
Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement and Integrated Falsification

Stefan Ratschan

Max-Planck-Institut für Informatik, Saarbrücken, Germany

Czech Academy of Sciences, Prague, Czech Republic

Zhikun She (Max-Planck-Institut für Informatik)

Jan-Georg Smaus (Albert-Ludwigs-Universität Freiburg)

The Problem

Safety verification/falsification of hybrid systems:

- Init: $x \leq 1$
- Unsafe: $s = \text{off} \wedge x \geq 10$
- Flow:
 $(s = \text{on} \rightarrow \dot{x} = x \sin x + y \wedge \dot{y} = x^2) \wedge$
 $(s = \text{off} \rightarrow \dot{x} = 1 \wedge 1.8 \leq \dot{y} \wedge \dot{y} \leq 2)$
- Jump: $y \geq 5 \rightarrow s' = \text{off}$

The Problem

Safety verification/falsification of hybrid systems:

- Init: $x \leq 1$
- Unsafe: $s = \text{off} \wedge x \geq 10$
- Flow:
 $(s = \text{on} \rightarrow \dot{x} = x \sin x + y \wedge \dot{y} = x^2) \wedge$
 $(s = \text{off} \rightarrow \dot{x} = 1 \wedge 1.8 \leq \dot{y} \wedge \dot{y} \leq 2)$
- Jump: $y \geq 5 \rightarrow s' = \text{off}$

Output: Either

- a counter-example: trajectory from a state satisfying Init to a state satisfying Unsafe (falsification), or
- if such a trajectory does not exist: “safe” (verification)

Contents

- Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement (HSCC'05, ACM_TECS)
- Verification-integrated falsification (IFAC ADHS'06)

Verification

Verification

classical approach (e.g., HyTech): compute reach set, if disjoint from set of unsafe states, done

Verification

classical approach (e.g., HyTech): compute reach set, if disjoint from set of unsafe states, done

- advantage: reach set often has its own value

Verification

classical approach (e.g., HyTech): compute reach set, if disjoint from set of unsafe states, done

- advantage: reach set often has its own value
- disadvantage:
 - might get stuck computing unnecessary information
 - does not use information on set of unsafe values

Abstraction Refinement

idea from software verification, adapted to hybrid systems (Alur et al. 02, Clarke et al. 03, Alur et al. 03)

Abstraction Refinement

idea from software verification, adapted to hybrid systems (Alur et al. 02, Clarke et al. 03, Alur et al. 03)

iteratively refines conservative over-approximations of the hybrid system by a finite systems (the *abstraction*)

Abstraction Refinement

idea from software verification, adapted to hybrid systems (Alur et al. 02, Clarke et al. 03, Alur et al. 03)

iteratively refines conservative over-approximations of the hybrid system by a finite systems (the *abstraction*)

Abstraction Refinement

idea from software verification, adapted to hybrid systems (Alur et al. 02, Clarke et al. 03, Alur et al. 03)

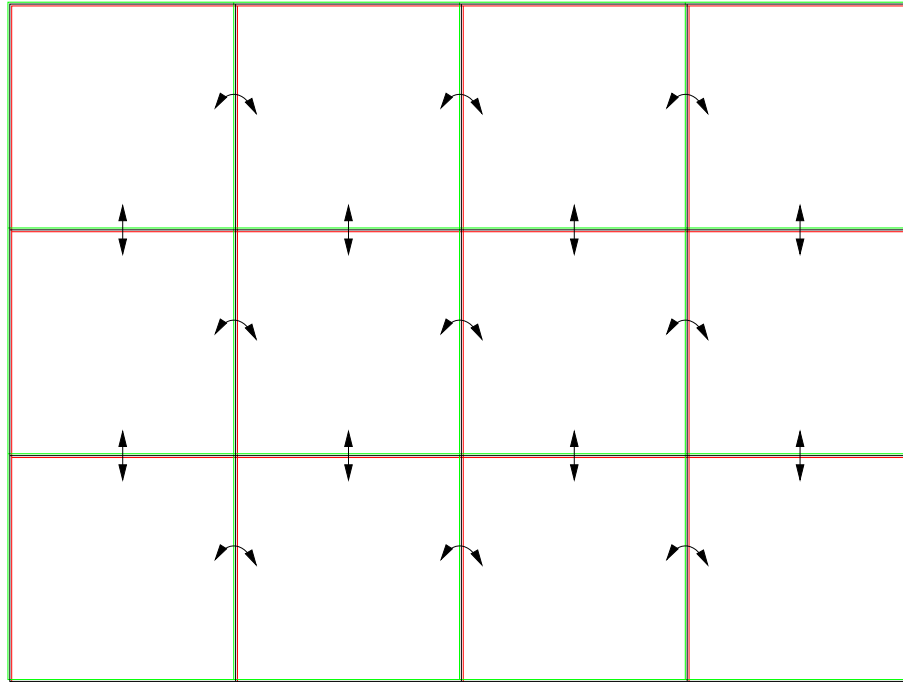
iteratively refines conservative over-approximations of the hybrid system by a finite systems (the *abstraction*)

Merge with Interval Grid Method

Stursberg/Kowalewski et. al., one-mode case:

Merge with Interval Grid Method

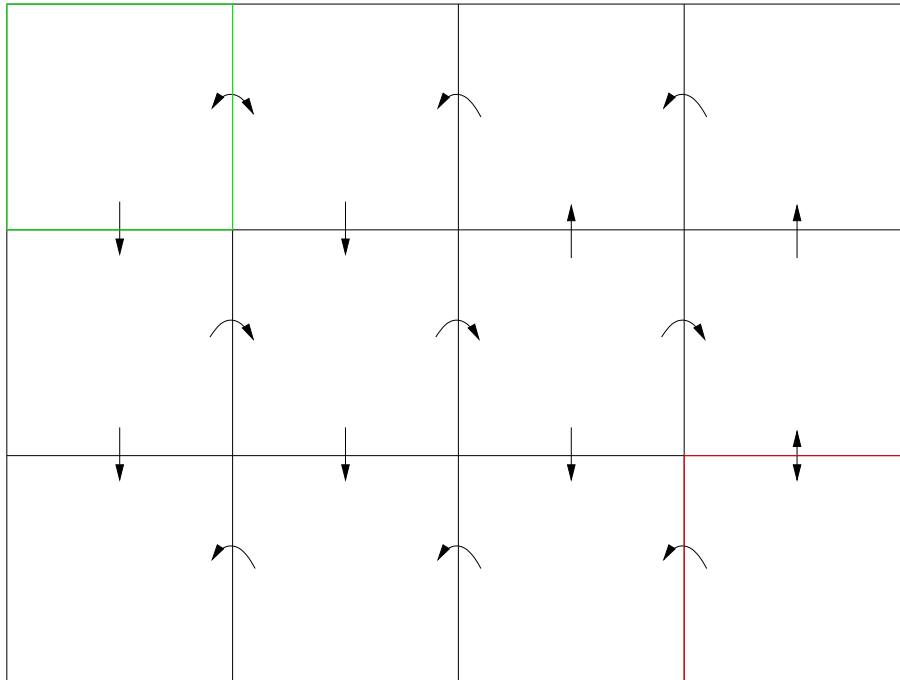
Stursberg/Kowalewski et. al., one-mode case:



- put transitions between all neighboring hyperrectangles (*boxes*), mark all as initial/unsafe

Merge with Interval Grid Method

Stursberg/Kowalewski et. al., one-mode case:

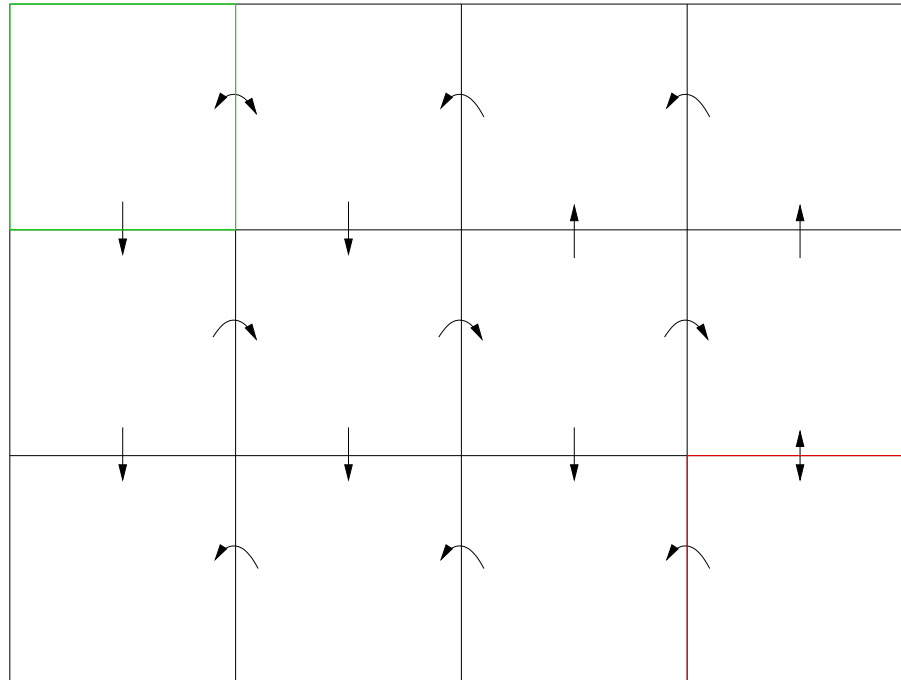


$$\dot{x} \in [-5, -1]$$

- put transitions between all neighboring hyperrectangles (*boxes*), mark all as initial/unsafe
- remove impossible transitions/marks (interval arithmetic check on boundaries/boxes)

Merge with Interval Grid Method

Stursberg/Kowalewski et. al., one-mode case:

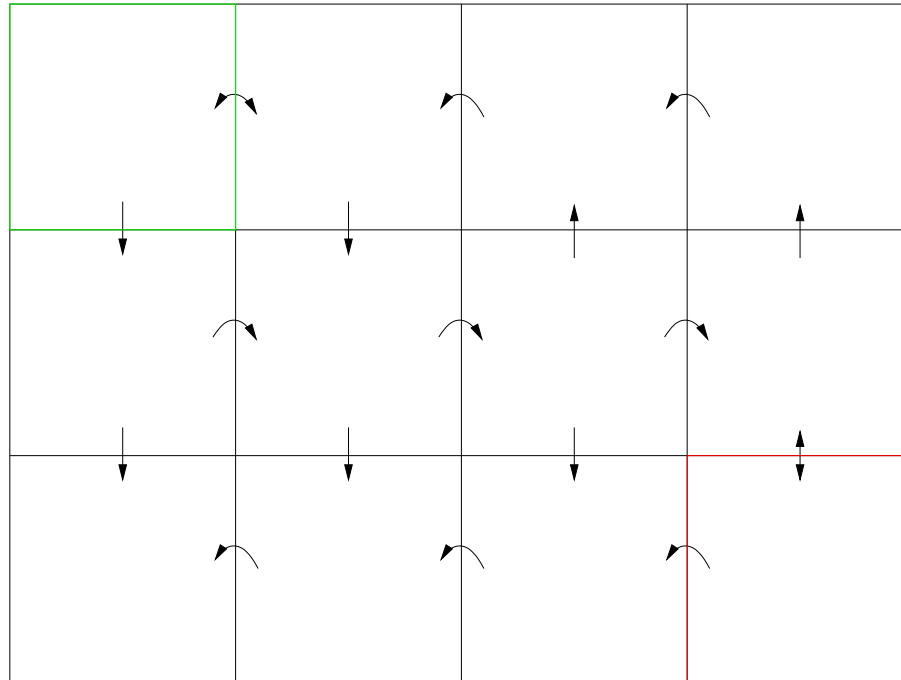


$$\dot{x} \in [-5, 1]$$

- put transitions between all neighboring hyperrectangles (*boxes*), mark all as initial/unsafe
- remove impossible transitions/marks (interval arithmetic check on boundaries/boxes)

Merge with Interval Grid Method

Stursberg/Kowalewski et. al., one-mode case:



$$\dot{x} \in [-5, 1]$$

- put transitions between all neighboring hyperrectangles (*boxes*), mark all as initial/unsafe
- remove impossible transitions/marks (interval arithmetic check on boundaries/boxes)

Result: finite abstraction

Interval Grid Method II

Check safety on resulting finite abstraction

Interval Grid Method II

Check safety on resulting finite abstraction

if safe: finished, otherwise: refine grid;
continue until success

Interval Grid Method II

Check safety on resulting finite abstraction

if safe: finished, otherwise: refine grid;
continue until success

More modes: separate grid for each mode

Interval Grid Method II

Check safety on resulting finite abstraction

if safe: finished, otherwise: refine grid;
continue until success

More modes: separate grid for each mode

Jumps: also check using interval arithmetic

Interval Grid Method II

Check safety on resulting finite abstraction

if safe: finished, otherwise: refine grid;
continue until success

More modes: separate grid for each mode

Jumps: also check using interval arithmetic
remove boxes not on a counter-examples

Discussion

Advantages:

Discussion

Advantages:

- concentrates refinement on relevant parts

Discussion

Advantages:

- concentrates refinement on relevant parts
- can do verification instead of verification modulo rounding errors

Discussion

Advantages:

- concentrates refinement on relevant parts
- can do verification instead of verification modulo rounding errors
- interval tests cheap (e.g., compare to explicit computation of continuous reach sets, or full decision procedures)

Discussion

Advantages:

- concentrates refinement on relevant parts
- can do verification instead of verification modulo rounding errors
- interval tests cheap (e.g., compare to explicit computation of continuous reach sets, or full decision procedures)

Disadvantages:

Discussion

Advantages:

- concentrates refinement on relevant parts
- can do verification instead of verification modulo rounding errors
- interval tests cheap (e.g., compare to explicit computation of continuous reach sets, or full decision procedures)

Disadvantages:

- may require a very fine grid to provide an affirmative answer (curse of dimensionality)

Discussion

Advantages:

- concentrates refinement on relevant parts
- can do verification instead of verification modulo rounding errors
- interval tests cheap (e.g., compare to explicit computation of continuous reach sets, or full decision procedures)

Disadvantages:

- may require a very fine grid to provide an affirmative answer (curse of dimensionality)
- ignores the continuous behavior within the grid elements

Removing Disadvantages

reflect more information in abstraction without creating more boxes by splitting

Removing Disadvantages

reflect more information in abstraction without creating more boxes by splitting

Observation: we do not need to include information on unreachable state space, remove such parts from boxes

Removing Disadvantages

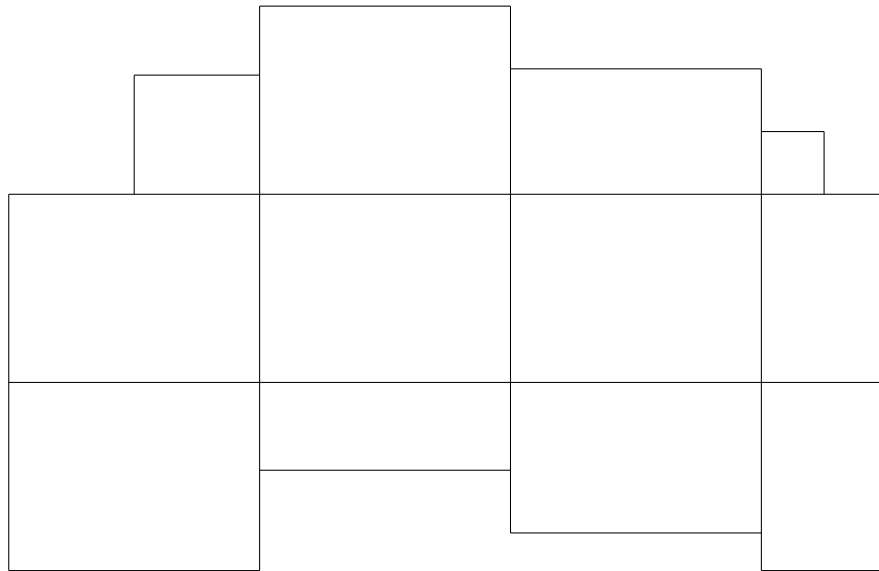
reflect more information in abstraction without creating more boxes by splitting

Observation: we do not need to include information on unreachable state space, remove such parts from boxes

Removing Disadvantages

reflect more information in abstraction without creating more boxes by splitting

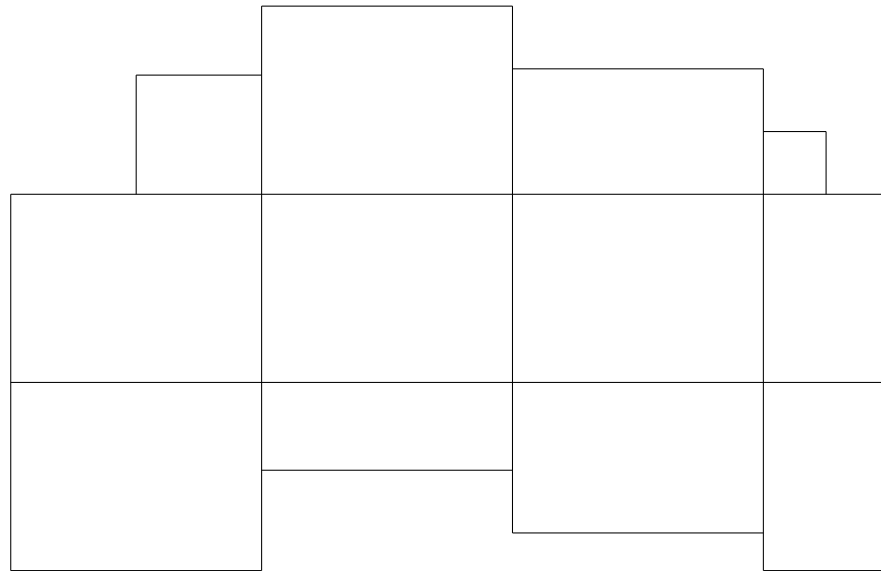
Observation: we do not need to include information on unreachable state space, remove such parts from boxes



Removing Disadvantages

reflect more information in abstraction without creating more boxes by splitting

Observation: we do not need to include information on unreachable state space, remove such parts from boxes



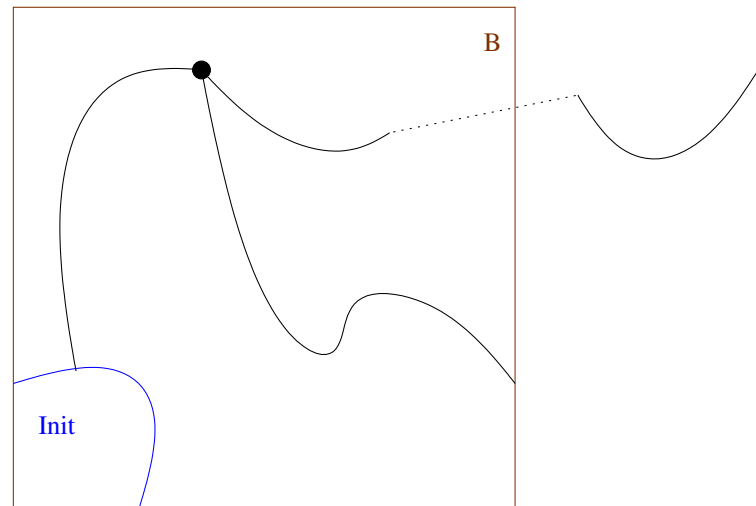
Method: form constraints that hold on reachable parts of state space, remove non-solutions by constraint solver

Reach Set Pruning

Reach Set Pruning

A point in a box B can be reachable

- from the initial set via a flow in B
- from a jump via a flow in B
- from a neighboring box via a flow in B

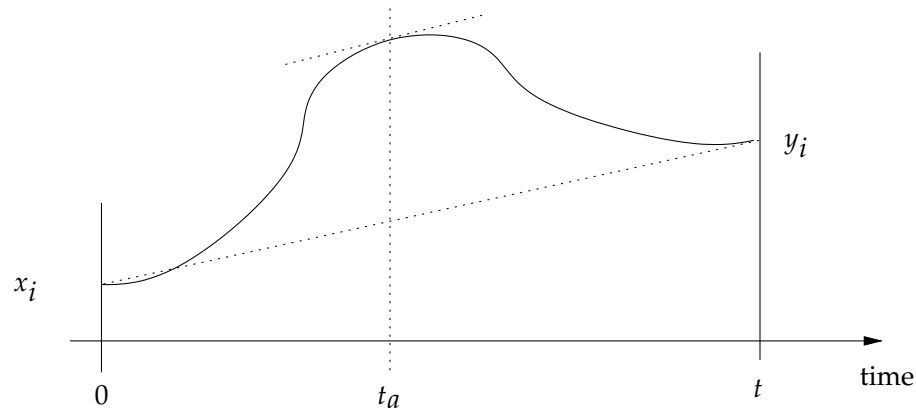


formulate corresponding constraints, remove all points from box that do not fulfill one of these constraints

Reachability Constraints

Lemma: For a box B , mode s , if a point $(y_1, \dots, y_n) \in B$ is reachable from a point $(x_1, \dots, x_n) \in B$ via a flow in B then

$$\exists t \in \mathbb{R}_{\geq 0} \bigwedge_{1 \leq i \leq n} \exists a_1, \dots, a_k, \dot{a}_1, \dots, \dot{a}_k [(a_1, \dots, a_k) \in B \\ \text{Flow}(s, (a_1, \dots, a_k), (\dot{a}_1, \dots, \dot{a}_k)) \wedge y_i = x_i + \dot{a}_i \cdot t]$$



Denote this constraint by $flow_B(s, \mathbf{x}, \mathbf{y})$.

Reachability Constraints

Lemma: For a box $B \subseteq \mathbb{R}^k$, mode s , if $\mathbf{y} \in B$ is reachable from the initial set via a flow in B then

$$\exists \mathbf{x} \in B [Init(s, \mathbf{x}) \wedge flow_B(s, \mathbf{x}, \mathbf{y})]$$

Reachability Constraints

Lemma: For a box $B \subseteq \mathbb{R}^k$, mode s , if $\mathbf{y} \in B$ is reachable from the initial set via a flow in B then

$$\exists \mathbf{x} \in B [Init(s, \mathbf{x}) \wedge flow_B(s, \mathbf{x}, \mathbf{y})]$$

Lemma: For a box $B \subseteq \mathbb{R}^k$, mode s , $\mathbf{y} \in B$, (s, \mathbf{y}) is reachable from a jump from a box B^* and mode s^* via a flow in B then

$$\exists \mathbf{x}^* \in B^* \exists \mathbf{x} \in B [Jump(s^*, \mathbf{x}^*, s, \mathbf{x}) \wedge flow_B(s, \mathbf{x}, \mathbf{y})]$$

Reachability Constraints

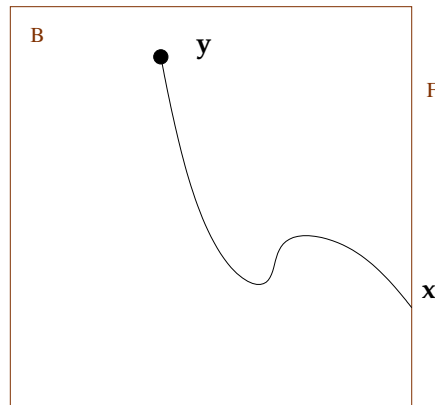
Lemma: For a box $B \subseteq \mathbb{R}^k$, mode s , if $\mathbf{y} \in B$ is reachable from a neighboring box over a face F of B and a flow in B then

$$\exists \mathbf{x} \in F [incoming_F(s, \mathbf{x}) \wedge flow_B(s, \mathbf{x}, \mathbf{y})],$$

where $incoming(s, \mathbf{x})$ is of the form

$$\exists \dot{x}_1, \dots, \dot{x}_k [Flow(s, \mathbf{x}, (\dot{x}_1, \dots, \dot{x}_k)) \wedge \dot{x}_j r 0]$$

where $r \in \{\leq, \geq\}$, $j \in \{1, \dots, k\}$ depends on the face F



for corners etc. a little bit more involved

Using Constraints

After substituting definitions,

Using Constraints

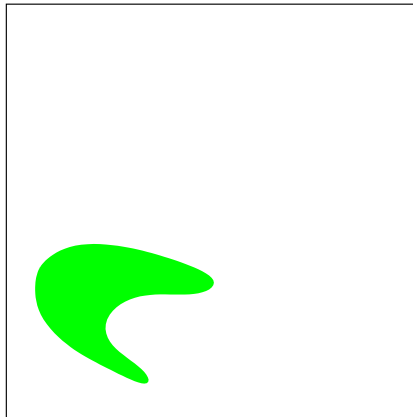
After substituting definitions, getting rid of quantifiers,

Using Constraints

After substituting definitions, getting rid of quantifiers, interval constraint propagation algorithms can remove parts from boxes not fulfilling such constraints.

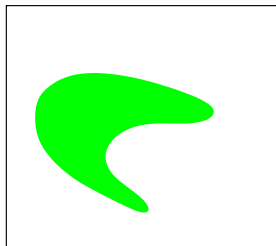
Using Constraints

After substituting definitions, getting rid of quantifiers, interval constraint propagation algorithms can remove parts from boxes not fulfilling such constraints.



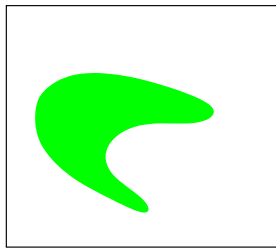
Using Constraints

After substituting definitions, getting rid of quantifiers, interval constraint propagation algorithms can remove parts from boxes not fulfilling such constraints.



Using Constraints

After substituting definitions, getting rid of quantifiers, interval constraint propagation algorithms can remove parts from boxes not fulfilling such constraints.



- correct handling of rounding errors
- almost negligible time
- result not necessarily tight (but tight for $flow_B(s, \mathbf{x}, \mathbf{y})$ in linear case)

<http://rsolver.sourceforge.net>

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

abstraction refinement: transitions for free

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

abstraction refinement: transitions for free

Prototype available from

`http://hsolver.sourceforge.net`

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

abstraction refinement: transitions for free

Prototype available from

`http://hsolver.sourceforge.net`

Can solve many examples for which the interval grid method fails

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

abstraction refinement: transitions for free

Prototype available from

`http://hsolver.sourceforge.net`

Can solve many examples for which the interval grid method fails

works for a more general class of hybrid systems than other tools

Implementation

By applying solver to constraints/grid elements:

- we remove non-reachable elements from boxes
- side-effect: removes drift within boxes!

abstraction refinement: transitions for free

Prototype available from

`http://hsolver.sourceforge.net`

Can solve many examples for which the interval grid method fails

works for a more general class of hybrid systems than other tools

But: still lots of room for improvement

Integrated Falsification

observation: we remove abstract states not on a counter-example

hence: abstraction does

- *not* over-approximate *all* trajectories of concrete system, but
- over-approximates set of concrete counter-examples

use it in search for counter-examples!

Integrated Falsification

observation: we remove abstract states not on a counter-example

hence: abstraction does

- *not* over-approximate *all* trajectories of concrete system, but
- over-approximates set of concrete counter-examples

use it in search for counter-examples!

assume: systems with non-deterministic continuous evolution

e.g.:

$$\begin{aligned}x_2 - 1 &\leq \dot{x}_1 \leq x_2 + 1 \\ -x_1 - 1 &\leq \dot{x}_2 \leq -x_1 + 1\end{aligned}$$

Integrated Falsification II

assume: abstract counter-example $(m_0, B_0), \dots, (m_k, B_k)$

Integrated Falsification II

assume: abstract counter-example $(m_0, B_0), \dots, (m_k, B_k)$

find: $\mathbf{x}_0 \in B_0, \dots, \mathbf{x}_k \in B_k$, such that a piecewise polynomial trajectory through $(m_0, \mathbf{x}_0), \dots, (m_k, \mathbf{x}_k)$ forms a concrete counter-example

Integrated Falsification II

assume: abstract counter-example $(m_0, B_0), \dots, (m_k, B_k)$

find: $\mathbf{x}_0 \in B_0, \dots, \mathbf{x}_k \in B_k$, such that a piecewise polynomial trajectory through $(m_0, \mathbf{x}_0), \dots, (m_k, \mathbf{x}_k)$ forms a concrete counter-example

concretization as a constraint solving problem:

$$\left(\bigwedge_{i \in \{1, \dots, k\}} \mathbf{x}_i \in B_i \right) \wedge \text{Init}_{m_0}(\mathbf{x}_0) \wedge$$

$$\bigwedge_{i \in \{1, \dots, k\}} \left(\text{Jump}_{m_{i-1}, m_i}(\mathbf{x}_{i-1}, \mathbf{x}_i) \vee \right.$$

$$\left. \left(m_{i-1} = m_i \wedge \text{flow}_{m_i}(\mathbf{x}_{i-1}, \mathbf{x}_i) \right) \right) \wedge$$

$$\text{Unsafe}_{m_k}(\mathbf{x}_k)$$

Polynomial Trajectories

polynomial trajectory $\phi_{\mathbf{a}}$ from time to state space in parameters \mathbf{a} (e.g., $t \mapsto (a_0t^2 + a_1t + a_2, a_3t + a_4)$)

Polynomial Trajectories

polynomial trajectory $\phi_{\mathbf{a}}$ from time to state space in parameters \mathbf{a} (e.g., $t \mapsto (a_0t^2 + a_1t + a_2, a_3t + a_4)$)

$flow_m(\mathbf{x}, \mathbf{y}) \equiv$

$$\exists \mathbf{a} \exists \tau > 0 [traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \wedge constr_m(\phi_{\mathbf{a}}, \tau)].$$

Polynomial Trajectories

polynomial trajectory $\phi_{\mathbf{a}}$ from time to state space in parameters \mathbf{a} (e.g., $t \mapsto (a_0t^2 + a_1t + a_2, a_3t + a_4)$)

$flow_m(\mathbf{x}, \mathbf{y}) \equiv$

$$\exists \mathbf{a} \exists \tau > 0 [traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \wedge constr_m(\phi_{\mathbf{a}}, \tau)].$$

where $traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \equiv$

$$\phi_{\mathbf{a}}(0) = \mathbf{x} \wedge \phi_{\mathbf{a}}(\tau) = \mathbf{y},$$

Polynomial Trajectories

polynomial trajectory $\phi_{\mathbf{a}}$ from time to state space in parameters \mathbf{a} (e.g., $t \mapsto (a_0t^2 + a_1t + a_2, a_3t + a_4)$)

$flow_m(\mathbf{x}, \mathbf{y}) \equiv$

$$\exists \mathbf{a} \exists \tau > 0 [traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \wedge constr_m(\phi_{\mathbf{a}}, \tau)].$$

where $traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \equiv$

$$\phi_{\mathbf{a}}(0) = \mathbf{x} \wedge \phi_{\mathbf{a}}(\tau) = \mathbf{y},$$

and $constr_m(\phi_{\mathbf{a}}, \tau) \equiv$

$$\forall t \in [0, \tau] [Flow_m(\phi_{\mathbf{a}}(t), \dot{\phi}_{\mathbf{a}}(t))],$$

Polynomial Trajectories

polynomial trajectory $\phi_{\mathbf{a}}$ from time to state space in parameters \mathbf{a} (e.g., $t \mapsto (a_0t^2 + a_1t + a_2, a_3t + a_4)$)

$flow_m(\mathbf{x}, \mathbf{y}) \equiv$

$$\exists \mathbf{a} \exists \tau > 0 [traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \wedge constr_m(\phi_{\mathbf{a}}, \tau)].$$

where $traj(\mathbf{x}, \mathbf{y}, \phi_{\mathbf{a}}, \tau) \equiv$

$$\phi_{\mathbf{a}}(0) = \mathbf{x} \wedge \phi_{\mathbf{a}}(\tau) = \mathbf{y},$$

and $constr_m(\phi_{\mathbf{a}}, \tau) \equiv$

$$\forall t \in [0, \tau] [Flow_m(\phi_{\mathbf{a}}(t), \dot{\phi}_{\mathbf{a}}(t))],$$

according to Tarski: done!?

Linear Trajectories

$$\phi_{\mathbf{a}_1, \mathbf{a}_0}(t) = \mathbf{a}_1 t + \mathbf{a}_0$$

Linear Trajectories

$$\phi_{\mathbf{a}_1, \mathbf{a}_0}(t) = \mathbf{a}_1 t + \mathbf{a}_0$$

Then (eliminate $\exists \mathbf{a}$):

$$\exists \tau > 0 \forall t \in [0, \tau] \left[\text{Flow}_m \left(\frac{\mathbf{y} - \mathbf{x}}{\tau} t + \mathbf{x}, \frac{\mathbf{y} - \mathbf{x}}{\tau} \right) \right],$$

Linear Trajectories

$$\phi_{\mathbf{a}_1, \mathbf{a}_0}(t) = \mathbf{a}_1 t + \mathbf{a}_0$$

Then (eliminate $\exists \mathbf{a}$):

$$\exists \tau > 0 \forall t \in [0, \tau] \left[\text{Flow}_m\left(\frac{\mathbf{y} - \mathbf{x}}{\tau} t + \mathbf{x}, \frac{\mathbf{y} - \mathbf{x}}{\tau}\right) \right],$$

Convex flow constraint

$$\exists \tau > 0 \left[\text{Flow}_m\left(\mathbf{x}, \frac{\mathbf{y} - \mathbf{x}}{\tau}\right) \wedge \text{Flow}_m\left(\mathbf{y}, \frac{\mathbf{y} - \mathbf{x}}{\tau}\right) \right]$$

Linear Flow Constraints

$$\text{Flow}(\mathbf{x}, \dot{\mathbf{x}}) \equiv \underline{A}\mathbf{x} - \underline{\mathbf{b}} \leq \dot{\mathbf{x}} \leq \overline{A}\mathbf{x} + \overline{\mathbf{b}},$$

Linear Flow Constraints

$$\text{Flow}(\mathbf{x}, \dot{\mathbf{x}}) \equiv \underline{A}\mathbf{x} - \underline{\mathbf{b}} \leq \dot{\mathbf{x}} \leq \overline{A}\mathbf{x} + \overline{\mathbf{b}},$$

$$\mathbf{x} + \tau \underline{A}\mathbf{x} - \underline{\mathbf{b}} \leq \mathbf{y} \leq \mathbf{x} + \tau \overline{A}\mathbf{x} + \overline{\mathbf{b}} \wedge$$

$$\mathbf{y} - \tau \overline{A}\mathbf{y} - \underline{\mathbf{b}} \leq \mathbf{x} \leq \mathbf{y} - \tau \underline{A}\mathbf{y} + \overline{\mathbf{b}}$$

and equivalently

$$(I + \tau \underline{A})\mathbf{x} - \underline{\mathbf{b}} \leq \mathbf{y} \leq (I + \tau \overline{A})\mathbf{x} + \overline{\mathbf{b}} \wedge$$

$$(I - \tau \overline{A})\mathbf{y} - \underline{\mathbf{b}} \leq \mathbf{x} \leq (I - \tau \underline{A})\mathbf{y} + \overline{\mathbf{b}}$$

Example

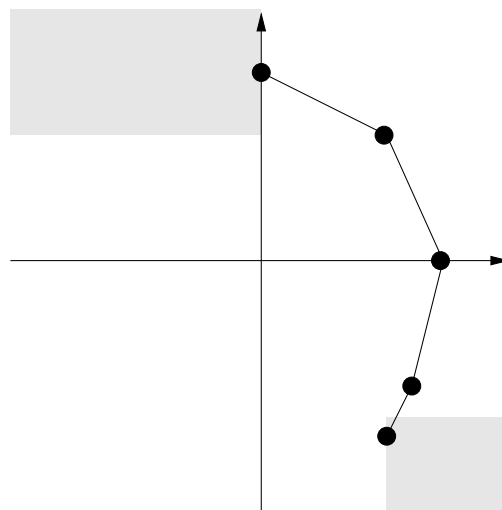
Computed with prototype implementation (linear trajectories, convex flow)

$$\begin{aligned}x_2 - 1 &\leq \dot{x}_1 \leq x_2 + 1 \\ -x_1 - 1 &\leq \dot{x}_2 \leq -x_1 + 1\end{aligned}$$

Init: $x_1 \leq 0, x_2 \geq 2$

Unsafe: $x_1 \geq 2, x_2 \leq -2.5$

State space $[-4, 4] \times [-4, 4]$.



Future Work

- improve verification engine . . .

Future Work

- improve verification engine . . .
- special, efficient methods for solving constraints

Future Work

- improve verification engine . . .
- special, efficient methods for solving constraints
- incrementality (wrt. abstraction refinement)

Future Work

- improve verification engine . . .
- special, efficient methods for solving constraints
- incrementality (wrt. abstraction refinement)
- hybrid systems with (partially) deterministic continuous evolutions

Conclusion

- abstractions can provide important information in the search for counter-examples
- instead of *search*, *solve* for counter-examples

Software: `http://hsolver.sourceforge.net`

Papers:

`http://www.mpi-sb.mpg.de/~ratschan/preprints.html`