SWIM 2012

Recent Improvements of iSAT-ODE

Andreas Eggers[⋆] Nacim Ramdani[∗] Nedialko S. Nedialkov[◦]
Martin Fränzle[⋆]

[⋆] Carl von Ossietzky Universität Oldenburg, Transregional Collaborative Research Center AVACS, Germany

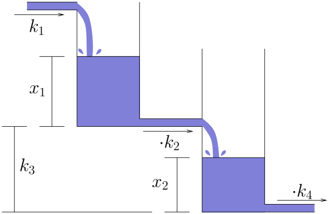
[∗] Université d'Orléans, PRISME, 63 av. de Lattre de Tassigny, 18020 Bourges, France

[◦] McMaster University, Hamilton, Ontario, Canada

2012-06-05

The Two Tank System

[Stursberg, Kowalewski, Hoffmann, and Preußig, 1997]



For $x_2 > k_3$:

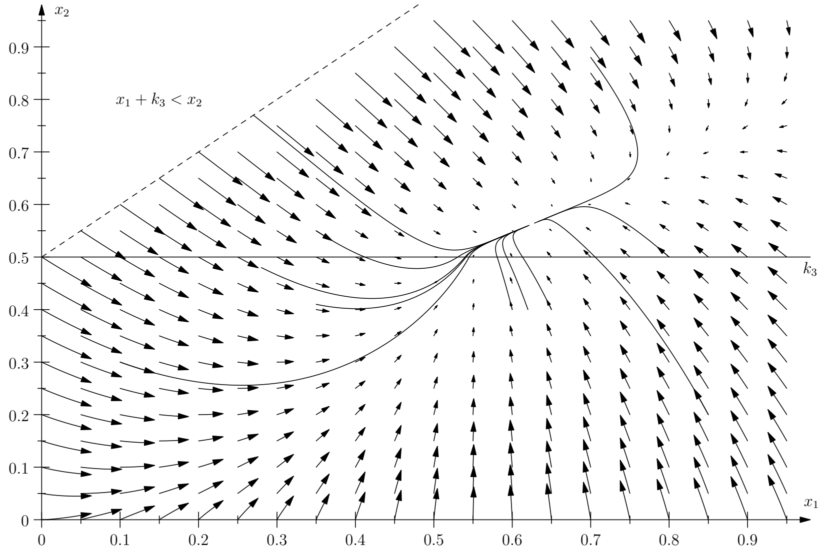
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} k_1 - k_2\sqrt{x_1 - x_2 + k_3} \\ k_2\sqrt{x_1 - x_2 + k_3} - k_4\sqrt{x_2} \end{pmatrix}$$

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

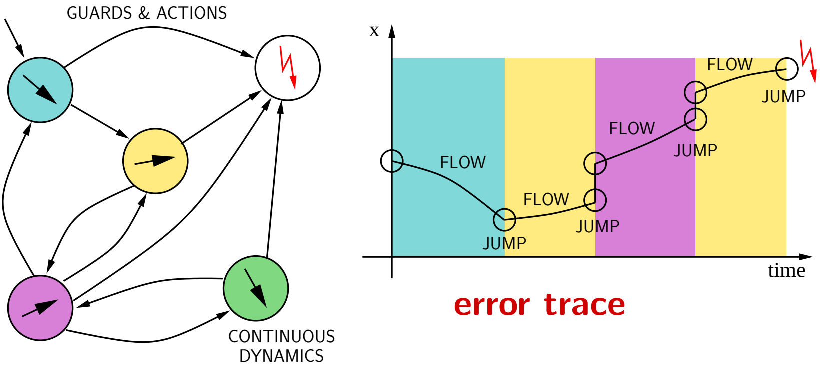


Characteristics of Hybrid Systems

- **Continuous dynamics** often described by ordinary differential equations (ODEs)
- Discrete components often described by **modes** of an automaton
- Active discrete mode governs continuous dynamics
- Change of continuous components may trigger mode switches (**guards** and **jumps**)
- Mode switches may change continuous components abruptly (**actions**)
- Traditional formal model: **Hybrid Automata**

Bounded Model Checking

[Biere, Cimatti, Clarke, and Zhu, 1999], [Audemard, Bozzano, Cimatti, and Sebastiani, 2005], [Fränzle and Herde, 2005]



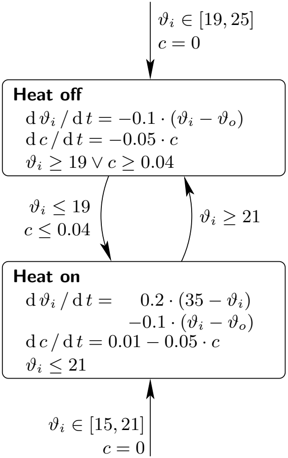
error trace

Model-checking question:

Can the system reach an unsafe state within k (discrete or continuous) transition steps **?**

Bounded Model Checking and Constraint Solving

[Audemard, Bozzano, Cimatti, and Sebastiani, 2005], [Fränzle and Herde, 2005], [Eggers, Fränzle, and Herde, 2008]



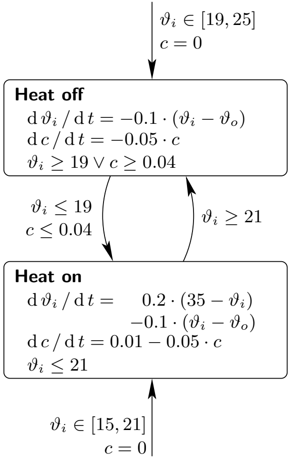
$$
\begin{aligned}
&init = \\
&\qquad -10 \leq \vartheta_o \leq 20 \land c = 0 \\
&\qquad \land \begin{pmatrix} & 19 \leq \vartheta_i \leq 25 \land \neg on \\ \lor & 15 \leq \vartheta_i \leq 21 \land on \end{pmatrix} \\
&trans = \\
&\qquad (\quad \neg on \land on' \land \vartheta_i \leq 19 \land c \leq 0.04 \\
&\qquad \land \quad \vartheta_i' = \vartheta_i \land \vartheta_o' = \vartheta_o \land c' = c) \\
&\quad \lor \ (\quad on \land \neg on' \land \vartheta_i \geq 21 \\
&\qquad \land \quad \vartheta_i' = \vartheta_i \land \vartheta_o' = \vartheta_o \land c' = c) \\
&\quad \lor \ (\quad \neg on \land \neg on' \\
&\qquad \land \quad \tfrac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = -0.1(\vartheta_i - \vartheta_o) \\
&\qquad \land \quad \tfrac{\mathrm{d}c}{\mathrm{d}t} = -0.05c \\
&\qquad \land \quad (\vartheta_i' \geq 19 \lor c' \geq 0.04) \land \vartheta_o' = \vartheta_o) \\
&\quad \lor \ (\quad on \land on' \\
&\qquad \land \quad \tfrac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = 0.2 \cdot 35 - 0.3\vartheta_i + 0.1\vartheta_o \\
&\qquad \land \quad \tfrac{\mathrm{d}c}{\mathrm{d}t} = 0.01 - 0.05c \\
&\qquad \land \quad \vartheta_i' \leq 21 \land \vartheta_o' = \vartheta_o) \\
&target = \\
&\qquad (c > 0.1)
\end{aligned}
$$

Bounded Model Checking (BMC):

Are there any trajectories leading from an inital to an unsafe state in k steps?

Bounded Model Checking and Constraint Solving

[Audemard, Bozzano, Cimatti, and Sebastiani, 2005], [Fränzle and Herde, 2005], [Eggers, Fränzle, and Herde, 2008]



$$
\begin{aligned}
&\mathit{init} = \\
&\qquad -10 \leq \vartheta_o \leq 20 \wedge c = 0 \\
&\qquad \wedge \begin{pmatrix} & 19 \leq \vartheta_i \leq 25 \wedge \neg \mathit{on} \\ \vee & 15 \leq \vartheta_i \leq 21 \wedge \mathit{on} \end{pmatrix} \\
&\mathit{trans} = \\
&\qquad\quad (\ \ \neg \mathit{on} \wedge \mathit{on}' \wedge \vartheta_i \leq 19 \wedge c \leq 0.04 \\
&\qquad\quad \wedge\ \ \vartheta_i' = \vartheta_i \wedge \vartheta_o' = \vartheta_o \wedge c' = c) \\
&\qquad \vee\ (\ \ \mathit{on} \wedge \neg \mathit{on}' \wedge \vartheta_i \geq 21 \\
&\qquad\quad \wedge\ \ \vartheta_i' = \vartheta_i \wedge \vartheta_o' = \vartheta_o \wedge c' = c) \\
&\qquad \vee\ (\ \ \neg \mathit{on} \wedge \neg \mathit{on}' \\
&\qquad\quad \wedge\ \ \tfrac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = -0.1(\vartheta_i - \vartheta_o) \\
&\qquad\quad \wedge\ \ \tfrac{\mathrm{d}c}{\mathrm{d}t} = -0.05c \\
&\qquad\quad \wedge\ \ (\vartheta_i' \geq 19 \vee c' \geq 0.04) \wedge \vartheta_o' = \vartheta_o) \\
&\qquad \vee\ (\ \ \mathit{on} \wedge \mathit{on}' \\
&\qquad\quad \wedge\ \ \tfrac{\mathrm{d}\vartheta_i}{\mathrm{d}t} = 0.2 \cdot 35 - 0.3\vartheta_i + 0.1\vartheta_o \\
&\qquad\quad \wedge\ \ \tfrac{\mathrm{d}c}{\mathrm{d}t} = 0.01 - 0.05c \\
&\qquad\quad \wedge\ \ \vartheta_i' \leq 21 \wedge \vartheta_o' = \vartheta_o) \\
&\mathit{target} = \\
&\qquad (c > 0.1)
\end{aligned}
$$

Bounded Model Checking (BMC): Check satisfiability of SMT formula

$$\Phi_k := \mathit{init}[0] \wedge \mathit{trans}[0,1] \wedge \cdots \wedge \mathit{trans}[k-1,k] \wedge \mathit{target}[k]$$

Satisfiability Modulo ODE

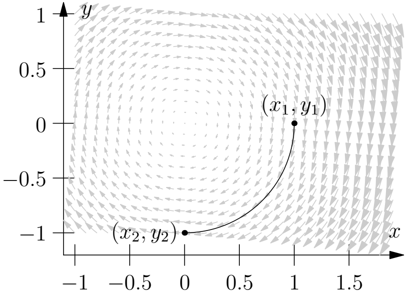
[Eggers, Fränzle, and Herde, 2008], [Ishii, Ueda, and Hosobe, 2011]

- **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. $z = \sin(y)$
  - Sufficiently smooth, time invariant **ordinary differential equations (ODEs)**, e.g. $\dot{x} = 2.4 \cdot x - y^2$
- Bounded Model Checking (BMC) formula structure: $\Phi = init[0] \land trans[0,1] \land \cdots \land trans[k-1,k] \land target[k]$
- ODEs only in transition system

Goal: Find a satisfying valuation for Φ or prove its unsatisfiability.

Satisfiability of SAT Modulo ODE Formulae

- Point-valued satisfaction: **standard for arithmetic constraints and simple bounds**, e.g. point $(x = 6, y = 3)$ satisfies $x = 2y$
- Definitionally-closed systems of **ODEs**, e.g. $\dot{x} = -y, \dot{y} = x$ satisfied by valuation $((x_1 = 1, y_1 = 0), (x_2 = 0, y_2 = -1),$ `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

[Fränzle, Herde, Ratschan, Schubert, and Teige, 2007]

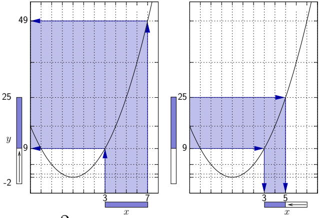
Generalization of DPLL/CDCL
solving manipulating **interval bounds**

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

Deductions:

prune off definite non-solutions

- Unit propagation:
  $\cdots \wedge (x > 8 \ \vee \ \underline{y = x^2}) \wedge \cdots$
- 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:

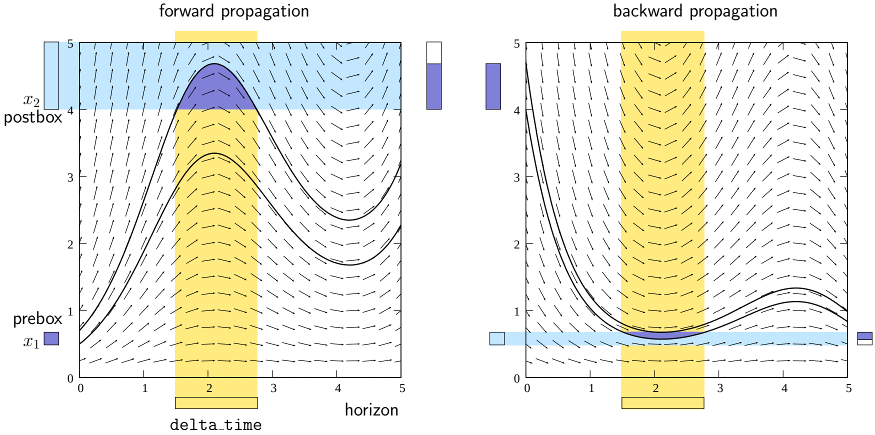
Stop search when

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

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

ODE Enclosures as Propagators

[Eggers, Fränzle, and Herde, 2008]



ODE Enclosure Problem

Given a system of (sufficiently-smooth, first order, time-invariant, Lipschitz-continuous) ordinary differential equations (ODEs), we want to **enclose all trajectories** emerging from a set of starting points over a limited temporal horizon.

$$\frac{d\vec{x}}{dt}(t) = \vec{f}\,(\vec{x}(t))\,, \qquad \vec{x}(0) \in \begin{pmatrix} [\underline{x_1(0)}, \overline{x_1(0)}] \\ \vdots \\ [\underline{x_n(0)}, \overline{x_n(0)}] \end{pmatrix}$$

Safely enclose all $\vec{x}(t)$ over $t \in [0, horizon]$.

- Use VNODE-LP to obtain such enclosures.

Beyond Taylor Series: VNODE-LP

[Nedialkov and Jackson, 1999], [Nedialkov, Jackson, and Pryce, 2001], [Nedialkov, 2006]

- **Hermite-Obreschkoff method**: generalization of Taylor series
- VNODE-LP:
  - Use **High-Order Enclosure (HOE)** to obtain a-priori enclosure of ODE
  - Use Interval Taylor Series (ITS) with coordinate transformation and QR-method as **predictor**
  - Use Interval Hermite-Obreschkoff (IHO) method as **corrector**
- IHO allows **larger stepsize** than ITS (ITS becomes numerically unstable for smaller stepsizes than IHO)
- **Local error** for nonlinear ODEs **much lower** for IHO than for ITS

Beyond Taylor Series: VNODE-LP

[Nedialkov and Jackson, 1999], [Nedialkov, Jackson, and Pryce, 2001], [Nedialkov, 2006]

- **Hermite-Obreschkoff method**: generalization of Taylor series
- VNODE-LP:
  - Use **High-Order Enclosure (HOE)** to obtain a-priori enclosure of ODE
  - Use Interval Taylor Series (ITS) with coordinate transformation and QR-method as **predictor**
  - Use Interval Hermite-Obreschkoff (IHO) method as **corrector**
- IHO allows **larger stepsize** than ITS (ITS becomes numerically unstable for smaller stepsizes than IHO)
- **Local error** for nonlinear ODEs **much lower** for IHO than for ITS

Efficiently Refine the VNODE-LP Enclosures

- Use Interval Taylor Series (ITS) with coordinate transformation and QR-method as **predictor**
  - Truncated Taylor series for midpoint of enclosure
  - Enclosure of the Lagrange remainder term over a-priori enclosure
  - Taylor coefficients for the variational equation (computing effect on parallelepiped and QR-orthogonalized representations)
  - Computed using Automatic Differentiation
  - **When extracting all components, can scale them for arbitrary interval subrange of the computed step**

⇒ Get a more efficient way to compute refined enclosures for substeps: requires only scaling of stepsize and recomputing ITS, i.e. no need to:

- reinitialize solver
- recompute a-priori enclosure
- recompute derivatives
- recompute coordinate transformations, and
- recompute IHO

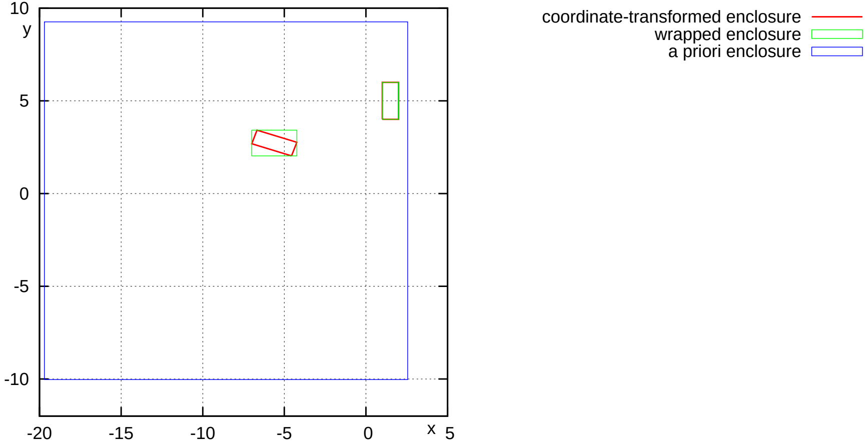
Efficiently Refine the VNODE-LP Enclosures

- Use Interval Taylor Series (ITS) with coordinate transformation and QR-method as **predictor**
  - Truncated Taylor series for midpoint of enclosure
  - Enclosure of the Lagrange remainder term over a-priori enclosure
  - Taylor coefficients for the variational equation (computing effect on parallelepiped and QR-orthogonalized representations)
  - Computed using Automatic Differentiation
  - **When extracting all components, can scale them for arbitrary interval subrange of the computed step**

$\Rightarrow$ Get a more efficient way to compute refined enclosures for substeps: requires only scaling of stepsize and recomputing ITS, i.e. no need to:

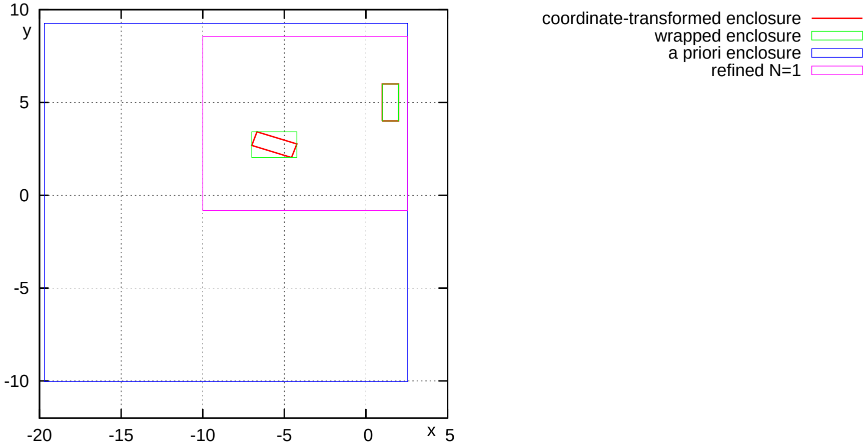
- reinitialize solver
- recompute a-priori enclosure
- recompute derivatives
- recompute coordinate transformations, and
- recompute IHO

VNODE-LP Output and Re-evaluated ITS



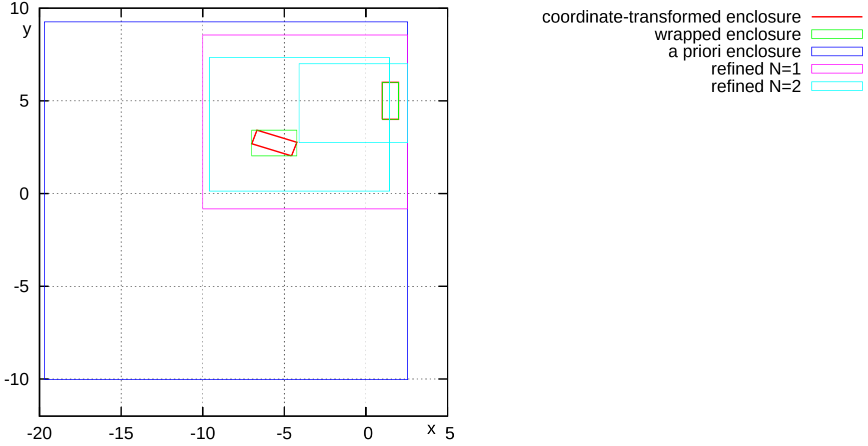
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



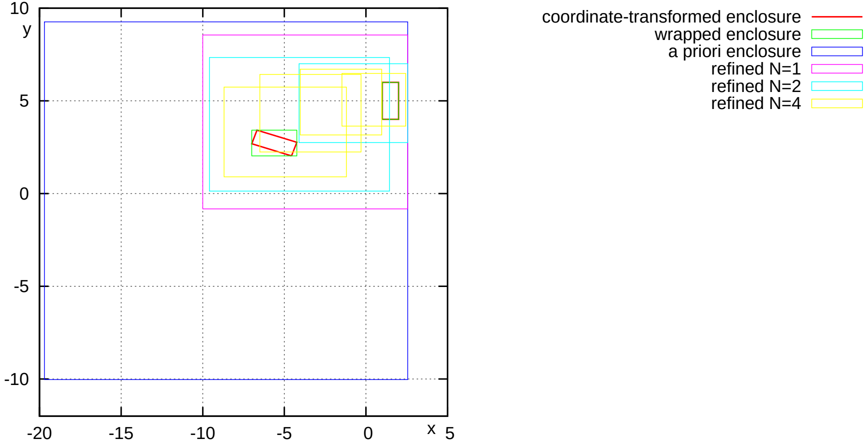
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1,2], \quad y_0 \in [4,6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



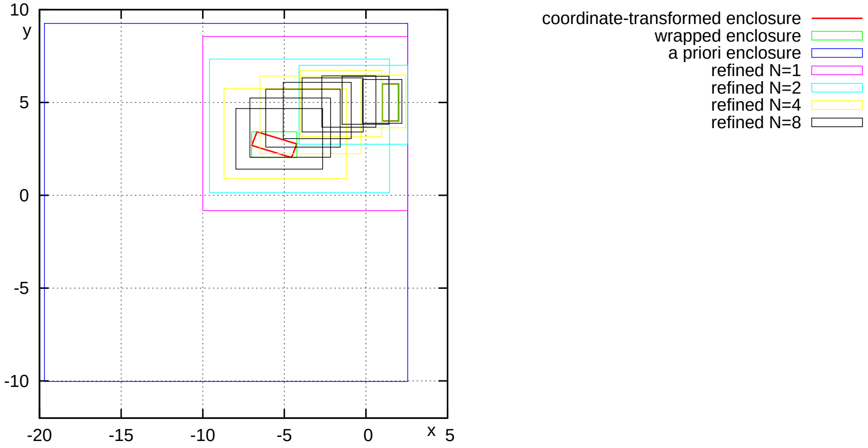
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



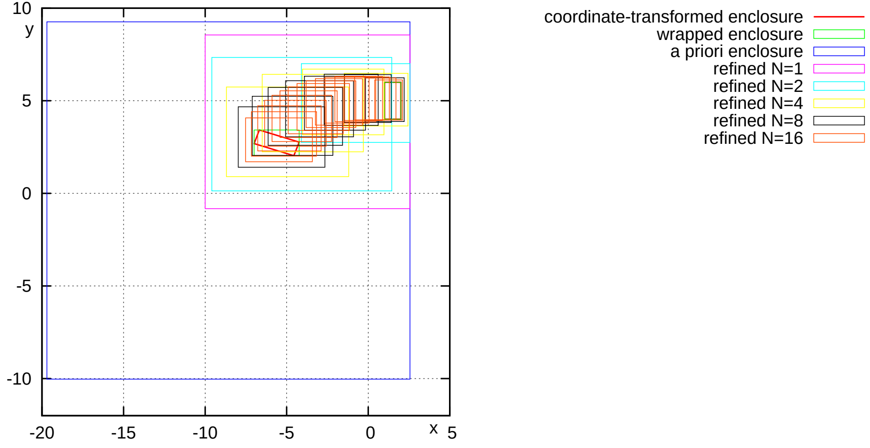
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



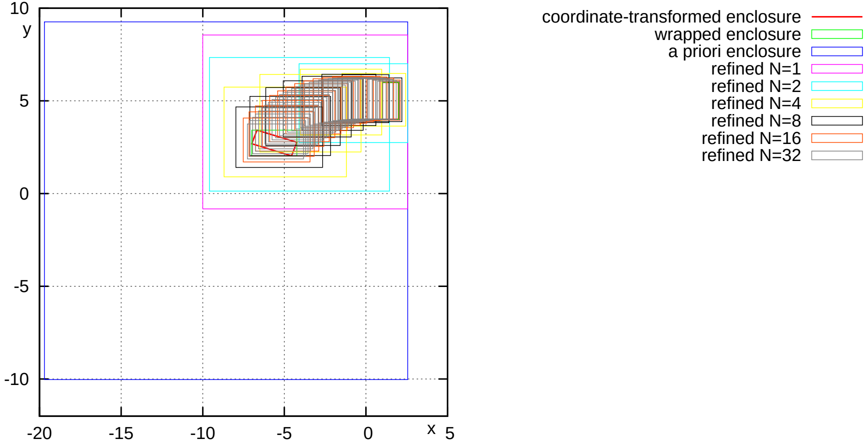
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



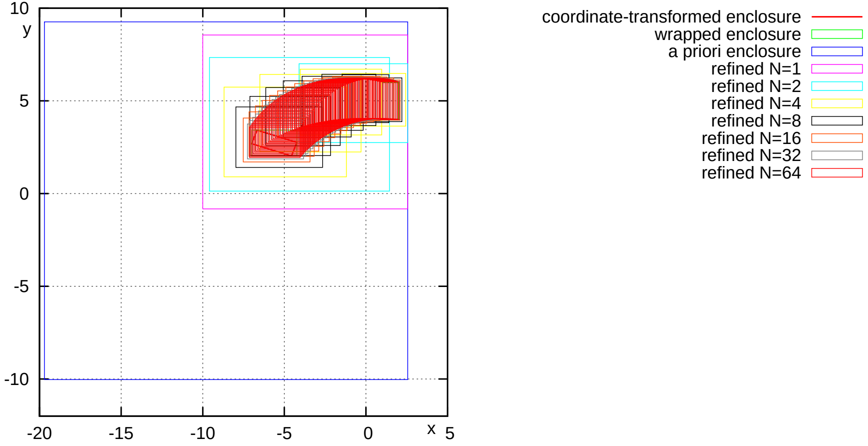
$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1, 2], \quad y_0 \in [4, 6], \quad t_1 = 1.6$$

VNODE-LP Output and Re-evaluated ITS



$$\dot{x} = -y, \quad \dot{y} = 0.6 \cdot x, \quad x_0 \in [1,2], \quad y_0 \in [4,6], \quad t_1 = 1.6$$

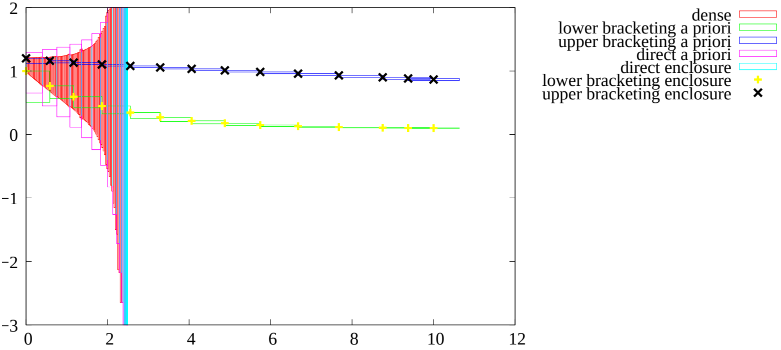
Bracketing Systems

[Ramdani, Meslem, and Candau, 2009], [Eggers, Ramdani, Nedialkov, and Fränzle, 2011]

- 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** [Müller, 1927]
- 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

[Eggers, Ramdani, Nedialkov, and Fränzle, 2011]



x dimension of

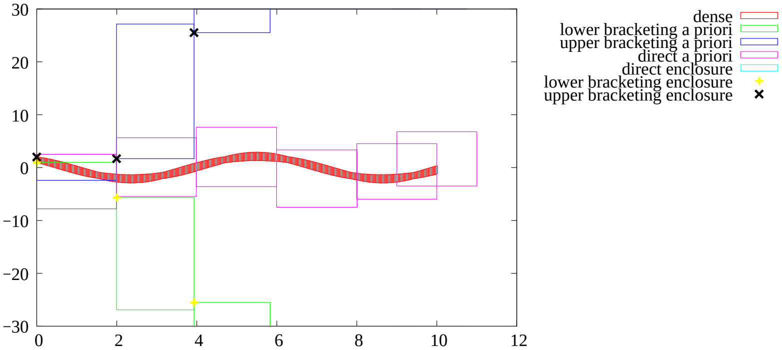
$$\dot{x} = -p_4 x - \frac{p_1 x}{1 + p_2 y} + p_3 y + 0.1$$

$$\dot{y} = p_4 x - p_3 y$$

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

Comparison: Direct vs. Bracketing

[Eggers, Ramdani, Nedialkov, and Fränzle, 2011]



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

- Complementary strengths

$\Rightarrow$ iSAT-ODE: **Intersect both enclosures for tighter result.**

Combining Direct and Bracketing Methods

Initially:

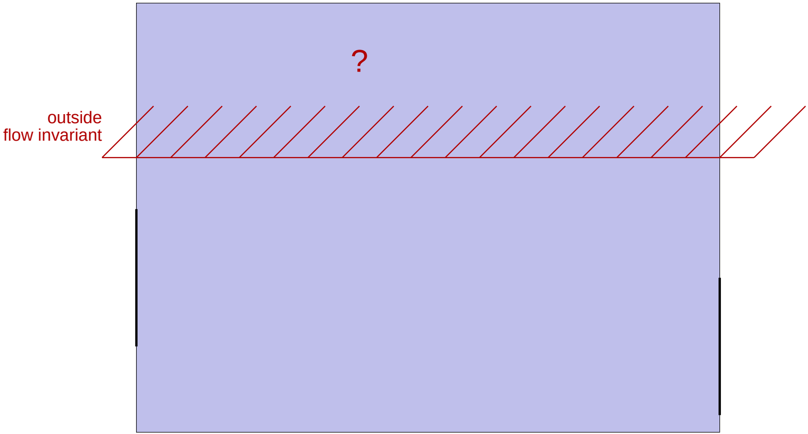
- Compute direct step if possible $\rightarrow t_{dir}$
- Compute bracketing step if possible $\rightarrow t_{br}$

Always re-evaluate stored Taylor Series for entire interval to get better enclosure than a-priori bounds.

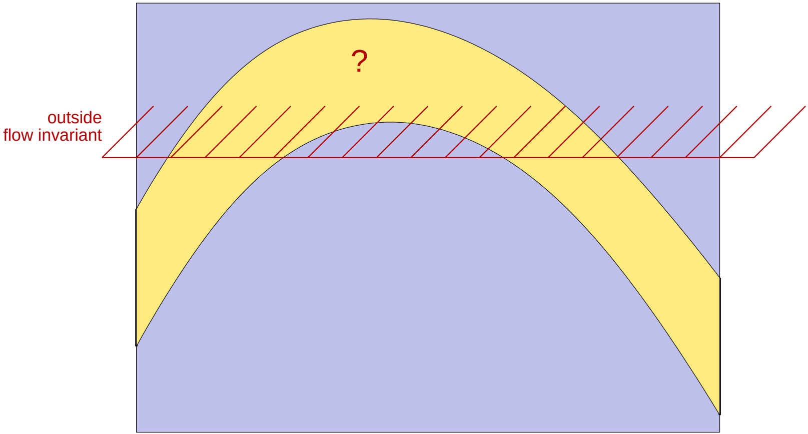
Iteratively:

- Intersect direct and bracketing enclosures up to $\min(t_{dir}, t_{br})$
  - If entirely outside flow invariant: stop enclosure
  - If entirely inside flow invariant: continue with next step
  - If partially outside: try to find out whether and when flow invariant is left
- Compute step with the method whose enclosure lags behind

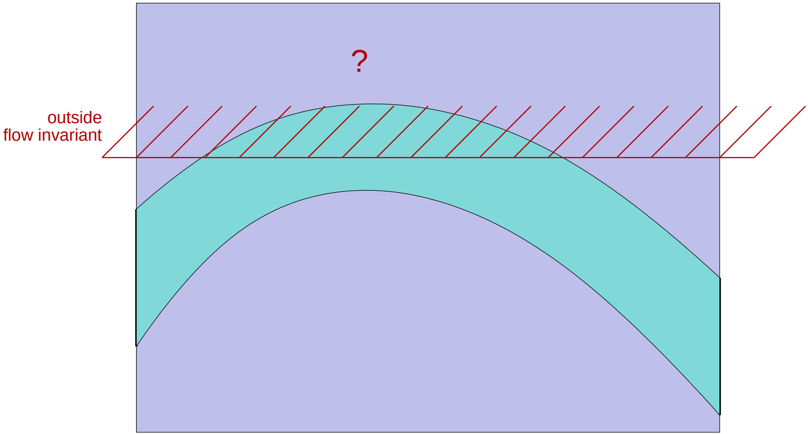
Detect Empty Intersection with Flow Invariant



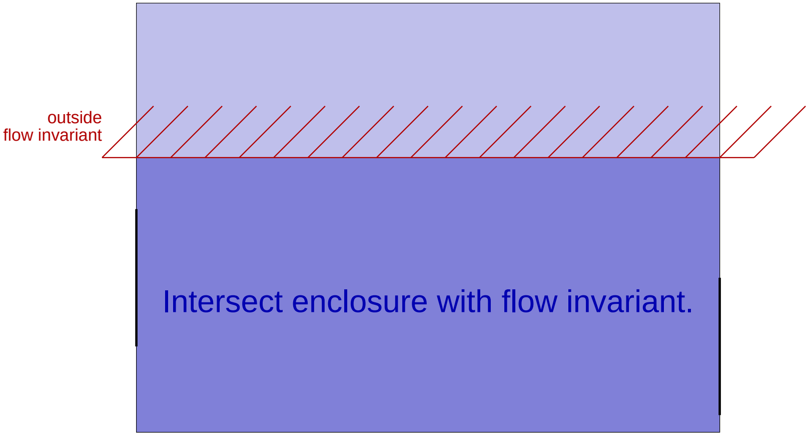
Detect Empty Intersection with Flow Invariant



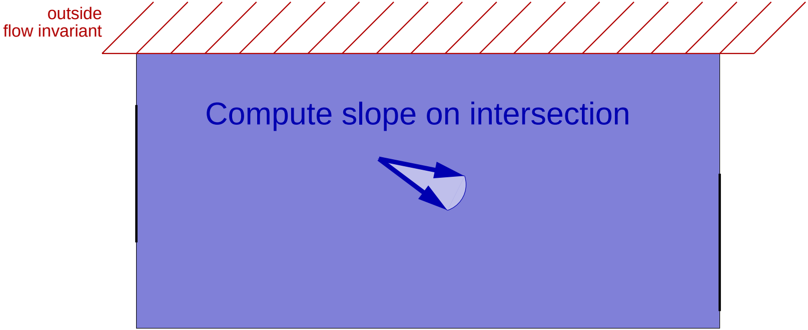
Detect Empty Intersection with Flow Invariant



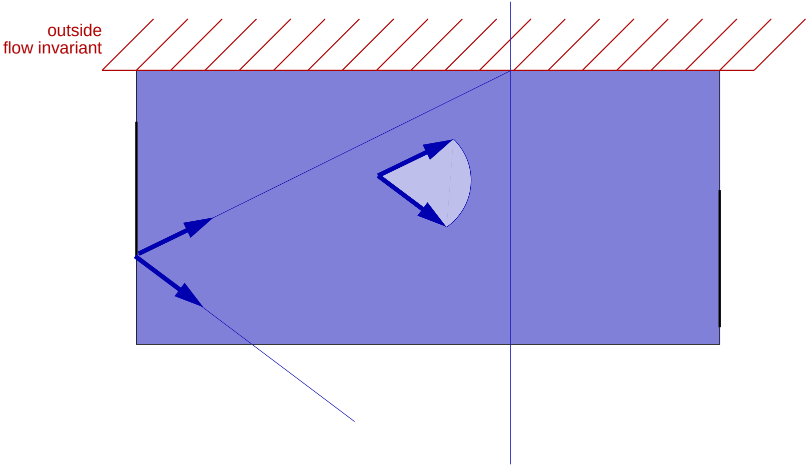
Detect Empty Intersection with Flow Invariant



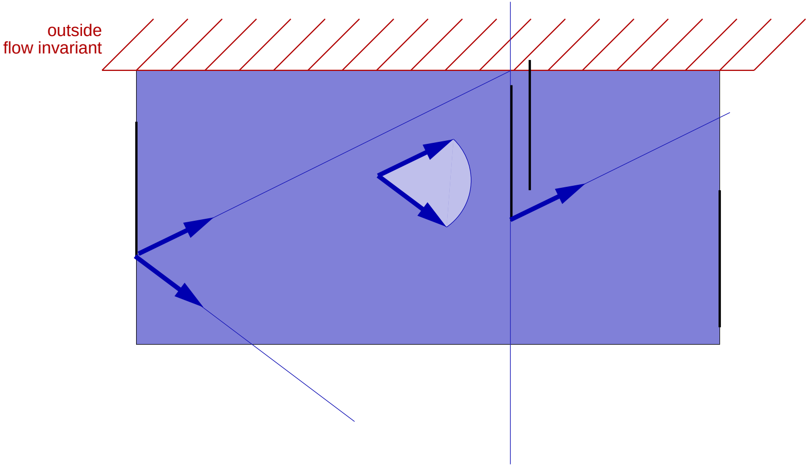
Detect Empty Intersection with Flow Invariant



Detect Empty Intersection with Flow Invariant

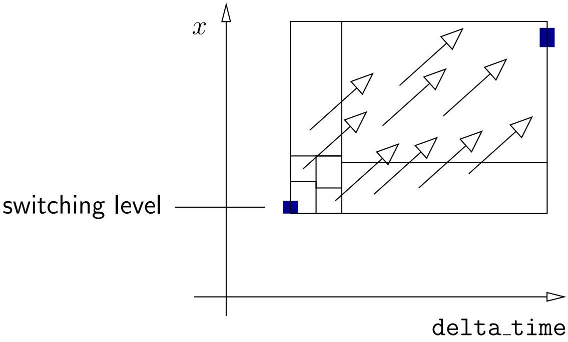


Detect Empty Intersection with Flow Invariant



Deducing Trajectory Directions

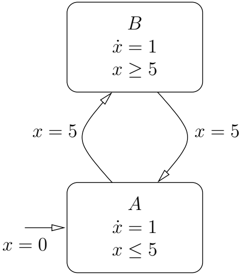
[Eggers, Ramdani, Nedialkov, and Fränzle, 2011]



- First enclosure always contains switching surface, no matter how much refinement is done
- Solver cannot exclude possibility of another switch

Deducing Trajectory Directions

[Eggers, Ramdani, Nedialkov, and Fränzle, 2011]



$(A, x = 0, t = 0), (A, x = 5, t = 5),$
$(B, x = 5, t = 5), (A, x = 5, t = 5), \ldots$

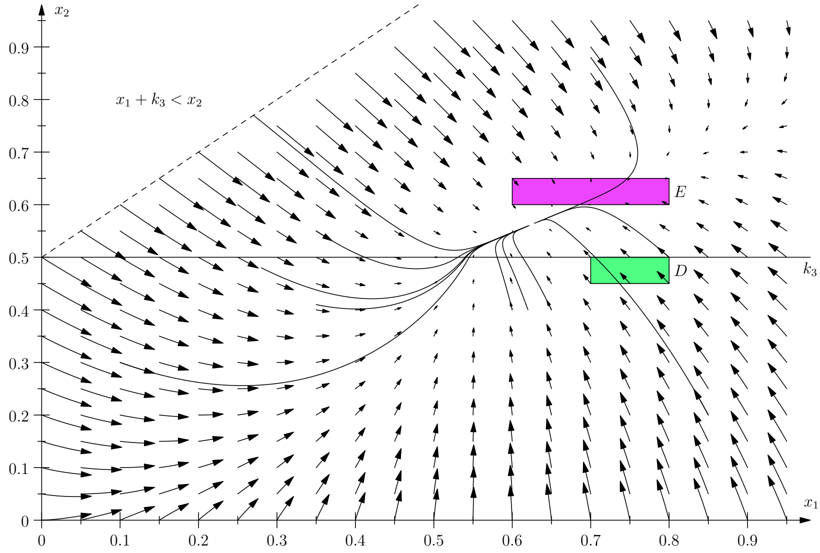
- $x = 5$ included even in tightest possible enclosure after switch
- Can thus *oscillate* between modes A and B
- Problematic when trying to *enforce time progress* (hinders time-bounded trajectories from becoming step-bounded)
- Solution: **deduce direction of trajectory at its beginning**, here, e.g. $\texttt{delta\_time} > 0 \land \texttt{delta\_time} \leq \tau \Rightarrow x' > x$
- In general: **evaluate ODE's right-hand side over prefix of trajectory**, e.g. as long as strictly positive: variable grows

Learning ODE Deductions

[Eggers, Fränzle, and Herde, 2009], [Eggers, Ramdani, Nedialkov, and Fränzle, 2011]

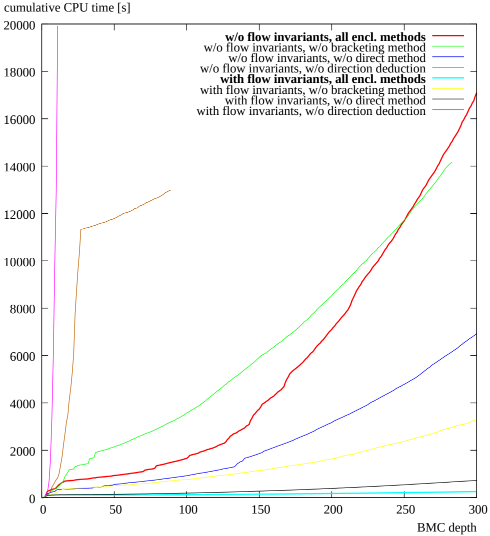
- ODE enclosures **very expensive** compared to simple interval constraint propagations
- **Preserve** once-learnt facts from deletion during backjumps (similar to learning conflict clauses [Marques-Silva and Sakallah, 1996])
- Learn deduced facts for all **isomorphic instances** (constraints replication [Shtrichman, 2000])
- Two main ingredients:
  - iSAT core: **learn new clauses** during search (multiple at once, not necessarily conflicts, potentially introducing new variables to safely represent constants)
  - ODE layer: **recognize** enclosure requests that have already been answered (or can be subsumed under previously answered requests)
- Additionally: store limited number of VNODE's **intermediate results** for reuse, when partial request is detected (e.g. compatible initial box but tighter `delta_time` range)

Bounded reachability



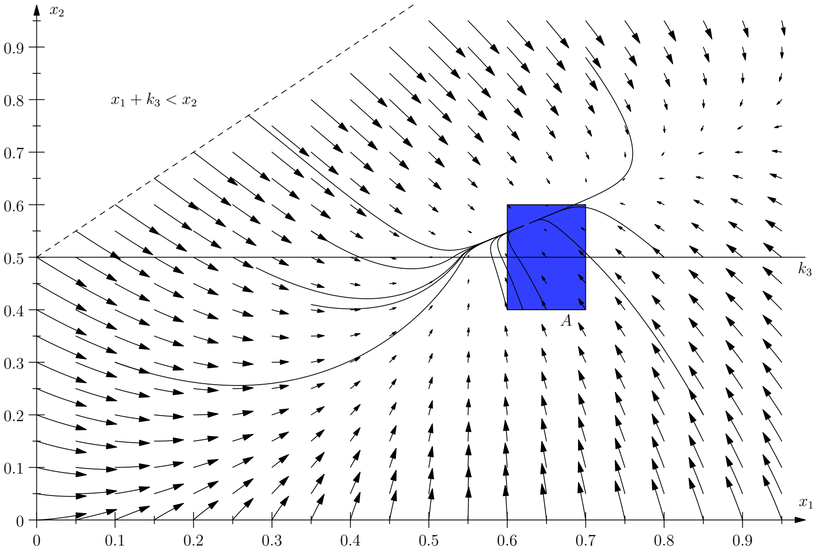
Bounded reachability

- **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.



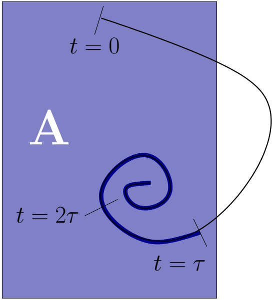
Two Tanks: Stabilization into Region

[Podelski and Wagner, 2007], [Eggers, Ramdani, Nedialkov, and Fränzle, 2011]

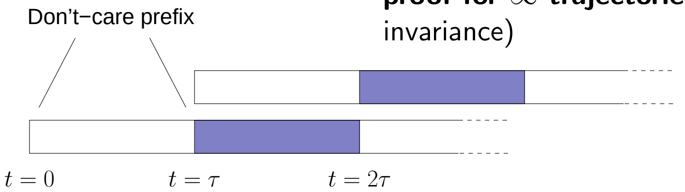


Two Tanks: Stabilization into Region

[Podelski and Wagner, 2007], [Eggers, Ramdani, Nedialkov, and Fränzle, 2011]



- Prove: all trajectories starting in A stay in A for $t = [\tau, 2\tau]$
- **Time-bounded** property
- Enforce **time progress** in each transition step
- **Step-bounded** property
- Target: have not reached 2τ or have left A within $[\tau, 2\tau]$
- When unsatisfiable, have **inductive proof for ∞-trajectories** (time invariance)



Two Tanks: Stabilization into Region

[Podelski and Wagner, 2007], [Eggers, Ramdani, Nedialkov, and Fränzle, 2011]

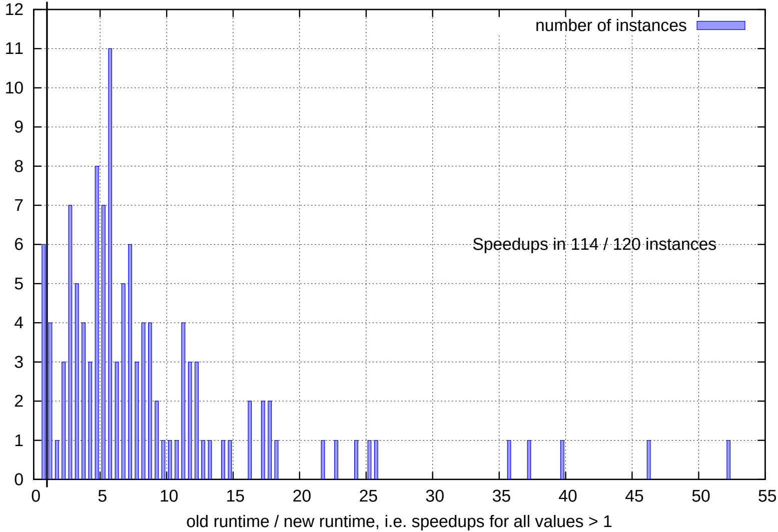
Solver results ($\tau = 5$) and runtimes in seconds (2.4 GHz AMD Opteron):

	all			no bracketing			no direct			no direction		
depth	old	new	o/n	old	new	o/n	old	new	o/n	old	new	o/n
1	unknown, 111.9	unknown, 149.2	0.75	unknown, 42.0	unknown, 4.7	9.01	unknown, 61.5	unknown, 94.0	0.65	unknown, 111.5	unknown, 146.7	0.76
2	unknown, 467.5	unknown, 157.9	2.96	unknown, 981.0	unknown, 451.0	2.18	unknown, 346.3	unknown, 102.9	3.36	unknown, 342.0	unknown, 39.7	8.61
3	**unsat**, 674.0	**unsat**, 147.8	4.56	**unsat**, 5011.6	**unsat**, 196.9	25.45	**unsat**, 404.2	**unsat**, 96.5	4.19	unknown, 478.8	unknown, 126.0	3.80
4	**unsat**, 812.1	**unsat**, 237.2	3.42	**unsat**, 1995.1	**unsat**, 706.4	2.82	**unsat**, 499.1	**unsat**, 92.4	5.40	unknown, 547.5	unknown, 196.0	2.79
5	**unsat**, 986.0	**unsat**, 270.3	3.65	**unsat**, 2431.7	**unsat**, 276.1	8.81	**unsat**, 601.1	**unsat**, 125.9	4.77	unknown, 682.4	unknown, 243.7	2.80
6	**unsat**, 1126.1	**unsat**, 227.2	4.96	**unsat**, 3303.4	**unsat**, 466.7	7.08	**unsat**, 705.0	**unsat**, 227.3	3.10	unknown, 834.2	unknown, 191.7	4.35
7	**unsat**, 1277.2	**unsat**, 254.8	5.01	**unsat**, 2486.8	**unsat**, 224.7	11.07	**unsat**, 803.6	**unsat**, 143.2	5.61	unknown, 982.5	unknown, 328.6	2.99
8	**unsat**, 1451.4	**unsat**, 279.4	5.20	**unsat**, 5273.3	**unsat**, 406.5	12.97	**unsat**, 890.8	**unsat**, 159.6	5.58	unknown, 1115.7	unknown, 434.1	2.57
9	**unsat**, 1584.6	**unsat**, 328.2	4.83	**unsat**, 4905.2	**unsat**, 444.0	11.05	**unsat**, 966.5	**unsat**, 151.5	6.38	unknown, 1235.8	unknown, 1203.0	1.03
10	**unsat**, 1706.6	**unsat**, 312.2	5.47	**unsat**, 6396.1	**unsat**, 430.7	14.85	**unsat**, 1053.2	**unsat**, 152.2	6.92	unknown, 1356.0	unknown, 807.6	1.68

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

Runtime Impact

Impact on the solver runtimes for three instances of stabilization proofs for the two tank system (unwinding depths 1–10, four settings each).



Conclusions & Future Work

- Use of **flow invariants** to stop enclosures when all trajectories have left admissible region
  - Currently confined to box-shaped invariants
  - Currently not trying to compute enclosures for intersections with flow invariant
- Improved **caching** (not detailed here): decision tree structure to accelerate finding previously-answered queries
- **Extraction of Taylor Coefficients** from VNODE-LP to reduce computational cost of substep enclosures

References I

- G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani (2005).
  Verifying industrial hybrid systems with mathsat.
  In *Proceedings of the 2nd International Workshop on Bounded Model Checking (BMC 2004)*, Volume 119 of *Electronic Notes in Theoretical Computer Science*, pp. 17 – 32. Elsevier.
- A. Biere, A. Cimatti, E. Clarke, and Y. Zhu (1999).
  Symbolic model checking without bdds.
  In W. Cleaveland (Ed.), *Tools and Algorithms for the Construction and Analysis of Systems*, Volume 1579 of *Lecture Notes in Computer Science*, pp. 193–207. Springer Berlin / Heidelberg.
- M. Fränzle and C. Herde (2005).
  Efficient proof engines for bounded model checking of hybrid systems.
  In *Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 04)*, Volume 133 of *Electronic Notes in Theoretical Computer Science (ENTCS)*, pp. 119–137. Elsevier.
- J. P. Marques-Silva and K. A. Sakallah (1996).
  Grasp - a new search algorithm for satisfiability.
  In *in Proceedings of the International Conference on Computer-Aided Design*, pp. 220–227.
- M. Müller (1927).
  Über das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen.
  Mathematische Zeitschrift 26, 619–645.
- A. Podelski and S. Wagner (2007).
  Region stability proofs for hybrid systems.
  In *FORMATS*, pp. 320–335.
- O. Shtrichman (2000).
  Tuning SAT checkers for bounded model checking.
  In E. Emerson and A. Sistla (Eds.), *Computer Aided Verification*, Volume 1855 of *LNCS*, pp. 480–494. Springer.

References II

- A. Eggers, M. Fränzle, and C. Herde (2008).
  SAT modulo ODE: A direct SAT approach to hybrid systems.
  In S. S. Cha, J.-Y. Choi, M. Kim, I. Lee, and M. Viswanathan (Eds.), *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis (ATVA'08)*, Volume 5311 of *Lecture Notes in Computer Science*, pp. 171–185. Springer.

- A. Eggers, M. Fränzle, and C. Herde (2009).
  Application of constraint solving and ode-enclosure methods to the analysis of hybrid systems.
  In T. E. Simos, G. Psihoyios, and C. Tsitouras (Eds.), *NUMERICAL ANALYSIS AND APPLIED MATHEMATICS: International Conference on Numerical Analysis and Applied Mathematics 2009*, Volume 1168 of *AIP Conference Proceedings*, Melville, New York, pp. 1326–1330. American Institute of Physics.

- A. Eggers, N. Ramdani, N. S. Nedialkov, and M. Fränzle (2011).
  Improving SAT modulo ODE for hybrid systems analysis by combining different enclosure methods.
  In G. Barthe, A. Pardo, and G. Schneider (Eds.), *Proceedings of the Ninth International Conference on Software Engineering and Formal Methods (SEFM)*, Volume 7041 of *LNCS*, pp. 172–187. Springer Berlin / Heidelberg.

- O. Stursberg, S. Kowalewski, I. Hoffmann, and J. Preußig (1997).
  Comparing timed and hybrid automata as approximations of continuous systems.
  In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry (Eds.), *Hybrid Systems IV*, Volume 1273 of *Lecture Notes in Computer Science*, pp. 361–377. Springer Berlin / Heidelberg.
  10.1007/BFb0031569.

- M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige (2007).
  Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure.
  JSAT Special Issue on Constraint Programming and SAT 1, 209–236.

- N. Nedialkov (2006).
  Vnode-lp – a validated solver for initial value problems in ordinary differential equations.
  Technical Report CAS-06-06-NN, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada.

References III

- N. Nedialkov and K. Jackson (1999).
  An interval hermite-obreschkoff method for computing rigorous bounds on the solution of an initial value problem for an ordinary differential equation.
  Reliable Computing 5, 289–310.

- N. S. Nedialkov, K. R. Jackson, and J. D. Pryce (2001).
  An effective high-order interval method for validating existence and uniqueness of the solution of an ivp for an ode.
  Reliable Computing 7, 449–465.
  10.1023/A:1014798618404.

- N. Ramdani, N. Meslem, and Y. Candau (2009).
  A hybrid bounding method for computing an over-approximation for the reachable space of uncertain nonlinear systems.
  IEEE Transactions on Automatic Control 54(10), 2352–2364.

- N. Ramdani, N. Meslem, and Y. Candau (2010).
  Computing reachable sets for uncertain nonlinear monotone systems.
  Nonlinear Analysis : Hybrid Systems 4(2), 263–278.

- D. Ishii, K. Ueda, and H. Hosobe (2011, March).
  An interval-based sat modulo ode solver for model checking nonlinear hybrid systems.
  International Journal on Software Tools for Technology Transfer (STTT), 1–13.