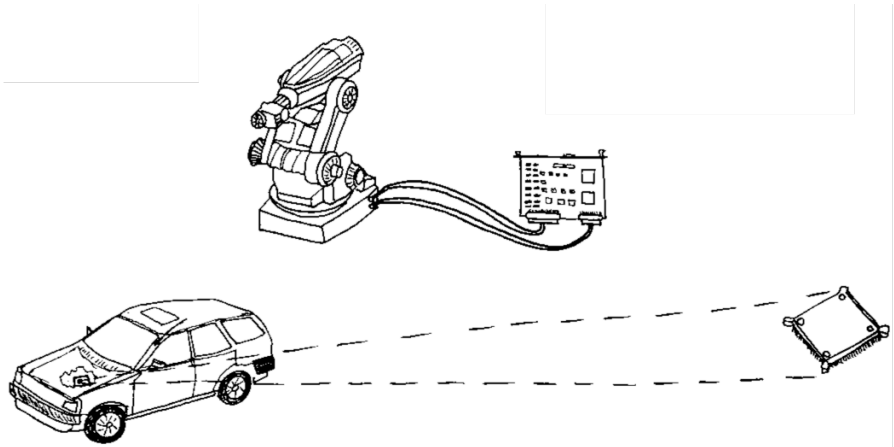
Why consider systems as nonlinear?

The environment is rarely linear, sometimes specifically exploiting nonlinearity (e.g., analog electronics).

Hybrid systems and nonlinearity

Why consider systems as hybrid?

Real systems composed of a controller with switching *discrete states* and an environment with evolving *continuous variables*.



Why consider systems as nonlinear?

The environment is rarely linear, sometimes specifically exploiting nonlinearity (e.g., analog electronics).

In general, for our *formal* results we want to be as faithful to the real system as possible

Reachability analysis

To verify whether a dynamical system satisfies some properties, we describe its behaviour by computing the set of reached states (reachable set) Re.

- It allows full observation of system evolution (compared to abstraction methods).

Reachability analysis

To verify whether a dynamical system satisfies some properties, we describe its behaviour by computing the set of reached states (reachable set) Re.

- It allows full observation of system evolution (compared to abstraction methods).

Re is not computable in general, in particular for **nonlinear** systems.

Reachability analysis

To verify whether a dynamical system satisfies some properties, we describe its behaviour by computing the set of reached states (reachable set) Re.

- It allows full observation of system evolution (compared to abstraction methods).

Re is not computable in general, in particular for **nonlinear** systems.

Re can be approximated, but not in *both* an effective and efficient way.

- Some operations on accurate representations are still undecidable.
- Coarse approximations are problematic in terms of reliability of results.

How to approximate regions

Possible choices of approximating Re:

1. **Inner approximation I:** Re strictly contains I.
2. **Outer approximation O:** Re is strictly contained in O (an over-approximation of Re).
3. **ε-lower approximation L_ε:** every point of L_ε is at a distance less than ε from Re (an over-approximation of a subset of Re).

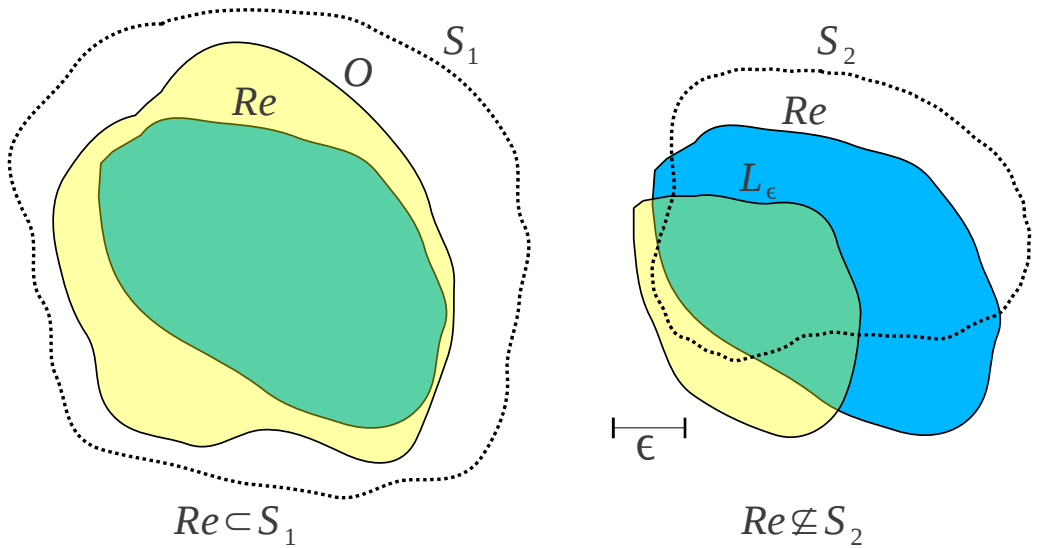
How to approximate regions

Possible choices of approximating Re:

1. **Inner approximation I:** Re strictly contains I.
2. **Outer approximation O:** Re is strictly contained in O (an over-approximation of Re).
3. **ε-lower approximation L_ε:** every point of L_ε is at a distance less than ε from Re (an over-approximation of a subset of Re).

- Inner approximation is not computable in general.
- Outer and ε-lower approximations can be used to verify/falsify properties.

Property satisfaction in terms of sets



- Re is the reachable set, which is unobservable.
- S_1, S_2 are sets satisfying given properties.
- O is the outer approximation of Re.
- L_ε is the ε-lower approximation of Re.

A summary of ARIADNE

- Developed by a joint team including the University of Verona, the University of Maastricht, the University of Padova, and the University of Barry (Florida)
- Uses the formalism of hybrid automata to describe nonlinear time-continuous systems.
- Based on rigorous semantics paired with interval arithmetics to guarantee correctness of verification over approximated sets.
- Written as a C++ library, released as an open source distribution: `http://www.ariadne-cps.org`

Semantics used for evolution

Upper semantics

When numerical inaccuracies make the transition undecidable, all possible choices are taken. Computed sets do not need to include a point of Re.

Semantics used for evolution

Upper semantics

When numerical inaccuracies make the transition undecidable, all possible choices are taken. Computed sets do not need to include a point of Re.

Lower semantics

When numerical inaccuracies make the transition undecidable, evolution stops. Computed sets must include at least one point of Re.

Reachable set approximations available

ARIADNE can compute the following approximations of the reachable set (available semantics under parentheses):

- An over-approximated subset, up to a given time t : for proving/disproving properties where a bound on the evolution time is identified [upper, lower];
- An outer approximation : for proving properties using infinite-time evolution [upper];
- For a given $\varepsilon > 0$, an ε-lower approximation : for disproving properties using infinite-time evolution [lower].

Some verification procedures available

Safety

Verify whether a system satisfies some given safety properties.

Dominance

Given two controllers, verify which one has the largest safety margin for given closed-loop requirements.

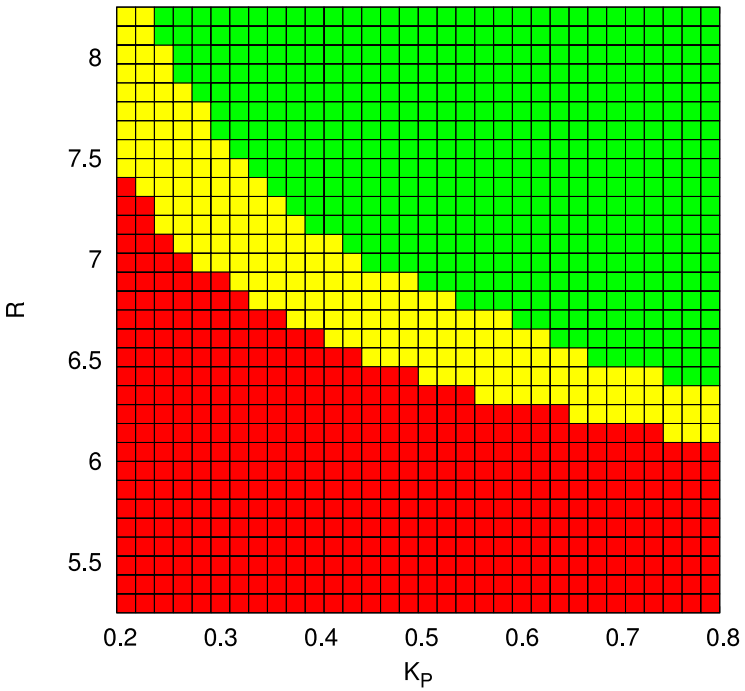
Some verification procedures available

Safety

Verify whether a system satisfies some given safety properties.

Dominance

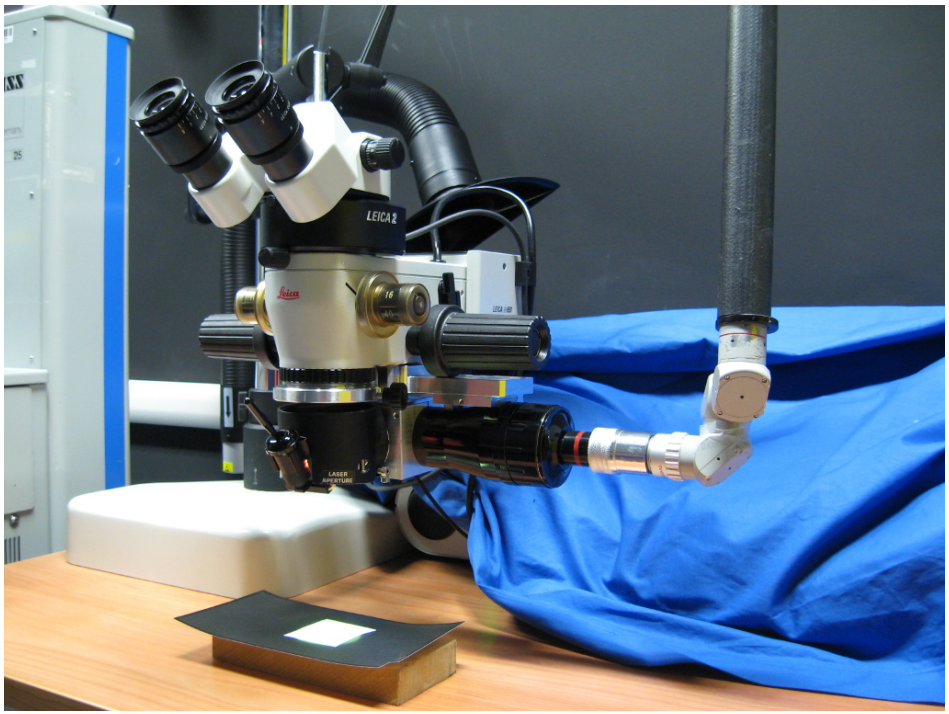
Given two controllers, verify which one has the largest safety margin for given closed-loop requirements.



Both these verifications can be performed parametrically by analyzing intervals of constants and splitting the resulting parameters space.

Current field of application: robotic surgery

- Very strict safety requirements.
- Increasing reliance on assisted control for improved accuracy.
- Traditionally focused on control theory specifications, recently adopting formal verification approaches.



Improving the engine

Better control of over-approximation error

Tune evolution parameters according to local dynamics and quality metrics provided by the user

Improving the engine

Better control of over-approximation error

Tune evolution parameters according to local dynamics and quality metrics provided by the user

Alternative set representations

Adopt different set representations for different subsystems, in order to improve scalability (requires decoupling of subsystems)

Improving the engine

Better control of over-approximation error

Tune evolution parameters according to local dynamics and quality metrics provided by the user

Alternative set representations

Adopt different set representations for different subsystems, in order to improve scalability (requires decoupling of subsystems)

Increasing capabilities of the library

Implement differential inclusions

Allow to model noisy inputs, thus expanding the validity of verification results. Also required to model unspecified inputs, enabling the verification of subsystems in partial isolation

Increasing capabilities of the library

Implement differential inclusions

Allow to model noisy inputs, thus expanding the validity of verification results. Also required to model unspecified inputs, enabling the verification of subsystems in partial isolation

Extend the model to introduce modularity

Allow component instances, safe substitution of components

Increasing capabilities of the library

Implement differential inclusions

Allow to model noisy inputs, thus expanding the validity of verification results. Also required to model unspecified inputs, enabling the verification of subsystems in partial isolation

Extend the model to introduce modularity

Allow component instances, safe substitution of components

Expand the verification/automation routines

For example:

- controller synthesis starting from desired properties
- contract-based design for modular systems