

Recent Progress on the Integration of ODE Enclosure Methods with the iSAT Solver

Andreas Eggers¹ Nacim Ramdani² Nedialko S. Nedialkov³ Martin Fränzle¹

¹Carl von Ossietzky Universität, Oldenburg, Germany
{eggers|fraenzle}@informatik.uni-oldenburg.de

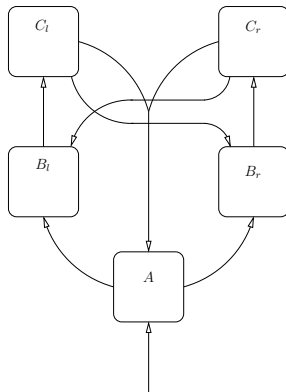
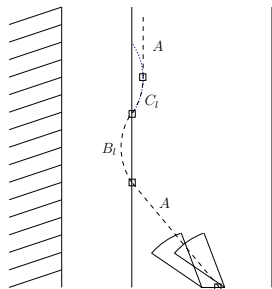
²Université d'Orléans, PRISME, 18020 Bourges, France
nacim.ramdani@bourges.univ-orleans.fr

³McMaster University, Hamilton, Ontario, Canada
nedialk@mcmaster.ca

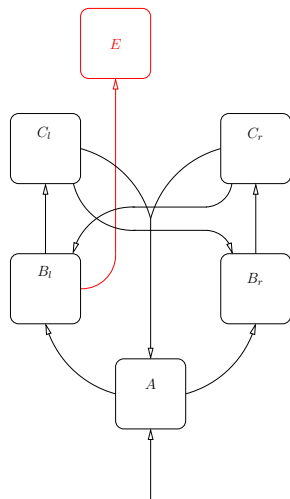
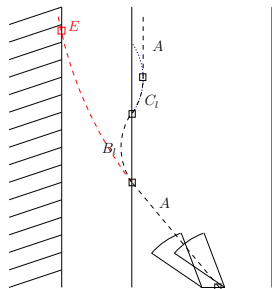
SWIM 2011, Bourges, France, 2011-06-14

Motivation

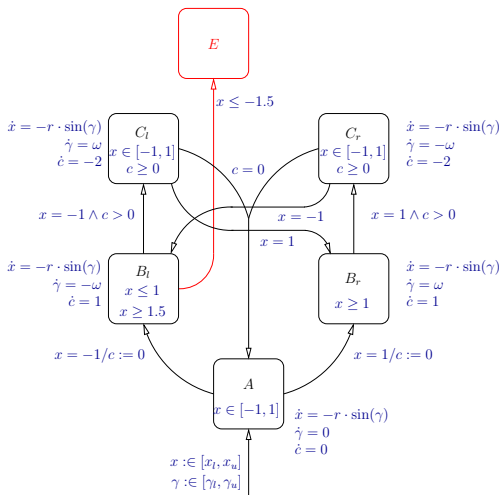
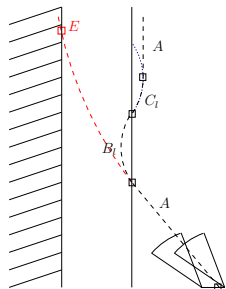
E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald, Verification of hybrid systems based on counterexample-guided abstraction refinement (2003). In: Proceedings of TACAS'03, LNCS, vol 2619, pp. 192-207



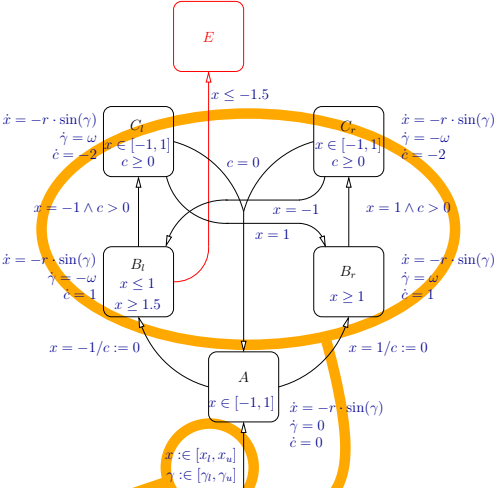
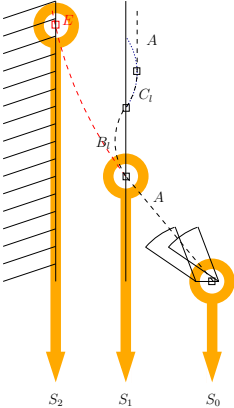
Motivation



Motivation

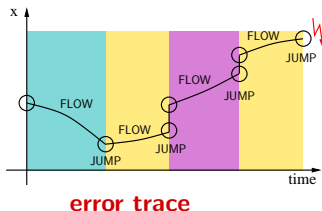
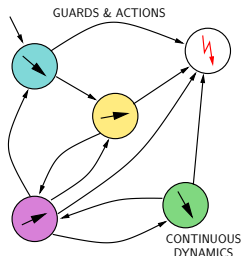


Motivation



$$\Phi = \bigwedge \begin{matrix} INIT(S_0) \\ TRANS(S_0, S_1) \\ TRANS(S_1, S_2) \\ TARGET(S_2) = E \end{matrix}$$

Error Traces and Overapproximation



- ▶ Can only show absence of k -bounded error traces if *all* trajectories have been analyzed
 - ▶ *All* original trajectories must have corresponding solutions of the predicative encoding (allowing spurious solutions without exact counterpart)
 - ▶ The solver *must never* discard any solution of that formula
- ⇒ Good reason to use interval methods

Outline

Motivation

Hybrid systems' trajectories as solutions of constraint systems

Rigorous analysis involves making sure not to lose any trajectories

The Extended iSAT Algorithm

Problem statement

Satisfiability

The Core iSAT Algorithm

Deduction for ODE Constraints

Embedded ODE Enclosure Methods

VNODE-LP

Bracketing Systems

Deducing Trajectory Directions

Experiments

The Two Tank System

Reachability

Stabilization

Conclusions & Future Work

An Example Input File

Goldsztejn, A., Mullier, O., Eveillard, D., Hosobe, H.: Including ordinary differential equations based constraints in the standard CP framework. In: Cohen, D. (ed.) Principles and Practice of Constraint Programming - CP 2010, Lecture Notes in Computer Science, vol. 6308, pp. 221–235. Springer Berlin / Heidelberg (2010)

DECL

```
float [-1, 1] ax, ay, bx, by;  
float [-10, 10] x;  
float [-10, 10] y;  
float [0, 10] time;  
float [1, 1] delta_time;
```

INIT

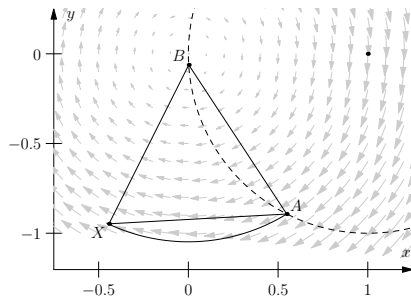
```
-- A and B must lie on the circle ,  
-- i.e. have distance 1 to (1, 0).  
(ay - 0)^2 + (ax - 1)^2 = 1^2;  
(by - 0)^2 + (bx - 1)^2 = 1^2;  
-- A and B must be distinct points.  
ax != bx or ay != by;  
-- Trajectory must start in A.  
x = ax; y = ay;  
time = 0;
```

TRANS

```
-- A and B stay the same.  
ax' = ax; ay' = ay;  
bx' = bx; by' = by;  
-- Trajectory.  
(d.x / d.time = y);  
(d.y / d.time = -x);  
time' = time + delta_time;
```

TARGET

```
-- Want an equilateral triangle , i.e.  
-- the distances between (ax,ay),  
-- (bx,by), and (x,y) must be equal.  
(ay - by)^2 + (ax - bx)^2  
= (ax - x)^2 + (ay - y)^2;  
(ay - by)^2 + (ax - bx)^2  
= (bx - x)^2 + (by - y)^2;
```



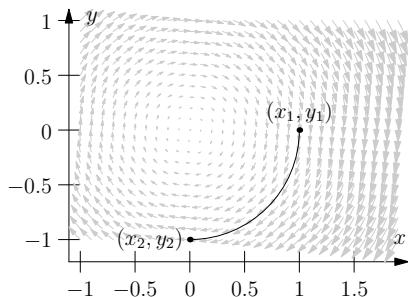
Problem Statement

- ▶ Boolean-, real-, and integer-valued variables with bounded domains
- ▶ Quantifier-free Boolean combination of
 - ▶ Simple bounds, e.g. $x \leq 3.2$
 - ▶ Arithmetic constraints, e.g. $z = \sin(y)$
 - ▶ Time invariant ordinary differential equations, e.g.
$$\dot{x} = 2.4 \cdot x - y^2$$
- ▶ Bounded Model Checking (BMC) formula structure:
$$\Phi = \text{init}[0] \wedge \text{trans}[0, 1] \wedge \dots \wedge \text{trans}[k - 1, k] \wedge \text{target}[k]$$
- ▶ ODEs only in transition system
- ▶ Rewritten into conjunction of clauses which are disjunctions of these atoms

Goal: Find a satisfying valuation for Φ .

Satisfiability

- ▶ Point-valued satisfaction: standard for arithmetic constraints and simple bounds, e.g. $(x = 6, y = 3)$ satisfy $x = 2y$
- ▶ Definitionally-closed systems of ODEs, e.g. $\dot{x} = -y, \dot{y} = x$ satisfied by valuation $((x_1 = 1, y_1 = 0), (x_2 = 0, y_2 = -1), \text{delta_time} = \pi/2)$ with $(x_1, y_1), (x_2, y_2)$ successive BMC instances of (x, y) and duration delta_time :



The Core iSAT Algorithm

Generalization of DPLL solving
manipulating **interval bounds**

$$x \in [3, 7], \quad y \in [-2, 25]$$

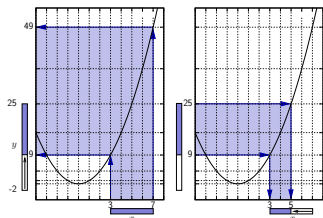
Deductions:

prune off definite non-solutions

▶ Unit propagation:

$$\dots \wedge (x > 8 \vee y = x^2) \wedge \dots$$

▶ Interval constraint propagation:



$$y = x^2 \wedge x \geq 3 \Rightarrow y \geq 9$$

$$y = x^2 \wedge y \leq 25 \Rightarrow x \leq 5$$

Decisions:

Split interval (e.g. at its midpoint),
propagate resulting bound

Conflict-driven Learning:

- ▶ Deduction can yield empty box
- ▶ Learn reasons from implication graph (conflict clause)
- ▶ Jump back undoing decisions

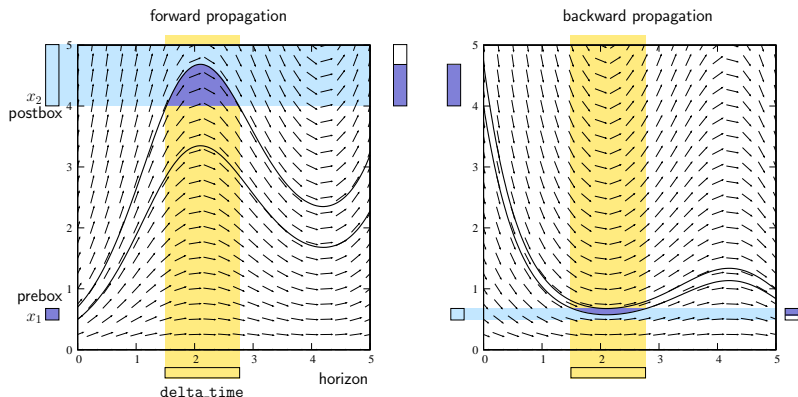
Termination:

Stop search when

- ▶ unresolvable conflict is found or
- ▶ reasonably small conflict-free box found

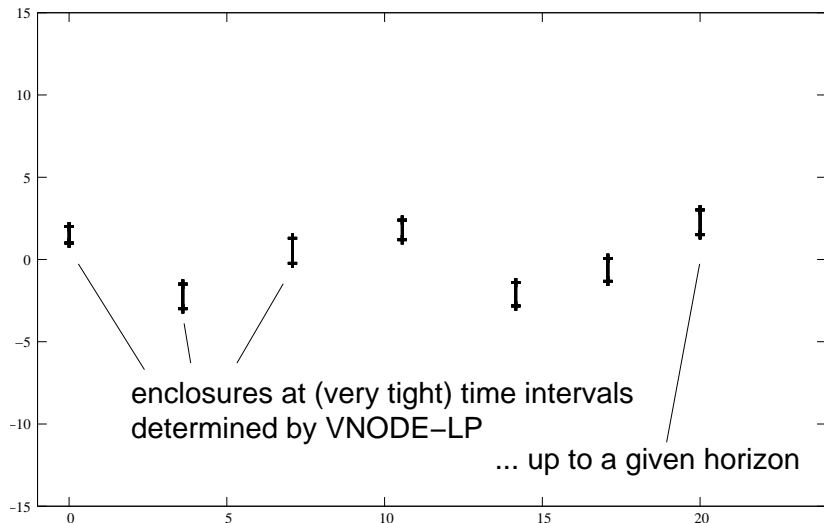
Use **optimizations from propositional SAT** (backjumps, two-watched literal scheme, isomorphy inference, restarts, ...)

Deduction for ODE Constraints

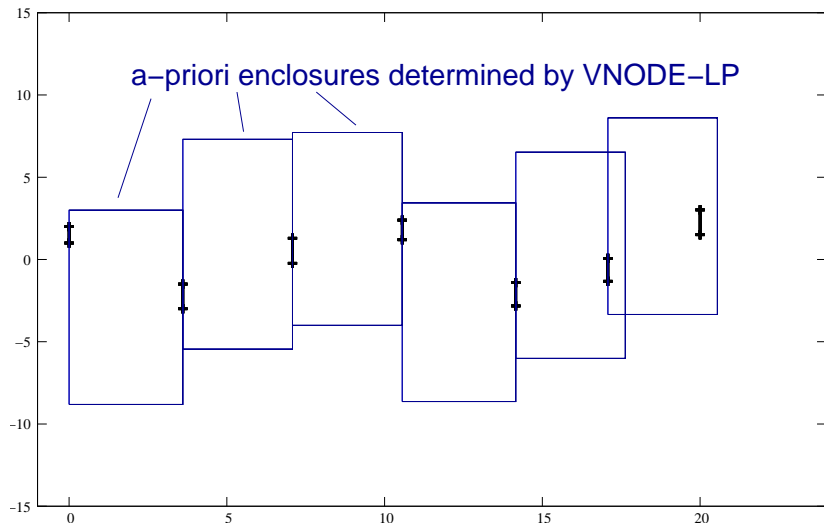


- ▶ Deduce tighter bounds for pre- and postbox and delta_time
- ▶ Add as clauses to learn deductions persistently
- ▶ Cache partial information and avoid repeating deductions

VNODE-LP as an Enclosure Method



VNODE-LP as an Enclosure Method



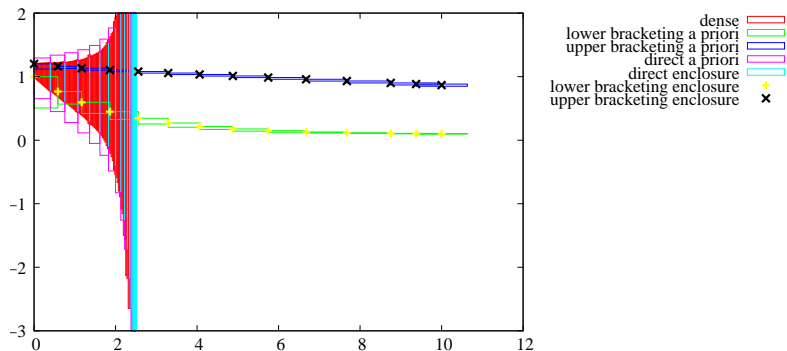
VNODE-LP as an Enclosure Method

- ▶ VNODE-LP: safe interval enclosures for initial value problems of ODEs
- ▶ Output: tight enclosures at end of step, a-priori enclosures covering timespan between
- ▶ Use as a propagator:
 - ▶ Generate dense enclosures over interval valuation of `delta_time`: tightening postbox (or prebox using negated system in backward propagation)
 - ▶ Empty intersection of enclosure with postbox: detect conflict or tighten valuation of `delta_time`
- ▶ Current approach: re-evaluate extreme parts of enclosure with refined stepsize → expensive
- ▶ Future: use new interpolation feature in VNODE-LP for faster & safe intermediate enclosures

Bracketing Systems

- ▶ Direct VNODE-LP enclosures may diverge quickly for large initial domains
- ▶ Evaluate signs of partial derivatives of ODE's right-hand side over current enclosure
- ▶ If all relevant entries each strictly positive / negative, proceed
- ▶ Using Müller's theorem, generate a bracketing system:
replace original variables by upper and lower bracketing variables depending on signs in Jacobian
- ▶ Enclose bracketing system using VNODE-LP: **twice the dimensionality but point-valued initial conditions**
- ▶ Re-evaluate Jacobian, check validity of signs (a-posteriori validation)

Comparison: Direct vs. Bracketing

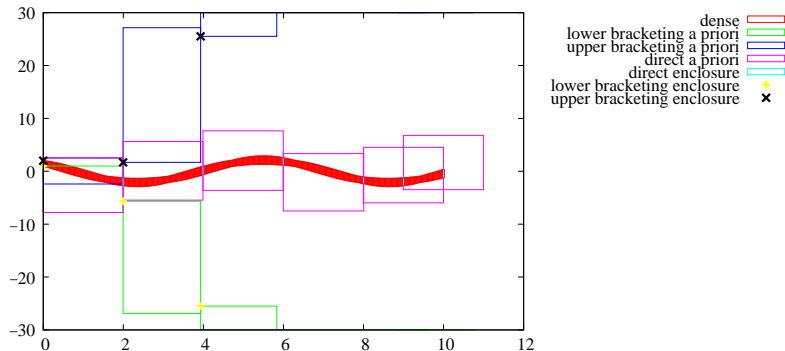


x dimension of $\dot{x} = -p_4x - \frac{p_1x}{1 + p_2y} + p_3y + 0.1$

$$\dot{y} = p_4x - p_3y$$

$x(0) \in [1, 1.2]$, $y(0) \in [0.8, 1]$, $p_1 \in [0.8, 1]$, $p_2 \in [1.0, 1.2]$, $p_3 \in [0.3, 0.5]$,
and $p_4 \in [0.20, 0.25]$.

Comparison: Direct vs. Bracketing

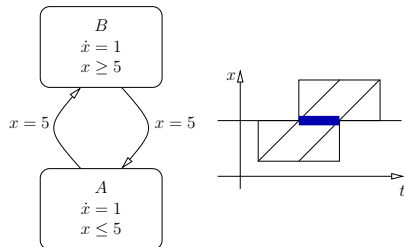


x dimension of $\dot{x} = y, \dot{y} = -x$,
 $x(0), y(0) \in [1, 2]$.

► Complementary strengths

⇒ **Intersect both enclosures for tighter result.**

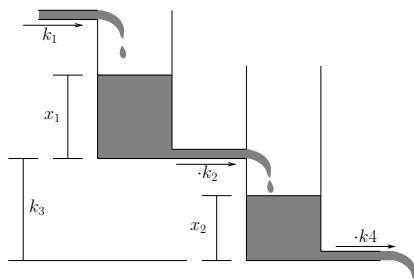
Deducing Trajectory Directions



- ▶ $x = 5$ included even in tightest possible enclosure after switch
- ▶ Can thus *oscillate* between modes A and B
- ▶ Problematic for certain analysis goals (∞ -traces)
- ▶ Solution: **deduce direction of trajectory at its beginning**, here, e.g. $\text{delta_time} > 0 \Rightarrow x' > x$
- ▶ In general: evaluate ODE's right-hand side over prefix of trajectory, e.g. as long as strictly positive: variable grows

The Two Tank System

Stursberg, O., Kowalewski, S., Hoffmann, I., Preußig, J.: Comparing timed and hybrid automata as approximations of continuous systems. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S. (eds.) Hybrid Systems IV, Lecture Notes in Computer Science, vol. 1273, pp. 361–377. Springer Berlin / Heidelberg (1997)



For $x_2 > k_3$:

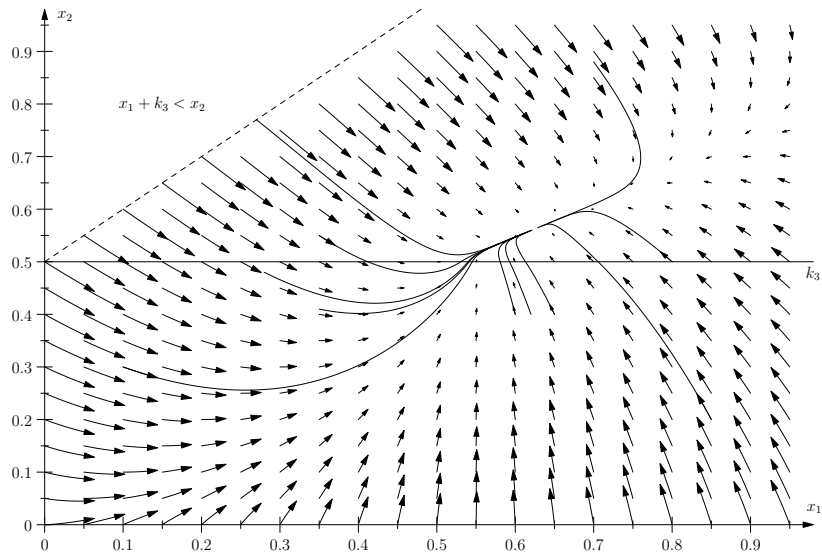
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} k_1 - k_2\sqrt{x_1 - x_2 + k_3} \\ k_2\sqrt{x_1 - x_2 + k_3} - k_4\sqrt{x_2} \end{pmatrix}$$

For $x_2 \leq k_3$:

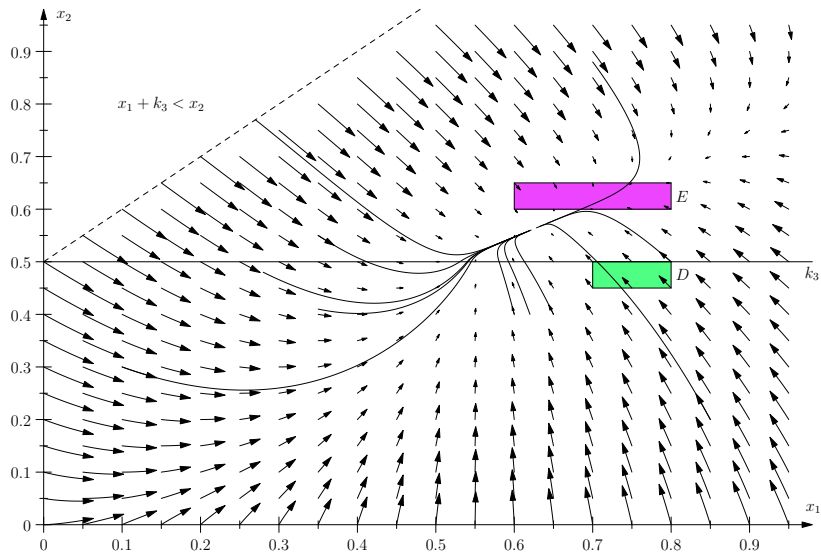
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} k_1 - k_2\sqrt{x_1} \\ k_2\sqrt{x_1} - k_4\sqrt{x_2} \end{pmatrix}$$

$$k_1 = 0.75, k_2 = 1, k_3 = 0.5, k_4 = 1$$

Simulated Dynamics

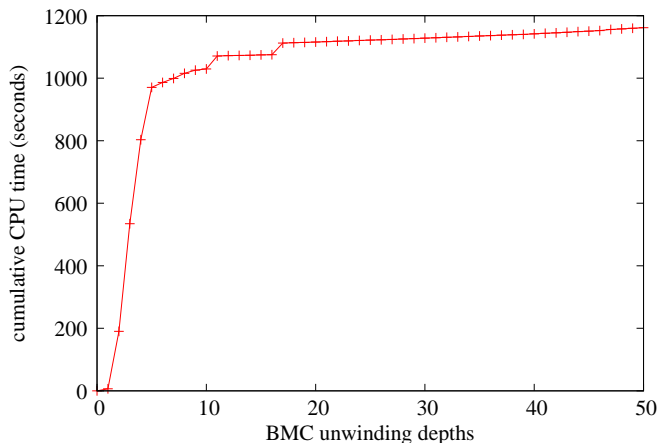


Bounded reachability

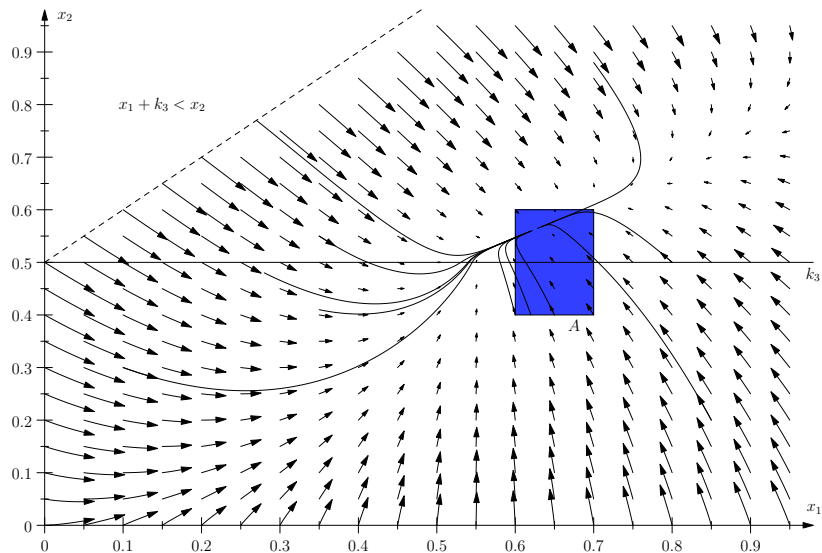


Bounded reachability

- ▶ Runtime for proving: **E cannot be reached from D** in k steps of at most 10 time units length each and within at most 100 time units in total. (2.4GHz AMD Opteron)



Stabilization



Stabilization

- ▶ Use **inductive argument** to show: trajectories can only leave A for limited time, then stay in A forever
 - ▶ Guess time τ (length of the prefix)
 - ▶ Set TARGET of BMC formula Φ :
 $((\text{time} < 2\tau) \vee (\text{trajectory outside } A \wedge \text{time} \in [\tau, 2\tau]))$
 - ▶ Must forbid trajectories that can stutter superfluously
 - ▶ Find first unwinding depth k such that Φ unsatisfiable
 - ▶ If found: proven that all k -length trajectories are longer than 2τ or stay inside A for all time $\in [\tau, 2\tau]$
 - ▶ Induction:
 - ▶ **Time invariance**: take $\vec{x}(\tau)$ on trajectory as new starting point
 - ▶ Already proven above: for $\tau + [\tau, 2\tau]$ trajectory stays inside A
 - ▶ Trajectory is also second half of a trajectory already shown to be in A for $[\tau, 2\tau]$
- ⇒ Trajectory stays in A forever after having left it for at most A time units.

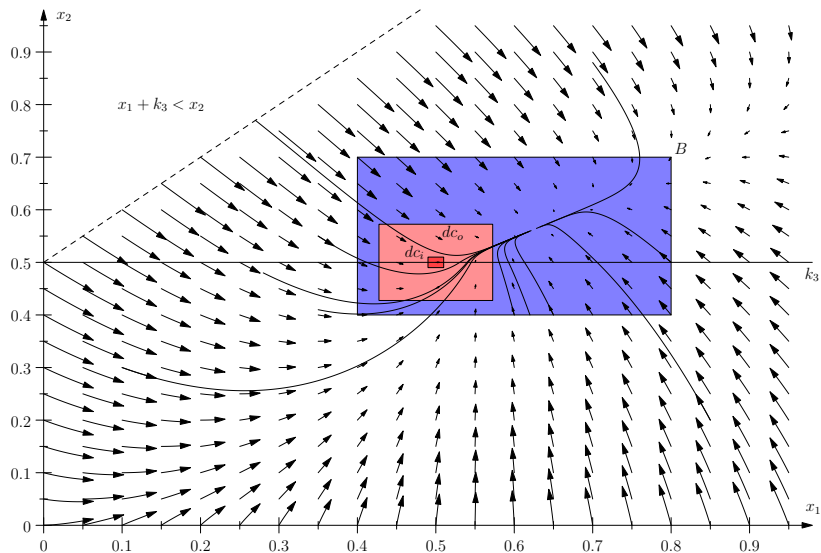
Stabilization

Solver results ($\tau = 5$) and runtimes in seconds (2.4 GHz AMD Opteron):

depth	all	no bracketing	no direct	no direction
1	unknown, 111.9	unknown, 42.0	unknown, 61.5	unknown, 111.5
2	unknown, 467.5	unknown, 981.0	unknown, 346.3	unknown, 342.0
3	unsat , 674.0	unsat , 5011.6	unsat , 404.2	unknown, 478.8
4	unsat , 812.1	unsat , 1995.1	unsat , 499.1	unknown, 547.5
5	unsat , 986.0	unsat , 2432.0	unsat , 601.1	unknown, 682.4
6	unsat , 1126.1	unsat , 3303.4	unsat , 705.0	unknown, 834.2
7	unsat , 1277.2	unsat , 2486.8	unsat , 803.7	unknown, 982.5
8	unsat , 1451.4	unsat , 5273.3	unsat , 890.8	unknown, 1115.7
9	unsat , 1584.6	unsat , 4905.2	unsat , 966.5	unknown, 1235.8
10	unsat , 1706.6	unsat , 6396.1	unsat , 1053.2	unknown, 1356.0

- ▶ Direction deduction essential for this kind of proof
- ▶ Bracketing system with direction deduction performs best

Stabilization & Artificial Hysteresis



Stabilization & Artificial Hysteresis

- ▶ Point $P = (0.5, 0.5)$ on switching surface has $\dot{x}_2 = 0$
- ⇒ Direction deduction **cannot deduce that surface must be left immediately**
- ▶ *But:* P well inside B
- ▶ Add “don’t care mode” around P : when trajectory reaches inner border dc_i , may jump after arbitrary amount of time to outer border $dc_o \Rightarrow$ **artificial hysteresis**
- ▶ Being inside dc_i is forbidden
- ⇒ Stuttering at P becomes impossible, movement back from dc_o to dc_i consumes time (ε -separation)
- ⇒ Proof scheme becomes applicable

Stabilization & Artificial Hysteresis

Solver results ($\tau = 0.0625$) and runtimes in seconds (2.4 GHz AMD Opteron):

depth	all	no bracketing	no direct	no direction
1	unknown, 17.7	unknown, 9.4	unknown, 12.9	unknown, 15.4
2	unknown, 163.9	unknown, 57.9	unknown, 81.9	unknown, 157.4
3	unknown, 198.9	unknown, 71.8	unknown, 126.9	unknown, 202.4
4	unknown, 666.6	unknown, 193.6	unknown, 146.7	unknown, 206.9
5	unsat , 2334.2	unsat , 3270.3	unknown, 183.4	unknown, 283.6
6	unsat , 4615.6	unsat , 1441.2	unknown, 182.2	unknown, 122.0
7	unsat , 2967.1	unknown, 1934.7	unknown, 144.1	unknown, 123.9
8	unsat , 2559.0	unsat , 2953.0	unknown, 201.6	unknown, 123.6
9	unsat , 2184.1	unsat , 4121.2	unknown, 135.2	unknown, 127.2
10	unsat , 5541.6	unsat , 7717.3	unknown, 272.5	unknown, 127.6

- ▶ Direct method and direction deduction required
- ▶ Combination of all methods more stable (one less spurious unsat) and partially more efficient

Conclusions & Future Work

- ▶ **Combine different enclosure methods** and simple trajectory information with iSAT
- ▶ Application to a non-linear hybrid system, **bounded and unbounded properties**
- ▶ Currently missing: **flow invariants**, i.e. overapproximation may allow mode's invariant region to be left between entering and exiting – may yield spurious solutions
- ▶ Competitive work: interval Newton proofs can **guarantee existence** of solution – our iSAT extension cannot do this yet
- ▶ Significant acceleration expected from using **continuous-interpolant**-like feature in next VNODE-LP version
- ▶ Currently only using naïve method for building bracketing system – could **combine direct and bracketing enclosures more tightly**