

Unbounded evolution of a set

Luca Geretti

University of Verona, Italy



Outline

Infinite vs finite time evolution

Finite time is simple, but may not be usable

Using finite time evolution to verify a system which evolves for infinite time requires the manual identification of a **time interval** that still gives formal guarantees.

- Example: if the behavior is **guaranteed** to be periodic, analyze only one period.

Infinite vs finite time evolution

Finite time is simple, but may not be usable

Using finite time evolution to verify a system which evolves for infinite time requires the manual identification of a **time interval that still gives formal guarantees**.

- Example: if the behavior is **guaranteed** to be periodic, analyze only one period.

Infinite-time evolution in practice

A sequence of finite-time evolutions, which terminates if no additional state space can be reached after a while.

In general, to verify some properties of the system we need to evolve the system for infinite time.

Convergence for infinite-time evolution

To obtain convergence, we have two requirements:

1. Be able to identify when no new state space is reached;
2. Limit the growth of the overapproximation error.

Convergence for infinite-time evolution

To obtain convergence, we have two requirements:

1. Be able to identify when no new state space is reached;
2. Limit the growth of the overapproximation error.

We need a set representation with

- operations like subtraction, intersection, splitting and merging;
- small memory usage, fast operations and good scalability;
- small overapproximation error.

Convergence for infinite-time evolution

To obtain convergence, we have two requirements:

1. Be able to identify when no new state space is reached;
2. Limit the growth of the overapproximation error.

We need a set representation with

- operations like subtraction, intersection, splitting and merging;
- small memory usage, fast operations and good scalability;
- small overapproximation error.

We use Taylor Sets to respect 2., switching temporarily to (Hybrid) Grid Sets for 1.

How and when to convert into a grid set

1. First, a proper choice of the root cell **shape** is important to partition the state space.
 - ▶ Slower changing variables require finer observation, hence a smaller root cell width (recall here that cells are split progressively across all dimensions).
 - ▶ By analysing the vector field in a given location, we can gather how fast each variable changes.

How and when to convert into a grid set

1. First, a proper choice of the root cell **shape** is important to partition the state space.
 - ▶ Slower changing variables require finer observation, hence a smaller root cell width (recall here that cells are split progressively across all dimensions).
 - ▶ By analysing the vector field in a given location, we can gather how fast each variable changes.
2. Conversion ideally should happen periodically with the same period of evolution, if any exists.
 - ▶ A point-based **simulation** of the system can identify any cycles and is fast enough to be performed before rigorous evolution; it can be improved upon subsequent refinements.

How and when to convert into a grid set

1. First, a proper choice of the root cell **shape** is important to partition the state space.
 - ▶ Slower changing variables require finer observation, hence a smaller root cell width (recall here that cells are split progressively across all dimensions).
 - ▶ By analysing the vector field in a given location, we can gather how fast each variable changes.
2. Conversion ideally should happen periodically with the same period of evolution, if any exists.
 - ▶ A point-based **simulation** of the system can identify any cycles and is fast enough to be performed before rigorous evolution; it can be improved upon subsequent refinements.
3. The obtained **lock-to-grid strategy** is a sufficiently long sequence of (hybrid) times that guess the finite-time evolution to be performed.
 - ▶ A sensibly less sophisticated choice of an infinite sequence of hybrid times is perfectly acceptable, e.g., convert after either one transition or 10 seconds have passed.

Infinite-time reachability at a glance

1. Identify a **bounding set** B to constrain evolution;
2. Identify a **lock-to-grid strategy**;
3. Compute the **finite-time** hybrid evolution of the automaton up to the next lock-to-grid time;
4. If the reached set is outside the bounding set, stop with failure;
5. If **new** cells have been found in this iteration, resume from (3);
6. Stop with success.

Specific comments on computing O

1. We resume evolution from the new intermediate cells, converted into Taylor sets.
 - ▶ This is also able to address the problem of merging multiple trajectories (e.g., after a transition), and of splitting a set when too large.

Specific comments on computing O

1. We resume evolution from the new intermediate cells, converted into Taylor sets.
 - ▶ This is also able to address the problem of merging multiple trajectories (e.g., after a transition), and of splitting a set when too large.
2. The new intermediate cells after one round of finite evolution must be obtained by comparison with the **accumulated intermediate cells** instead of the accumulated reached cells.
 - ▶ Not all points in the reached cells have been actually evolved.
3. If B is crossed, no information can be gathered: either the infinite-time reachable set is actually outside B , or the accuracy is insufficient for convergence.

Specific comments on computing L_ϵ (1)

We cannot split a set

We would lose the information on which cells contain points of the reachable set: $\|Re - L_\epsilon\|$ would be untrackable.

→ We cannot resume evolution from cells.

Specific comments on computing L_ϵ (1)

We cannot split a set

We would lose the information on which cells contain points of the reachable set: $\|Re - L_\epsilon\|$ would be untrackable.

→ We cannot resume evolution from cells.

Any L_ϵ of a subset of Re is still a valid L_ϵ of Re

An empty set for example is valid, while uninformative.

- The termination clause can be relaxed: we may stop as soon as the set width becomes as large as ϵ
- For efficiency purposes we can still evolve subsets of initial sets: particularly useful when large sets would be involved

Specific comments on computing L_ϵ (2)

How can we "approximate" infinite-time reachability?

- We maintain a **separate grid set**, which is updated in parallel only to check if no new cells are reached.
- We do not resume from the intermediate cells, but from the original intermediate Taylor sets (**or any subset of them**).
- We may terminate prematurely, but that is fine: if convergence is possible, with increasing accuracy we **converge from below**.

Specific comments on computing L_ε (2)

How can we "approximate" infinite-time reachability?

- We maintain a **separate grid set**, which is updated in parallel only to check if no new cells are reached.
- We do not resume from the intermediate cells, but from the original intermediate Taylor sets (**or any subset of them**).
- We may terminate prematurely, but that is fine: if convergence is possible, with increasing accuracy we **converge from below**.

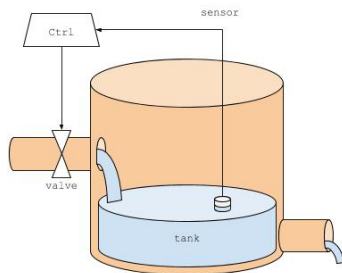
An invalid B can be detected using L_ε

If the reached set is outside B more than ε , then B is too restrictive with respect to Re and **should be enlarged**.

- 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

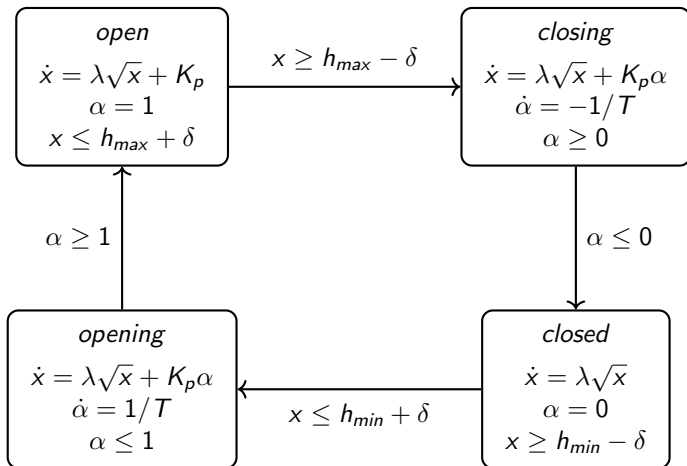
Outline

The watertank example



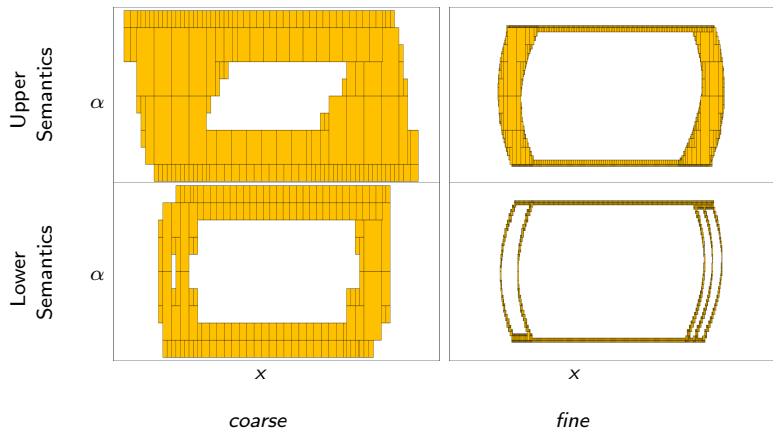
- Outlet flow F_{out} depends on the water level $x(t)$:
$$F_{out}(t) = \lambda \sqrt{x(t)}$$
- Inlet flow F_{in} is controlled by the valve position $\alpha(t)$:
$$F_{in}(t) = K_p \cdot \alpha(t)$$
- The controller senses the water level and sends the appropriate commands to the valve.

The watertank automaton



Reachability results

On the watertank example



Outline

- We use again
`tutorials/hybrid_evolution/hybrid_evolution_tutorial.cpp`
 1. Run with verbosity 2, commenting `compute_evolution` and uncomment `compute_reachability`
 2. Change the maximum grid fineness down
 3. Run with verbosity 3, commenting graphical output