The General Case

Beyond Initialized Systems

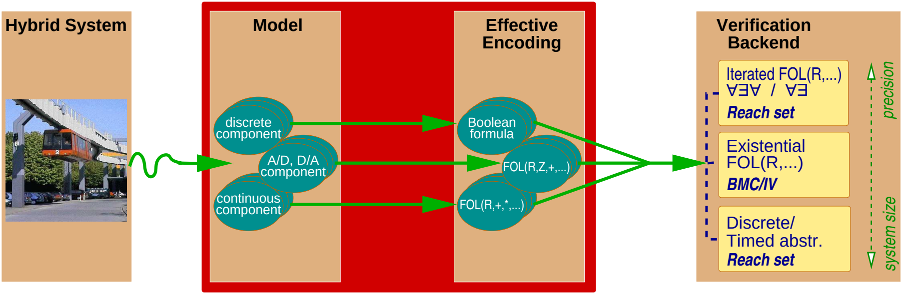
Further Agenda

1. Translation of high-level models
   - Simulink + Stateflow
   - Compositional translation
   - based on predicative encoding of block invariants
2. Basic principles of state-exploratory analysis of HA
   - Finite-state abstraction vs. hybridisation vs. image computation of ODEs
   - iterating a FO-definable map
3. A sample tool set
   - SAT-modulo-theory based
   - four (increasingly experimental) levels:
     - linear hybrid automata vs. LinSAT
     - non-linear assignments
     - non-linear differential equations
     - probabilistic hybrid systems

Verification Frontend

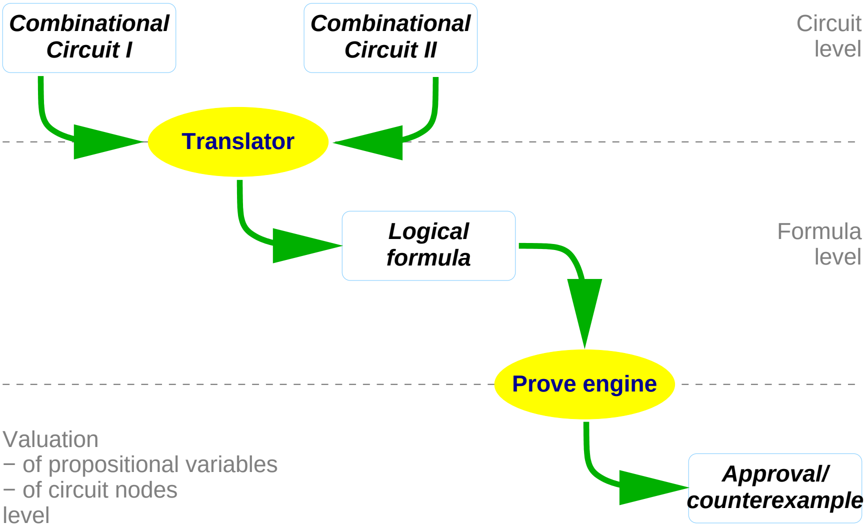
Translation of hybrid systems to arithmetic constraints

Translation



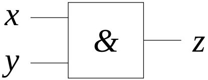
- Compositional translation into many-sorted logics

Analogy: Combinatorial Circuits

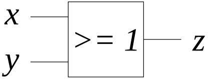


Mapping circuits to formulae

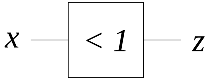
A gate is mapped to a propositional formula formalizing its invariant:



$\mapsto \quad x \wedge y \Leftrightarrow z$



$\mapsto \quad x \vee y \Leftrightarrow z$



$\mapsto \quad \neg x \Leftrightarrow z$

$\vdots \qquad \mapsto$ combinations thereof.

Circuit behavior corresponds to conjunction of all its gate formulae.

Formalizing circuit equivalence

- Given two circuits C and D, we obtain formulae ϕ_C and ϕ_D,
- furthermore, have correspondence lists $I \subset Node_C \times Node_D$ and $O \subset Node_C \times Node_D$ for in- and outputs.
- generate formula $Eq(C, D) =$

$$\left(\phi_C \wedge \phi_D \wedge \bigwedge_{(i,j) \in I} (i \Leftrightarrow j) \right) \Longrightarrow \bigwedge_{(o,p) \in O} (o \Leftrightarrow p)$$

☺ $\neg Eq(C, D)$ is satisfiable iff the two circuits are functionally different.

☺ Each satisfying valuation provides a counterexample to circuit equivalence.

Enumerating valuations

. . . is completely out-of-scope:

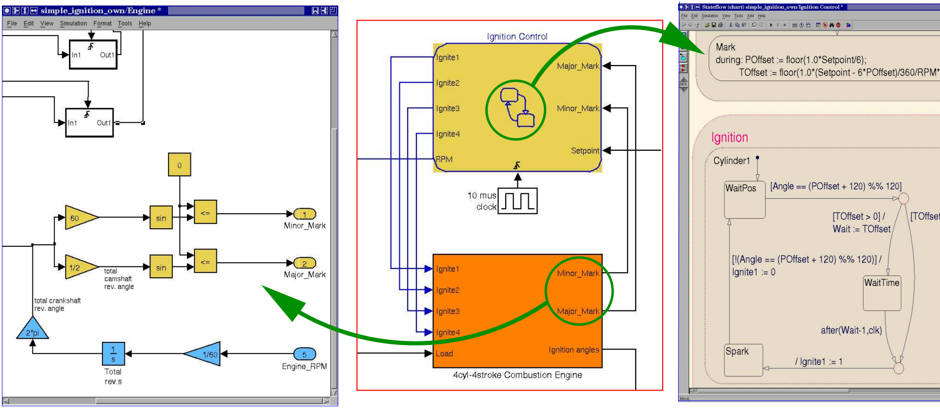
- When comparing two circuits of (only) 10.000 nodes, we need to explore $4 \cdot 10^{6020}$ possible valuations.
- If we were able to explore $10^8 \frac{\text{valuations}}{s}$, this would take $7 \cdot 10^{6017}$ years.

Enumerating only inputs is *considerably* more efficient, but still out-of-scope:

- When comparing two circuits with 100 input nodes, we need to explore $1.3 \cdot 10^{30}$ possible valuations.
- If we were able to explore $10^8 \frac{\text{input valuations}}{s}$, this would still take $9.6 \cdot 10^{15}$ years.

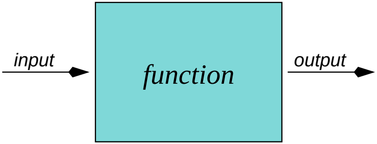
Yet routinely solved by recent propositional satisfiability solvers!

Generalizing the concept: Simulink+Stateflow



Functional blocks / signal transducers

- Dynamic system is a network of *basic blocks*:



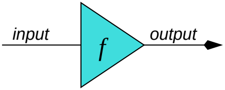
- Blocks are connected via *directed links* that share a state variable
- The time model is (two-dimensional) time over real-valued physical time,
  yielding a continuous-time data flow semantics.

Basic blocks

Basic blocks are *signal transducers* with a 'simple' characterization in the time domain, e.g.

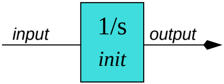
- *'algebraic' blocks:* output is a *time-invariant* function of input:

$$out(t) = f\,(in(t))$$

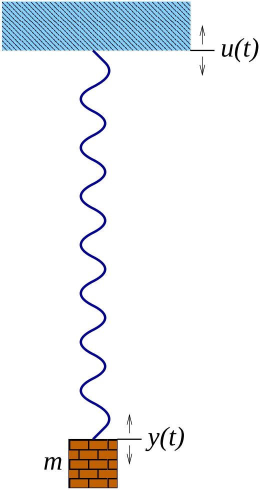


- *state-holding blocks:* integrators & friends, e.g.

$$out(t) = init + \int_0^t in(u)\,\mathrm{d}u$$



Example: spring-mass system w. disturbance



- **Basic model:**

$$\ddot{y}(t) = \frac{F(t)}{m}$$
$$F(t) = k\,(l(t) - l_0)$$
$$l(t) = u(t) - y(t)$$

- **Replace higher-order derivatives:**

Add $v(t) = \dot{y}(t)$.

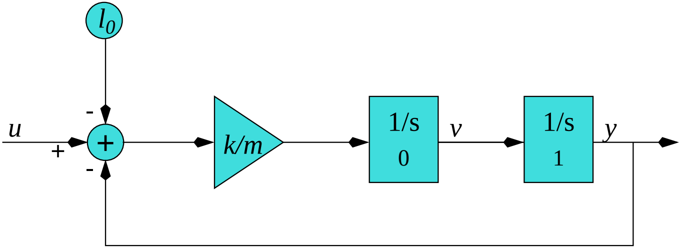
Gives
$$\dot{y}(t) = v(t)$$
$$\dot{v}(t) = \frac{k}{m}(u(t) - y(t) - l_0)$$

Example: spring-mass system w. disturbance

- **DE:**
$$\begin{aligned} \dot{y}(t) &= v(t), & y(0) &= 1 \\ \dot{v}(t) &= \frac{k}{m}(u(t) - y(t) - l_0), & v(0) &= 0 \end{aligned}$$

- **After integration:**
$$\begin{aligned} y(t) &= 1 + \int_0^t v(z)\mathrm{d}z \\ v(t) &= 0 + \int_0^t \frac{k}{m}(u(z) - y(z) - l_0)\,\mathrm{d}z \end{aligned}$$

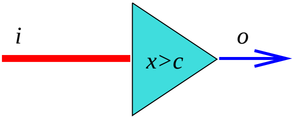
- **Functional block model:**



A/D coupling components

have an idealized, *delay-free* semantics:

- **Threshold sensor:**

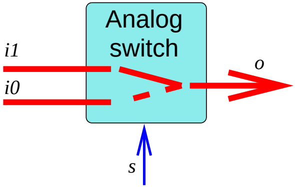


  - Analog input $i : Time \longrightarrow \mathbb{R}$,
  - digital output $o : Time \longrightarrow \mathbb{B}$,
  - dynamics: $o(t) = (i(t) > c)$.

D/A coupling components

also have an idealized, *delay-free* semantics:

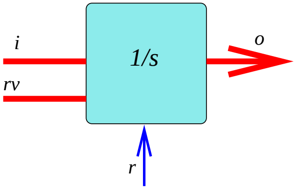
- **Analog switch:**



  - Analog inputs $i_{0,1} : Time \longrightarrow \mathbb{R}$,
  - digital input $s : Time \longrightarrow \mathbb{B}$,
  - analog output $o : Time \longrightarrow \mathbb{R}$,
  - dynamics: $o(t) = \begin{cases} i_1(t) & \text{, if } s(t) \\ i_0(t) & \text{, if } \neg s(t) \end{cases}$.

D/A coupling components cntd.

- **Resettable integrator:**



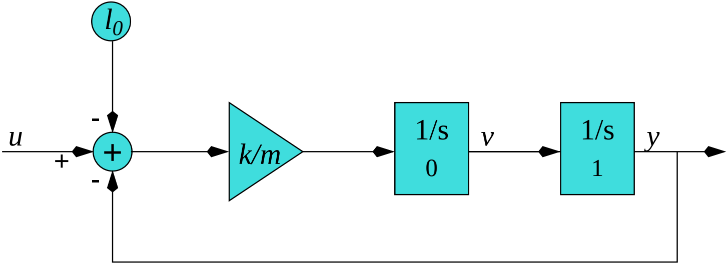
  - Analog inputs/output $i, rv, o : Time \to \mathbb{R}$,
  - Digital input $r : Time \to \mathbb{B}$,
  - dynamics: $$\begin{aligned} o(t) &= rv(t_r) + \int_{t_r}^{t} i(t)\,\mathrm{d}t \quad , \text{ where} \\ t_r &= \sup\{t' \le t \mid r(t')\}. \end{aligned}$$

Dynamics of networks

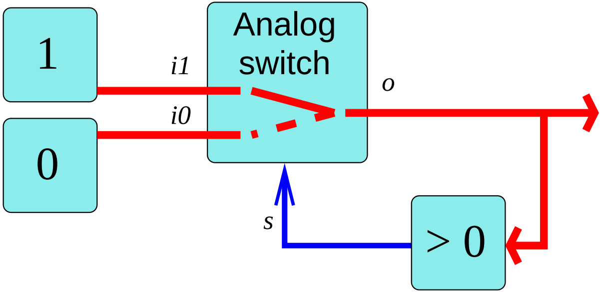
1. The individual blocks impose relations between their input and output waveforms.
2. These relations are adequately covered by the aforementioned characteristic equations of the various basic blocks.
3. Consequently, the dynamics of a network of basic blocks coincides to (solutions of) the conjunction of the characteristic equations of the entailed blocks.

But how to avoid spontaneous, non-causal state changes?

The sane case



The insane case



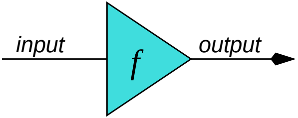
$$o(t) = \begin{cases} 1 & \text{, if } o(t) > 0 \\ 0 & \text{, if } o(t) \leq 0 \end{cases}$$

Semantics permits non-causal switching, i.e. full non-determinism.

Avoiding non-causality

1. Simulink (and many other languages) forbids delay-free loops:
   - each loop in the "circuit" has to contain at least one delaying element
     - an integrator
     - a delay block
     - ...
   - if a two-dimensional time model is adopted, even δ-delays suffice!
2. some modeling frameworks interpret delay-free loops as fixed point equations
   - try to solve these equations
   - solution is taken if it is unique

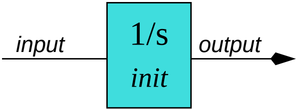
Towards FO Representation: ‘Algebraic’ blocks



- time-invariant transfer function $\mathit{output}(t) = f\,(\mathit{input}(t))$
- made 1st-order by making time implicit: $\mathit{Flow} \equiv \mathit{output} = f\,(\mathit{input})$
- no constraints on initial value: $\mathit{Init} \equiv \mathrm{true}$,
- discontinuous jumps always admissible $\mathit{Jump} \equiv \mathrm{true}$,

All the formulae are elements of a suitably rich 1st-order logics over $\mathbb{R}$.

Towards FO Representation: Integrators



- integrates its input over time: $output(t) = init + \int_0^t input(u)\,\mathrm{d}u$.
- made semi-1st-order by using derivatives: $Flow \equiv \frac{doutput}{dt} = input$
- initial value is rest value: $Init \equiv output = init$,
- discontinuous jumps don't affect output $Jump \equiv output = \overleftarrow{output}$,

Use in Model Exploration

Given: Transition pred. $trans(x, x')$, initial state pred. $init(x)$, conj. invar. $\phi(x)$.

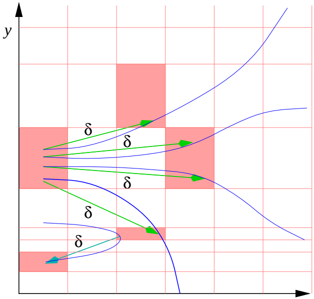
E.g., **Bounded Model Checking (BMC) algorithm:**

1. For given $i \in \mathbb{N}$ check for satisfiability of
$$\neg \left(\begin{array}{ll} & init(x_0) \wedge trans(x_0, x_1) \wedge \ldots \wedge trans(x_{i-1}, x_i) \\ \Rightarrow & \phi(x_0) \wedge \ldots \wedge \phi(x_i) \end{array} \right).$$
If test succeeds then report violation of goal.
2. Otherwise repeat with larger i.

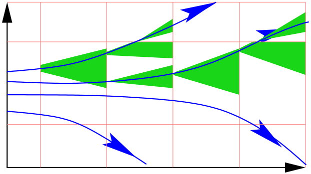
Can we use the predicates off-the-shelf?
No, as dynamics is not in terms of pure pre-/post-relations.

Images of ODEs: Approaches

1. *Safe finite-state abstraction:*
   - ☺ E.g., discretization through quantization (and overapproximation); yields finite-state system.
   - ☹ exponential in dimension of system
   - ☹ coarse abstractions give many false negatives ⇝ CEGAR



2. *Hybridization:* chop the phase space; do piecewise safe approximation by tractable dynamics (e.g., maps definable in decidable logics over $\mathbb{R}$)
   - ☺ potentially more concise,
   - ☹ yet still exponential in dimension of system



3. (Safely approximate) *on-the-fly computation of ODE images.*

Hybridization

Will not elaborate on into this issue here: approaches range from

- approximation by piecewise (i.e., in a grid element) constant differential inclusions obtained via interval-based safe approx. of upper and lower bounds on individual derivatives:

  $$\frac{\mathrm{d}x}{\mathrm{d}t} = x^2 + 2y \wedge x \in [1,2] \wedge y \in [5,7] \quad \rightsquigarrow \quad \frac{\mathrm{d}x}{\mathrm{d}t} \in [11,18]$$

  a.o. [Henzinger, Kopke, Puri, Varaiya 1998] [Stursberg, Kowalewski 1999]
- to approximation by piecew. affine / multi-affine vector fields [Asarin, Dang, Girard 06]
- and to Taylor approximations [Piazza et al. 05, Lanotte, Tini 05]

For Lipschitz-continuous ODEs, imprecision generally is

- linear in grid width (though with different constants),
- exponential in length of time frame.

e.g., [Girard 2002; Asarin, Dang, Girard 2006]

Impact on decidability

Due to the (worst-case) exponential deviation over time, such hybridizations are not sufficient for approximate (up to some ε) computation of the reachable state space over unbounded time frames.

Hence, questions like

- "If the distance of the reachable state space from a set of bad states is larger than ε then provide a proof of this fact."

for flows lacking a closed-form solution are i.g. not "decidable" by hybridization and related approximation schemes.

[Platzer, Clarke 2006]

...unless the flow is attracting such that it cancels the accumulating error.

[Asarin, Dang, Girard 2006]

Principles of hybrid state-space exploration:

Iterating a 1st-order definable map

Checking safety

...in a finite Kripke structure:

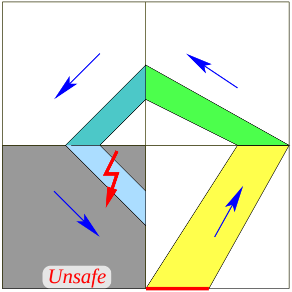
1. For increasing n, calculate the set $Reach^{\leq n}$ of states reachable in at most n steps.
2. Chain $Reach^{\leq 1} \subseteq Reach^{\leq 2} \subseteq \dots$ has only a finite ascending sub-chain due to finiteness of state-space.

$\Rightarrow$ Set $\bigcup_{n \in \mathbb{N}} Reach^{\leq n}$ of reachable states can be constructed in finitely many steps.

3. Check for intersection with set of unsafe states.

...in a hybrid automaton:

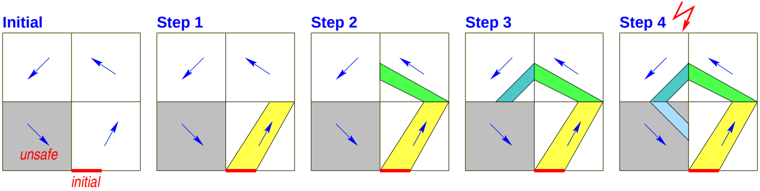
Similar fixpoint construction



need not terminate,
but yields an effective procedure for falsification.

Making the idea operational: the ingredients

Idea: Iterate transition relation and continuous dynamics until an unsafe state is hit:



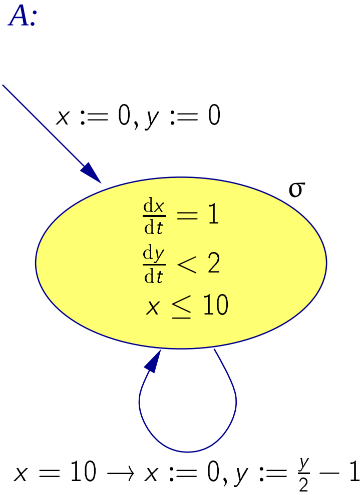
Result: Terminates iff HA is unsafe.

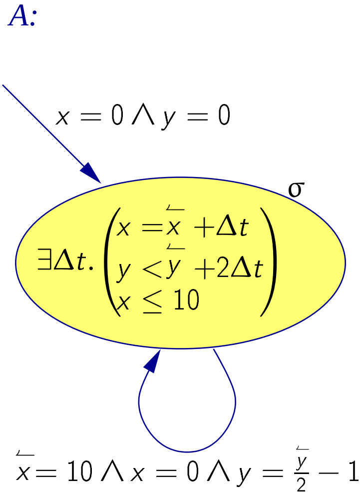
Requires: Effective representations of transition relation, continuous dynamics, and initial, intermediate, and unsafe state sets s.t.

1. Calculation of the state set reachable within $n \in \mathbb{N}$ steps is effective,
2. Emptiness of intersection of unsafe state set with the state set reachable in n steps is decidable.

(implemented in, e.g., HyTech [Henzinger, Ho, Wong-Toi, 1995–])

From hybrid automata to logic





Convexity of behaviors required, continuity is not FO-expressible!

Essentials of polynomial HA

- Finite set Σ of discrete states, finite vector $\mathbf{x}$ of cont. variables
- An activity predicate $act_\sigma \in \mathrm{FOL}(\mathbb{R}, =, +, \times)$ defines the possible evolution of the continuous state while the system is in discrete state σ
- A transition predicate $trans_{\sigma \to \sigma'} \in \mathrm{FOL}(\mathbb{R}, =, +, \times)$ defines guard and effect of transition from discrete state σ to discrete state σ'
- A path is a sequence $\langle(\sigma_0, \mathbf{y}_0), (\sigma_1, \mathbf{y}_1), \ldots\rangle \in (\Sigma \times \mathbb{R}^d)^{\star|\omega}$ entailing an alternation of transitions and activities:
  - $(\overleftarrow{\mathbf{x}} := \mathbf{y}_i, \mathbf{x} := \mathbf{y}_{i+1}) \models trans_{\sigma_i \to \sigma_{i+1}}$ if i is odd
  - $(\overleftarrow{\mathbf{x}} := \mathbf{y}_i, \mathbf{x} := \mathbf{y}_{i+1}) \models act_{\sigma_i}$ and $\sigma_i = \sigma_{i+1}$ if i is even
  - $(\mathbf{x} := \mathbf{y}_0) \models initial_{\sigma_0}$

Decidability of $\mathrm{FOL}(\mathbb{R}, =, +, \times)$ yields decision procedures for temporal properties of paths of *finitely fixed length*

Reachability

of a final discrete state σ' from an initial discrete state σ and through an execution containing n transitions can be formalized through the inductively defined predicate $\phi^n_{\sigma \to \sigma'}$, where

$$\phi^0_{\sigma \to \sigma'} = \begin{cases} \text{false}, & \text{if } \sigma \neq \sigma', \\ act_\sigma, & \text{if } \sigma = \sigma', \end{cases}$$

$$\phi^{n+1}_{\sigma \to \sigma'} = \bigvee_{\tilde{\sigma} \in \Sigma} \exists \mathbf{x}_1, \mathbf{x}_2 . \begin{pmatrix} \phi^n_{\sigma \to \tilde{\sigma}}[\mathbf{x}_1/\mathbf{x}] \wedge \\ trans_{\tilde{\sigma} \to \sigma'}[\mathbf{x}_1, \mathbf{x}_2/\overleftarrow{\mathbf{x}}, \mathbf{x}] \wedge \\ act_{\sigma'}[\mathbf{x}_2/\overleftarrow{\mathbf{x}}] \end{pmatrix}$$

Safety of hybrid automata

⇒ An unsafe state is reachable within n steps iff

$$Unsafe_n = \bigvee_{\sigma' \in \Sigma} Reach^{\leq n}_{\sigma'} \wedge \neg safe_{\sigma'}$$

is satisfiable, where

$$Reach^{\leq n}_{\sigma'} = \bigvee_{i \in \mathbb{N}_{\leq n}} \bigvee_{\sigma \in \Sigma} \phi^{i}_{\sigma \to \sigma'} \wedge initial_{\sigma}[\overleftarrow{\mathbf{x}} / \mathbf{x}]$$

characterizes the continuous states reachable in at most n steps within discrete state σ'.

⇒ An unsafe state is reachable iff there is some $n \in \mathbb{N}$ for which $Unsafe_n$ is satisfiable.

The semi-decision procedure

1. $\mathrm{FOL}(\mathbb{R}, =, +, \times)$ is decidable. [Tarski 1948]
2. *$Unsafe_n$* is a formula of $\mathrm{FOL}(\mathbb{R}, =, +, \times)$.

$\Rightarrow$ For arbitrary $n \in \mathbb{N}$ it is *decidable whether an unsafe state is reachable within n steps.*

3. By successively testing increasing n, this yields a *semi-decision procedure for reachability of unsafe states*:
   1. Select some $n \in \mathbb{N}$,
   2. check *$Unsafe_n$*.
   3. If this yields true then an unsafe state is reachable.
      Report this and terminate.
   4. Otherwise select strictly larger $n \in \mathbb{N}$ and redo from step (b).

The semi-decision procedure — contd.

Note that in general the semi-decision procedure can only detect being unsafe, yet does not terminate iff the HA is safe. Hence, it

- ☺ *can be used for falsifying HA*,
- ☹ *but not for verifying them*.

However, there are cases where $Reach^{\leq n+1}_{\sigma'} \Rightarrow Reach^{\leq n}_{\sigma'}$ holds for some $n \in \mathbb{N}$ s.t. the reachable state set can be calculated in a finite number of steps.

But the reachability problem is undecidable in general!

Decidability

The problem is undecidable already for very restricted subclasses of hybrid automata:

- Stopwatch automata [Čerāns 1992; Wilke 1994; Henzinger, Kopke, Puri, Varaiya 1995]
- 3-dimensional piecewise constant derivative systems [Asarin, Maler, Pnueli 1995]
- ...

Decidable subclasses tend to abandon interplay between changes in continuous dynamics and transition selection/effect, or the dimensionality is extremely low:

- Timed automata [Alur, Dill 1994] and initialized rectangular automata [Henzinger, Kopke, Puri, Varaiya 1995]
- multi-priced timed automata [Larsen, Rasmussen 2005], priced timed automata with pos. and neg. rates [Boyer, Brihaye, Bruyère, Raskin 2007]
- 2-dimensional piecewise constant derivative systems [Maler, Pnueli 1994], also non-deterministic [Asarin, Schneider, Yovine 2001]
- ...

Iterating over the state-space

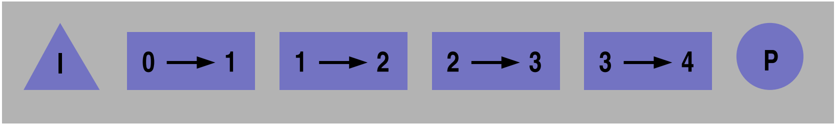
...how do we do this in practice

- on very large state spaces, both continuous and discrete?
- for non-polynomial assignments / pre-post-relations?
- for non-linear differential equations?

SAT Modulo Theory

An engine for bounded model checking of linear hybrid automata

Bounded Model Checking (BMC)



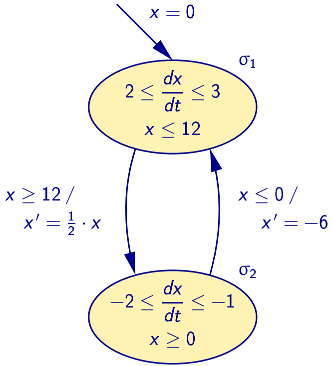
Method:

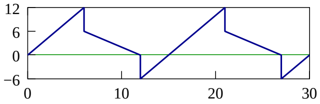
- construct formula that is satisfiable iff error trace of length k exists
- formula is a k–fold unwinding of the system's transition relation, concatenated with a characterization of the initial state(s) and the (unsafe) state to be reached
- use appropriate decision procedure to decide satisfiability of the formula
- usually BMC is carried out incrementally for $k = 0, 1, 2, \ldots$ until an error trace is found or tired

Bounded Model Checking (BMC) algorithm

1. For given $i \in \mathbb{N}$ check for satisfiability of
$$\neg \left(\begin{array}{ll} & init(x_0) \wedge trans(x_0, x_1) \wedge \ldots \wedge trans(x_{i-1}, x_i) \\ \Rightarrow & \phi(x_0) \wedge \ldots \wedge \phi(x_i) \end{array} \right).$$
If test succeeds then report violation of goal.
2. Otherwise repeat with larger i.

BMC of Linear Hybrid Automata





Initial state:

$$\sigma_1^0 \;\wedge\; \neg\sigma_2^0 \;\wedge\; x^0 = 0.0$$

Jumps:

$$\sigma_1^i \wedge \sigma_2^{i+1} \;\rightarrow (x^i \geq 12) \;\wedge\; (x^{i+1} = 0.5 \cdot x^i) \;\wedge\; t^i = 0$$

Flows:

$$\sigma_1^i \wedge \sigma_1^{i+1} \;\rightarrow \begin{cases} & (x^i + 2\,t^i) \;\leq\; x^{i+1} \;\leq\; (x^i + 3\,t^i) \\ \wedge & (x^{i+1} \leq 12) \\ \wedge & (t^i > 0) \end{cases}$$

Quantifier−free Boolean combinations of linear arithmetic constraints over the reals

Parallel composition corresponds to conjunction of formulae

⟶ No need to build product automaton

Ingredients of a Solver for BMC of LHA

BMC of LHA yields very large boolean combination of linear arithmetic facts.

Davis Putnam based SAT-Solver:

- ☺ tackle instances with $\gg 10.000$ variables
- ☺ efficient handling of disjunctions
- ☹ Boolean variables only

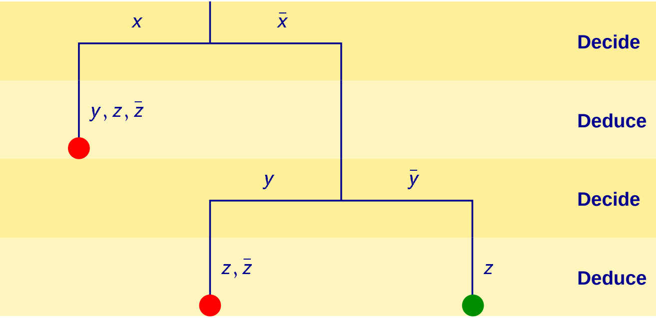
Linear Programming Solver:

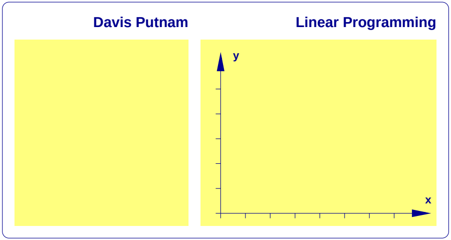
- ☺ solves large conjunctions of linear arithmetic inequations
- ☺ efficient handling of continuous variables ($> 10^6$)
- ☹ no disjunctions

Idea: Combine both methods to overcome shortcomings.
$\leadsto$ **SAT modulo theory**

(Old-fashioned) DPLL Procedure

$$(x \vee y \vee z)$$
$$\wedge\ (\bar{x} \vee y)$$
$$\wedge\ (\bar{y} \vee z)$$
$$\wedge\ (\bar{x} \vee \bar{y} \vee \bar{z})$$
$$\wedge\ (x \vee \bar{y} \vee \bar{z})$$





Input formula:

$$
\begin{aligned}
\Phi = & \ (\overline{e} \rightarrow C \wedge D) \\
& \wedge (\overline{f} \rightarrow A \wedge B) \\
& \wedge (\overline{f} \vee g \vee e) \\
& \wedge (\overline{g} \vee \overline{f}) \\
& \wedge (e \rightarrow (C \vee D) \wedge g) \\
& \wedge (A \rightarrow (4x - 2y \geq 9)) \\
& \wedge (B \rightarrow (2x - 4y \leq -7)) \\
& \wedge (C \rightarrow (x + y \leq 5)) \\
& \wedge (D \rightarrow (x \leq 7))
\end{aligned}
$$

DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT

Davis Putnam

$2e + C + D \geq 2$

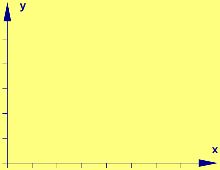
$2f + A + B \geq 2$

$\overline{f} + g + e \geq 1$

$\overline{g} + \overline{f} \geq 1$

$3\overline{e} + 2g + C + D \geq 3$

Linear Programming



Input formula:

$$
\begin{aligned}
\Phi = & (\overline{e} \rightarrow C \wedge D) \\
& \wedge (\overline{f} \rightarrow A \wedge B) \\
& \wedge (\overline{f} \vee g \vee e) \\
& \wedge (\overline{g} \vee \overline{f}) \\
& \wedge (e \rightarrow (C \vee D) \wedge g) \\
& \wedge (A \rightarrow (4x - 2y \geq 9)) \\
& \wedge (B \rightarrow (2x - 4y \leq -7)) \\
& \wedge (C \rightarrow (x + y \leq 5)) \\
& \wedge (D \rightarrow (x \leq 7))
\end{aligned}
$$

DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT

Davis Putnam

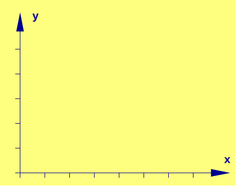
$C + D \geq 2$

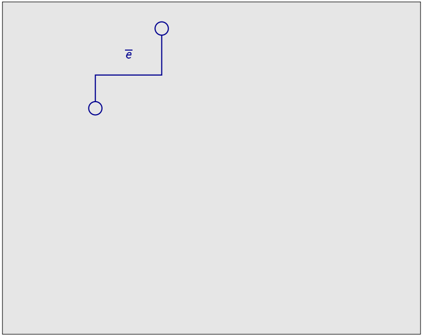
$2f + A + B \geq 2$

$\overline{f} + g \geq 1$

$\overline{g} + \overline{f} \geq 1$

Linear Programming

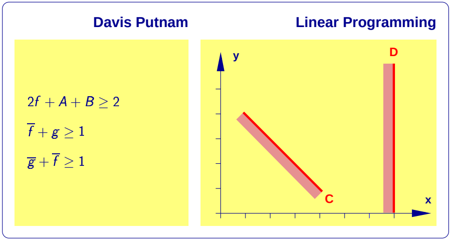


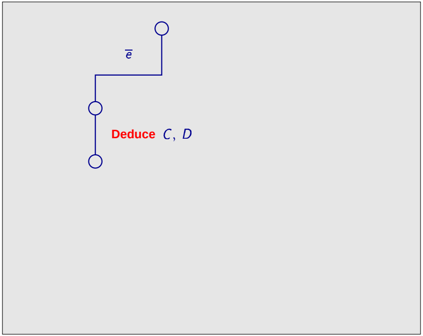


DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT

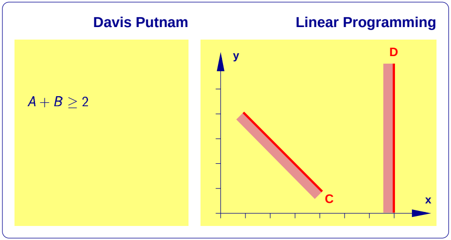


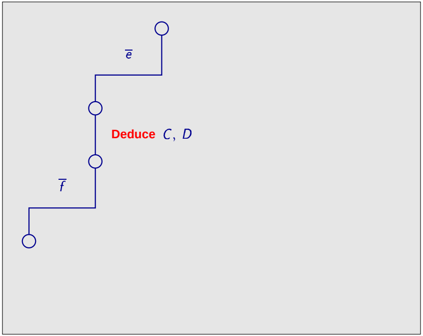


DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT

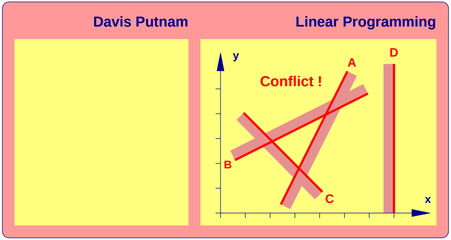




DPLL search

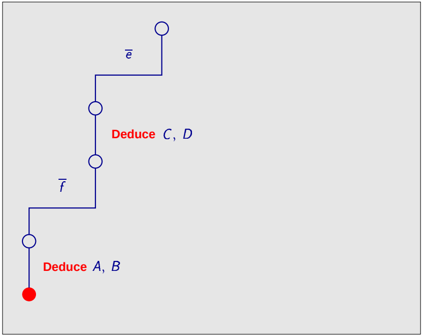
1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT



Irreducible infeasible subsystem is $\{A, B, C\}$

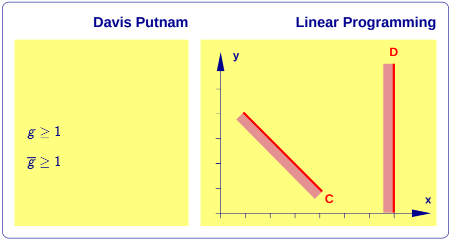
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



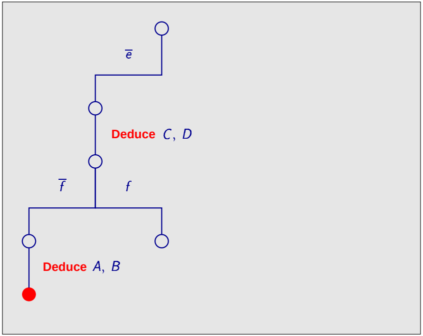
DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT



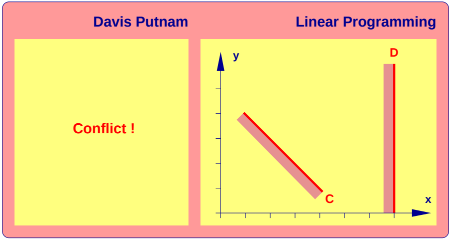
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



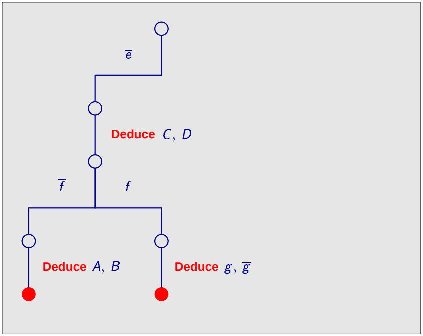
DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT



Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

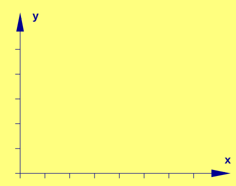
(Simplified) SAT Modulo Theory Scheme: LinSAT

Davis Putnam | **Linear Programming**

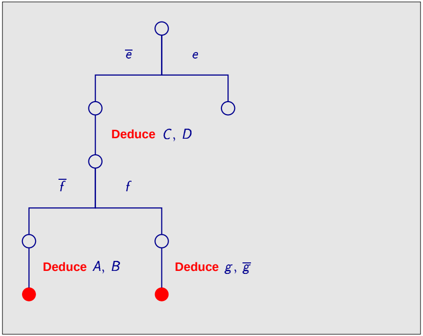
$2f + A + B \geq 2$

$\overline{g} + \overline{f} \geq 1$

$2g + C + D \geq 3$



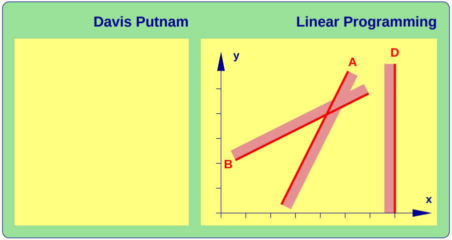
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



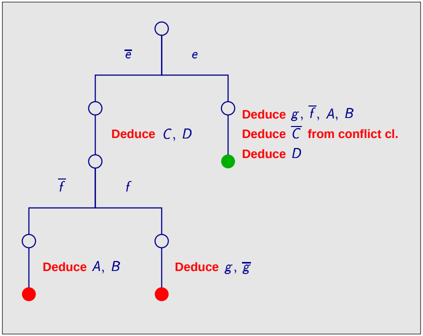
DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT



Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



DPLL search

1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

Deciding the conjunctive T-problems

For T being linear arithmetic over $\mathbb{R}$, this can be done by linear programming:

$$\bigwedge_{i=1}^{n} \sum_{j=1}^{m} A_{i,j} x_j \leq b_j \quad \text{iff} \quad A\mathbf{x} \leq \mathbf{b}$$

$\rightsquigarrow$ Solving LP

$$\begin{array}{rl} \text{maximize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \end{array}$$

with arbitrary $\mathbf{c}$ provides consistency information.

Deciding the conjunctive T-problems (cntd.)

To cope with systems C containing *strict* inequations $\sum_{j=1}^{m} A_{i,j}x_j < b_j$, one

classically: introduces a slack variable ε,

- then replaces $\sum_{j=1}^{m} A_{i,j}x_j < b_j$ by $\sum_{j=1}^{m} A_{i,j}x_j + \varepsilon \leq b_j$,
- solves the resultant LP L, maximizing the objective function ε

$\rightsquigarrow$ C is satisfiable iff L is satisfiable with optimum solution > 0.

more elegantly: treat ε symbolically:

- use 1 and ε as fundamental units of the number system,
- represent all numbers and coefficients in inequations as linear combinations of 1 and ε

[Dutertre, de Moura 2006: Yices]

Extracting reasons for T-conflicts

Goal: In case that the original constraint system

$$C = \begin{pmatrix} & \bigwedge_{i=1}^{k} & \sum_{j=1}^{n} \mathbf{A}_{i,j}\mathbf{x}_j \leq \mathbf{b}_i \\ \wedge & \bigwedge_{i=k+1}^{n} & \sum_{j=1}^{n} \mathbf{A}_{i,j}\mathbf{x}_j < \mathbf{b}_i \end{pmatrix}$$

is infeasible, we want a subset $I \subseteq \{1, \ldots, n\}$ such that

- the subsystem $C|_I$ of the constraint system containing only the conjuncts from I also is infeasible,
- yet the subsystem is *irreducible* in the sense that any proper subset J of I designates a feasible system $C|_J$.

Such an irreducible infeasible subsystem (IIS) is a prime implicant of all the possible reasons for failure of the constraint system C.

Extracting IIS

Provided constraint system C contains only non-strict inequations,

- extraction of IIS can be reduced to finding extremal solutions of a dual system of linear inequations, similar to Farkas' Lemma (Gleeson & Ryan 1990; Pfetsch, 2002)
- to keep the objective function bounded, one can use dual LP

$$\begin{array}{lrcl} \text{maximize} & \mathbf{w}^T\mathbf{y} & & \\ \text{subject to} & \mathbf{A}^T\mathbf{y} & = & 0 \\ & \mathbf{b}^T\mathbf{y} & = & 1 \\ & \mathbf{y} & \geq & 0 \\ \text{where} & \mathbf{w}_i = & \multicolumn{2}{l}{\begin{cases} -1 & \text{if } b_i \leq 0, \\ 0 & \text{if } b_i > 0 \end{cases}} \end{array}$$

- choice of $\mathbf{w}$ guarantees boundedness of objective function

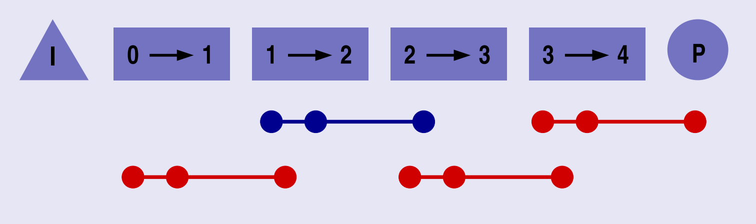
$\Longrightarrow$ optimal solution exists whenever the LP is feasible.

! For such a solution, $I = \{i \mid \mathbf{y}_i \neq 0\}$ is an IIS.

SAT modulo theory for LinSAT

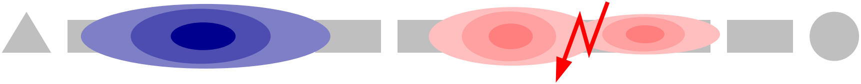
- SAT modulo theory solvers reasoning over linear arithmetic as a theory are readily available: E.g.,
  - LPSAT [Wolfman & Weld, 1999]
  - ICS [Filliatre, Owre, Rueß, Shankar 2001], Simplics [de Moura, Dutertre 2005], Yices [Dutertre, de Moura 2006]
  - MathSAT [Audemard, Bertoli, Cimatti, Kornilowicz, Sebastiani, Bozzano, Juntilla, van Rossum, Schulz 2002–]
  - SVC [Barrett, Dill, Levitt 1996], CVC [Stump, Barrett, Dill 2002], CVC Lite [Barrett, Berezin 2004], CVC3 [Barrett, Fuchs, Ge, Hagen, Jovanovic 2006]
  - HySAT I [Herde & Fränzle, 2004]
  - ...
- Their use for analyzing linear hybrid automata has been advocated a number of times (e.g. in [Audemard, Bozzano, Cimatti, Sebastiani 2004]).
- They combine symbolic handling of discrete state components (via SAT solving) with symbolic handling of continuous state components.
- Formulae arising in BMC have a specific structure, which can be exploited for accelerating SAT search [Strichman 2004]

Pimp my SMT Solver: Isomorphy Inference



- learning schemes employed in SAT solvers account for a major fraction of the running time
- creation of a conflict clause is even more expensive in a combined solver as it entails the extraction of an IIS
- idea: exploit symmetric structure to add isomorphic copies of a conflict clause to the problem
- thus multiplying the benefit taken from the time–consuming reasoning process

Pimp my SMT Solver: Decision Strategies

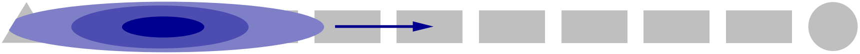


General–Purpose Decision Heuristics:

- distant cycles of the transition relation are being satisfied independently
- until they finally turn out to be incompatible, often entailing the need to backtrack over long distances

For BMC we can try decision strategies respecting the temporal structure!

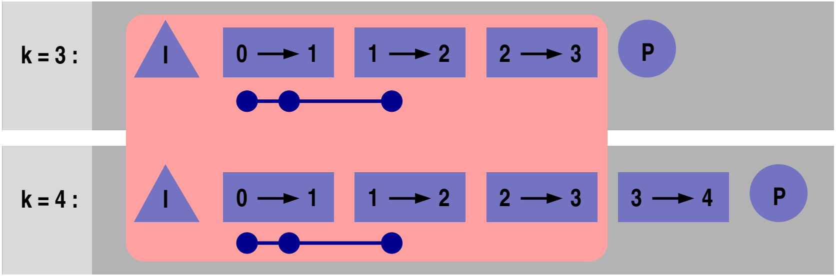
Pimp my SMT Solver: Decision Strategies



Forward–Heuristics:

- select decision variables in the natural order induced by the linear structure of the BMC formula
- e.g. starting with variables from cycle 0, then from cycle 1, 2, etc.
- thereby extending prefixes of legal runs of the system
- allows conflicts to be detected and resolved more locally

Pimp my SMT Solver: Knowledge Reuse

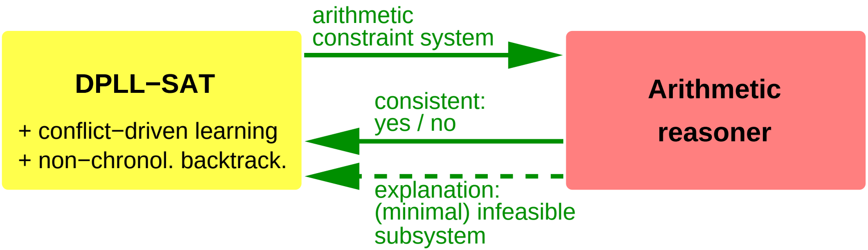


- when carrying out BMC incrementally the consecutive formulas share a large number of clauses
- thus, when moving from instance k to $k + 1$ (or doing them in parallel), we can conjoin the conflict clauses derived when solving the k–instance to the $k + 1$–instance (and vice versa)
- only sound for conflict clauses inferred from clauses which are common to both instances

Satisfiability solving in undecidable arithmetic domains

iSAT algorithm

Classical Lazy TP Layout



Problems with extending it to richer arithmetic domains:

- undecidability: answer of arithmetic reasoner no longer two-valued; don't know cases arise
- explanations: how to generate (nearly) minimal infeasible subsystems of undecidable constraint systems?

The Task

Find satisfying assignments (or prove absence thereof) for large (thousands of Boolean connectives) formulae of shape

$$
\begin{array}{ll}
 & (b_1 \implies x_1^2 - \cos y_1 < 2y_1 + \sin z_1 + e^{u_1}) \\
\wedge & (x_5 = \tan y_4 \vee \tan y_4 > z_4 \vee \ldots) \\
\wedge & \ldots \\
\wedge & (\frac{dx}{dt} = -\sin x \wedge x_3 > 5 \wedge x_3 < 7 \wedge x_4 > 12 \wedge \ldots) \\
\wedge & \ldots
\end{array}
$$

Conventional solvers

- do either address much smaller fragments of arithmetic
  - decidable theories: no transcendental fct.s, no ODEs
- or tackle only small formulae
  - some dozens of Boolean connectives.

Algorithmic basis:

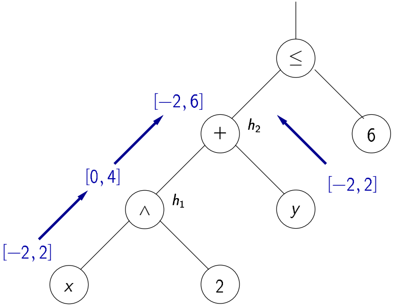
Interval constraint propagation (Hull consistency version)

Interval Constraint Solving (1)

- Complex constraints are rewritten to "triplets" (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{lll} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- "Forward" interval propagation yields justification for constraint satisfaction:



$x \in [-2, 2]$
$\wedge\ y \in [-2, 2]$

$\Downarrow$

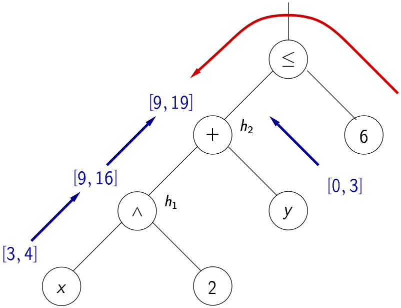
$h_2 \leq 6$ is satisfied in box

Interval Constraint Solving (1)

- Complex constraints are rewritten to "triplets" (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rcl} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval propagation (fwd & bwd) yields witness for unsatisfiability:



$x \in [3, 4]$
$\wedge\ y \in [0, 3]$

$\Downarrow$

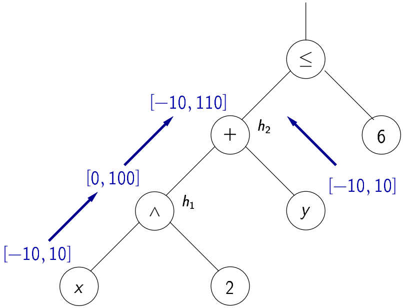
$h_2 \leq 6$ is unsat. in box

Interval Constraint Solving (1)

- Complex constraints are rewritten to "triplets" (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{lll} c_1 : & & h_1 \triangleq x \wedge 2 \\ c_2 : & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fixpoint is reached) yields contraction of box:



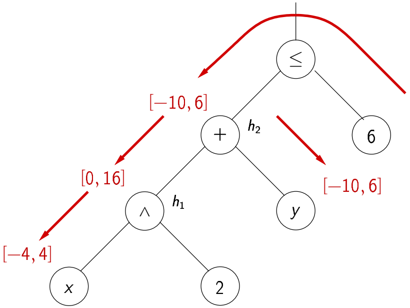
$$\begin{array}{rl} & x \in [-10, 10] \\ \wedge & y \in [-10, 10] \end{array}$$

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rll} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fixpoint is reached) yields contraction of box:



$$x \in [-10, 10] \wedge y \in [-10, 10]$$

$$\Downarrow$$

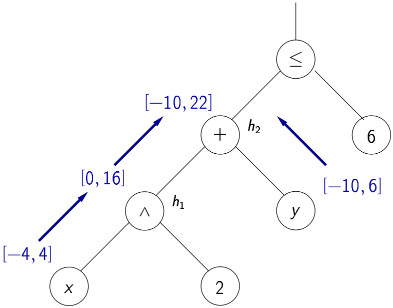
$$x \in [-4, 4] \wedge y \in [-10, 6]$$

Interval Constraint Solving (1)

- Complex constraints are rewritten to “triplets” (primitive constraints):

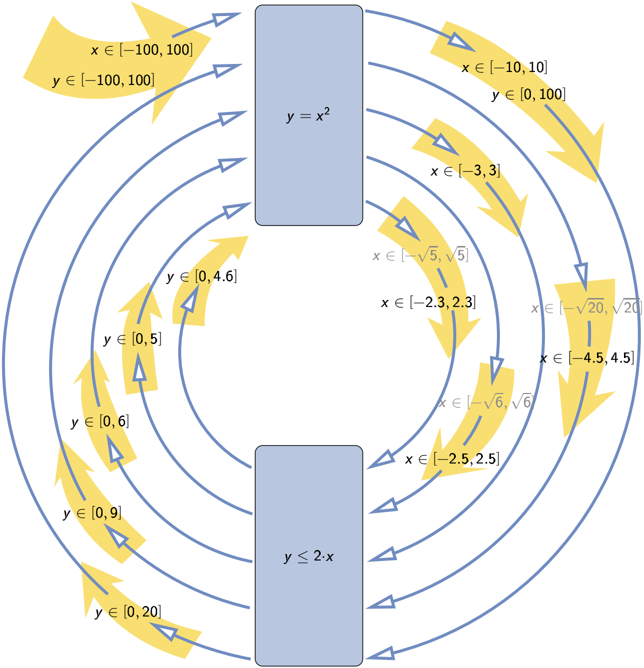
$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{lll} c_1 : & & h_1 \triangleq x \wedge 2 \\ c_2 : & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

- Interval prop. (fwd & bwd until fixpoint is reached) yields contraction of box:



Constraint is not satisfied by the contracted box!

$$x \in [-4, 4]$$
$$\wedge \ y \in [-10, 6]$$



x ∈ [−100, 100]
y ∈ [−100, 100]
y = x^2
x ∈ [−10, 10]
y ∈ [0, 100]
x ∈ [−3, 3]
x ∈ [−√5, √5]
x ∈ [−2.3, 2.3]
x ∈ [−√20, √20]
x ∈ [−4.5, 4.5]
x ∈ [−√6, √6]
x ∈ [−2.5, 2.5]
y ∈ [0, 4.6]
y ∈ [0, 5]
y ∈ [0, 6]
y ∈ [0, 9]
y ∈ [0, 20]
y ≤ 2·x

Interval contraction

Backward propagation yields rectangular overapproximation of non-rectangular pre-images.

Thus, interval contraction provides a highly incomplete deduction system:

$$\begin{array}{ll} & x \in [0,\infty) \\ \wedge & h \hat{=} x \cdot y \\ \wedge & h > 5 \end{array} \quad \Longrightarrow \quad \begin{array}{ll} & x \in (0,\infty) \\ \wedge & y \in (0,\infty) \end{array} \quad \Longrightarrow \quad h \in (0,\infty) \quad \not\Longrightarrow \quad h > 5$$

$\rightsquigarrow$ enhance through branch-and-prune approach.

Schematic Interval-CP based CS Alg. / DPLL

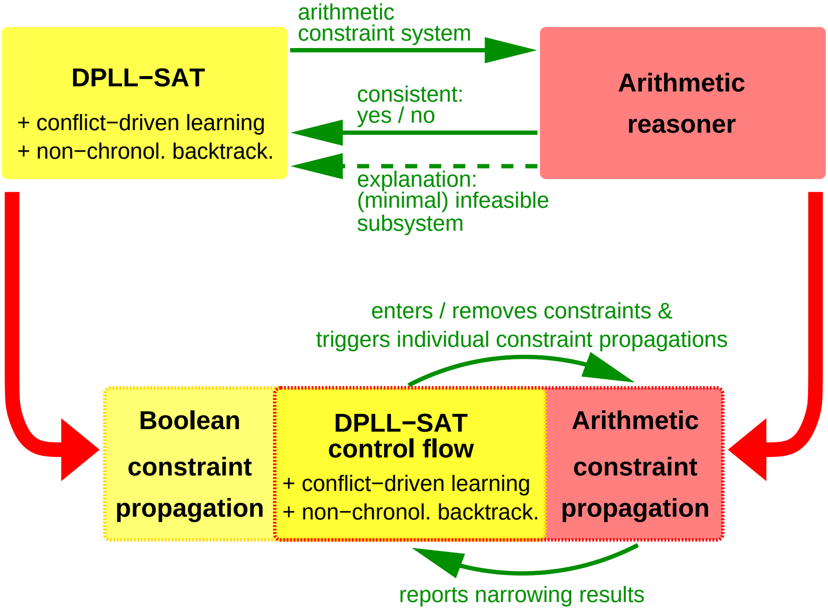
Given: Constraint / clauseset $C = \{c_1, \ldots, c_n\}$,
initial box (= cartesian product of intervals) B in $\mathbb{R}^{|\mathrm{free(C)}|}$ / $\mathbb{B}^{|\mathrm{free(C)}|}$

Goal: Find box $B' \subseteq B$ containing satisfying valuations throughout *or* show non-existence of such B'.

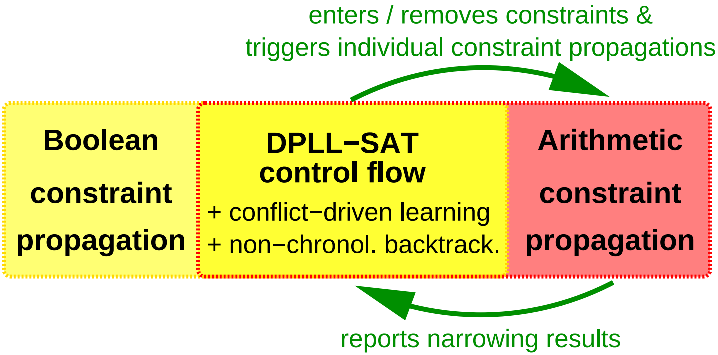
Alg.:

1. $L := \{B\}$
2. If $L \neq \emptyset$ then take some box $b :\in L$, (LIFO) otherwise report "unsatisfiable" and stop.
3. Use contraction to determine a sub-box $b' \subseteq b$. (Unit Prop.)
4. If $b' = \emptyset$ then set $L := L \setminus \{b\}$, goto 2.
5. Use forward interval propagation to determine whether all constraints are satisfied throughout b'; if so then report b' as satisfying and stop.
6. If $b' \subset b$ then set $L := L \setminus \{b\} \cup \{b'\}$, goto 2.
7. Split b into subboxes b_1 and b_2, set $L := L \setminus \{b\} \cup \{b_1, b_2\}$, goto 2.

Lazy TP: Tightening the Interaction



Properties of Modified Layout



- SAT engine has introspection into CP
- thus can keep track of inferences *and their reasons*
- ☺ can use recent SAT mechanisms for generalizing reasons of conflicts and learning them, thus pruning the search tree

How iSAT works

c_1 :	$(\neg a \lor \neg c \lor d)$
c_2 :	$\land\ (\neg a \lor \neg b \lor c)$
c_3 :	$\land\ (\neg c \lor \neg d)$
c_4 :	$\land\ (b \lor x \geq -2)$
c_5 :	$\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$
c_6 :	$\land\ h_1 = x^2$
c_7 :	$\land\ h_2 = -2 \cdot y$
c_8 :	$\land\ h_3 = h_1 + h_2$

- Use Tseitin-style (i.e. definitional) transformation to rewrite input formula into a conjunction of constraints:
  - ▷ n-ary disjunctions of bounds
  - ▷ arithmetic constraints having at most one operation symbo
- Boolean variables are regarded as 0-1 integer variables. Allows identification of literals with bounds on Booleans:

$$b \equiv b \geq 1$$
$$\neg b \equiv b \leq 0$$

- Float variables h_1, h_2, h_3 are used for decomposition of complex constraint $x^2 - 2y \geq 6.2$.

How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

c_4 : $\land\ (b \lor x \geq -2)$

c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

c_8 : $\land\ h_3 = h_1 + h_2$

DL 1: $a \geq 1$

How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

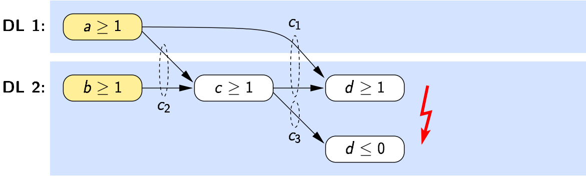
c_4 : $\land\ (b \lor x \geq -2)$

c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

c_8 : $\land\ h_3 = h_1 + h_2$



How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

c_4 : $\land\ (b \lor x \geq -2)$

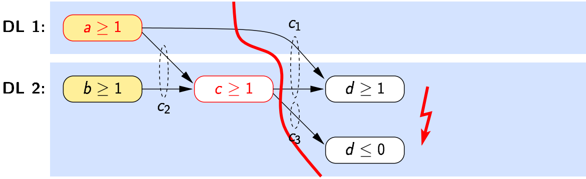
c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

c_8 : $\land\ h_3 = h_1 + h_2$

c_9 : $\land\ (\neg a \lor \neg c)$



How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

c_4 : $\land\ (b \lor x \geq -2)$

c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

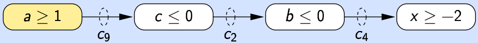
c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

c_8 : $\land\ h_3 = h_1 + h_2$

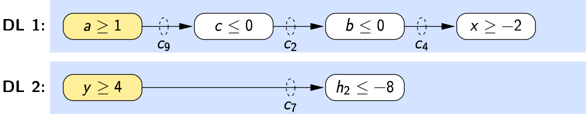
c_9 : $\land\ (\neg a \lor \neg c)$

DL 1:



How iSAT works

$$
\begin{aligned}
c_1 &: \quad (\neg a \vee \neg c \vee d) \\
c_2 &: \wedge\ (\neg a \vee \neg b \vee c) \\
c_3 &: \wedge\ (\neg c \vee \neg d) \\
c_4 &: \wedge\ (b \vee x \geq -2) \\
c_5 &: \wedge\ (x \geq 4 \vee y \leq 0 \vee h_3 \geq 6.2) \\
c_6 &: \wedge\ h_1 = x^2 \\
c_7 &: \wedge\ h_2 = -2 \cdot y \\
c_8 &: \wedge\ h_3 = h_1 + h_2 \\
c_9 &: \wedge\ (\neg a \vee \neg c)
\end{aligned}
$$



How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

c_4 : $\land\ (b \lor x \geq -2)$

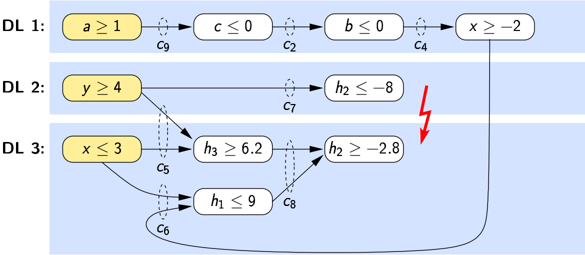
c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

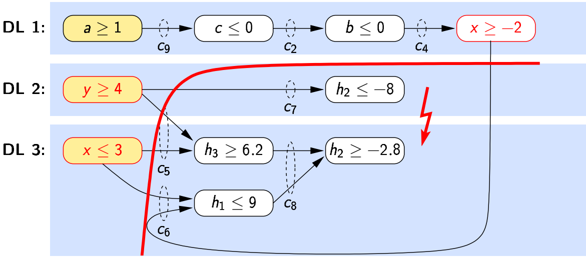
c_8 : $\land\ h_3 = h_1 + h_2$

c_9 : $\land\ (\neg a \lor \neg c)$



How iSAT works

c_1 :	$(\neg a \lor \neg c \lor d)$
c_2 :	$\land\ (\neg a \lor \neg b \lor c)$
c_3 :	$\land\ (\neg c \lor \neg d)$
c_4 :	$\land\ (b \lor x \geq -2)$
c_5 :	$\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$
c_6 :	$\land\ h_1 = x^2$
c_7 :	$\land\ h_2 = -2 \cdot y$
c_8 :	$\land\ h_3 = h_1 + h_2$
c_9 :	$\land\ (\neg a \lor \neg c)$
c_{10} :	$\land\ (x < -2 \lor y < 3 \lor x > 3)$



← conflict clause = symbolic description

of a rectangular region of the search space

which is excluded from future search

How iSAT works

c_1 : $(\neg a \lor \neg c \lor d)$

c_2 : $\land\ (\neg a \lor \neg b \lor c)$

c_3 : $\land\ (\neg c \lor \neg d)$

c_4 : $\land\ (b \lor x \geq -2)$

c_5 : $\land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

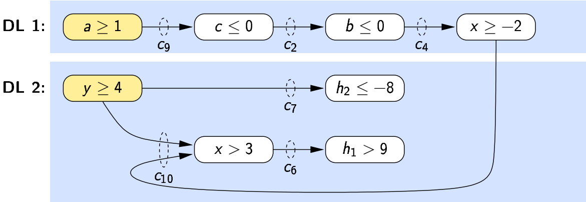
c_6 : $\land\ h_1 = x^2$

c_7 : $\land\ h_2 = -2 \cdot y$

c_8 : $\land\ h_3 = h_1 + h_2$

c_9 : $\land\ (\neg a \lor \neg c)$

c_{10} : $\land\ (x < -2 \lor y < 3 \lor x > 3)$



How iSAT works

$c_1 :\quad (\neg a \lor \neg c \lor d)$

$c_2 :\quad \land\ (\neg a \lor \neg b \lor c)$

$c_3 :\quad \land\ (\neg c \lor \neg d)$

$c_4 :\quad \land\ (b \lor x \geq -2)$

$c_5 :\quad \land\ (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

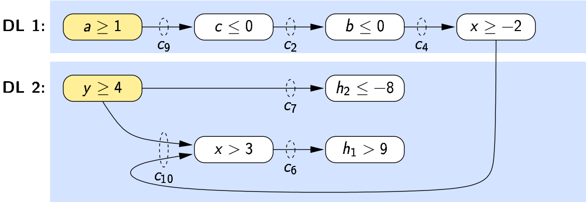
$c_6 :\quad \land\ h_1 = x^2$

$c_7 :\quad \land\ h_2 = -2 \cdot y$

$c_8 :\quad \land\ h_3 = h_1 + h_2$

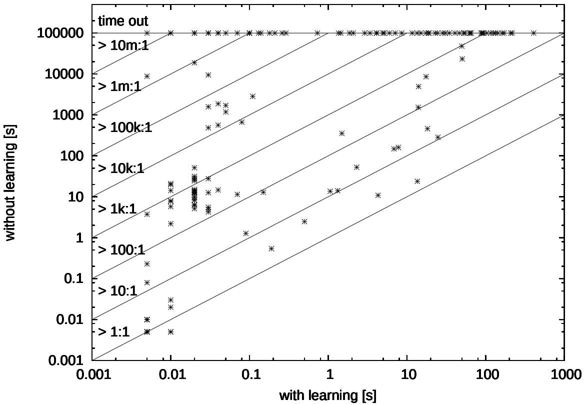
$c_9 :\quad \land\ (\neg a \lor \neg c)$

$c_{10} :\quad \land\ (x < -2 \lor y < 3 \lor x > 3)$



- Continue do split and deduce until either
  - ▷ formula turns out to be UNSAT (unresolvable conflict)
  - ▷ solver is left with 'sufficiently small' portion of the search space for which it cannot derive any contradiction
- Avoid infinite splitting and deduction:
  - ▷ minimal splitting width
  - ▷ discard a deduced bound if it yields small progress only

The Impact of Learning: Runtime



Examples:
BMC of

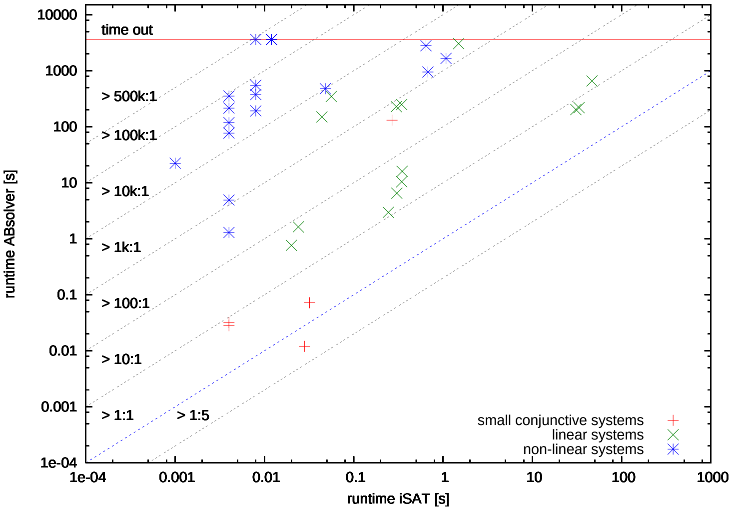
- platoon ctrl.
- bounc. ball
- gingerbread map
- oscillatory logistic map

Intersect. of geometric bodies

Size:
Up to 2400 var.s,
$\gg 10^3$ Boolean connectives.

[2.5 GHz AMD Opteron, 4 GByte physical memory, Linux]

The Competition: ABsolver



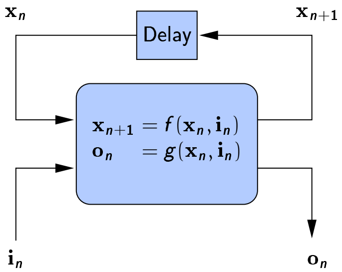
ABsolver: Bauer, Pister, Tautschnig, "Tool support for the analysis of hybrid systems and models", DATE '07

Hybrid BMC in Practice

ETCS Train separation in HySAT II

Bounded Model Checking of Hybrid Systems (1)

Given:



Non-linear discrete-time hybrid dynamical system

$\mathbf{x}$	—	state vector
$\mathbf{i}$	—	input vector
$\mathbf{o}$	—	output vector
f	—	next-state function
g	—	output function

f, g potentially non-linear.

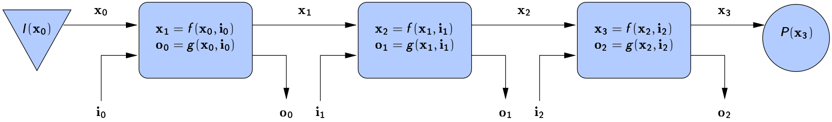
Goal:

Check whether some unsafe state is reachable within k steps of the system

Bounded Model Checking of Hybrid Systems (2)

Method:

- Construct formula that is satisfiable if error trace of length k exists
- Formula is a k–fold unrolling of the transition relation, concatenated with a characterization of the initial state(s) and the (unsafe) state to be reached

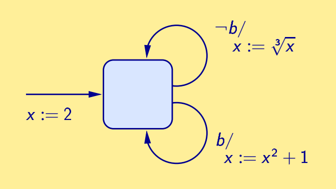


- Use appropriate decision procedure to decide satisfiability of the formula

Needed:

Solvers for large, non-linear arithmetic formulae with a rich Boolean structure

Bounded Model Checking with HySAT



Safety property:
There's no sequence of input values such that $3.14 \leq x \leq 3.15$

```
DECL
    boole b;
    float [0.0, 1000.0] x;
INIT
    - Characterization of initial state.
    x = 2.0;
TRANS
    - Transition relation.
    b -> x' = x^2 + 1;
    !b -> x' = nrt(x, 3);
TARGET
    - State(s) to be reached.
    x >= 3.14 and x <= 3.15;
```

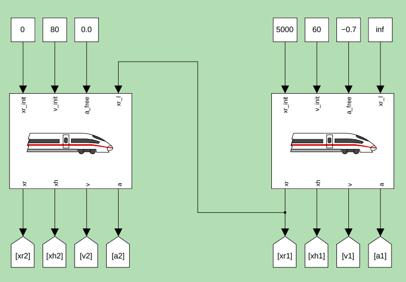
HySAT

COUNTEREXAMPLE

```
SOLUTION:
    b (boole):
        @0: [0, 0]
        @1: [1, 1]
        @2: [1, 1]
        @3: [0, 0]
        @4: [1, 1]
        @5: [1, 1]
        @6: [0, 0]
        @7: [1, 1]
        @8: [0, 0]
        @9: [1, 1]
        @10: [1, 1]
        @11: [0, 0]

    x (float):
        @0: [2, 2]
        @1: [1.25992, 1.25992]
        @2: [2.5874, 2.5874]
        @3: [7.69464, 7.69464]
        @4: [1.97422, 1.97422]
        @5: [4.89756, 4.89756]
        @6: [24.9861, 24.9861]
        @7: [2.92347, 2.92347]
        @8: [9.5467, 9.5467]
        @9: [2.12138, 2.12138]
        @10: [5.50024, 5.50024]
        @11: [31.2526, 31.2526]
        @12: [3.14989, 3.14989]
```

BMC of Matlab/Simulink Model

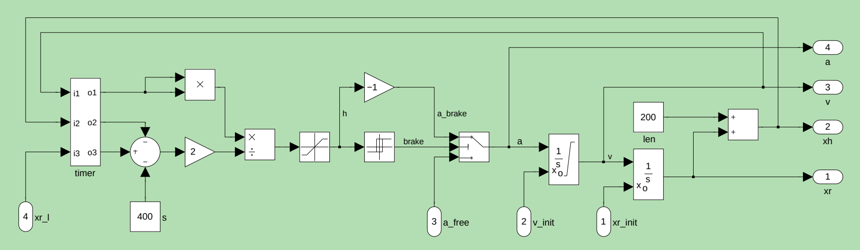
Example: Train Separation in Absolute Braking Distance



Minimal admissible distance d between two successive trains equals braking distance d_b of the second train plus a safety distance S.

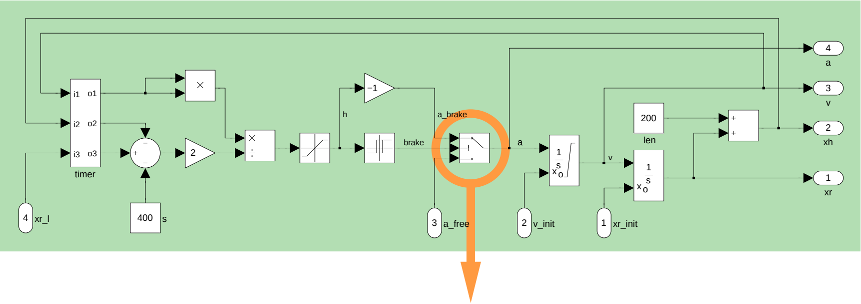
First train reports position of its tail to the second train every 8 seconds.
Controller in second train automatically initiates braking to maintain a safe distance.

Model of Controller & Train Dynamics



Property to be checked: Does the controller guarantee that collisions don't occur in any possible scenario of use?

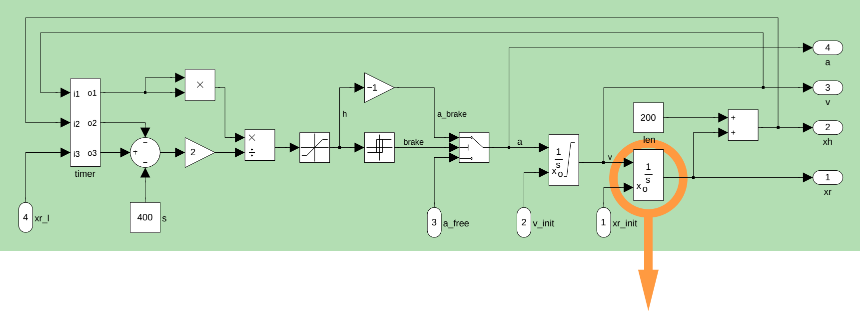
Translation to HySAT




```
- Switch block: Passes through the first input or the third input
- based on the value of the second input.

 brake -> a = a_brake;
!brake -> a = a_free;
```

BMC of Matlab/Simulink Model

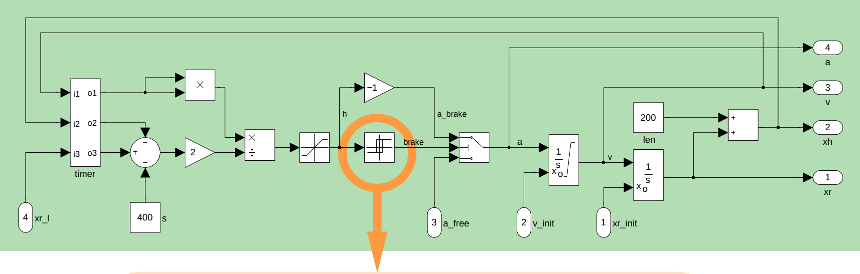
Translation to HySAT




```
- Euler approximation of integrator block
xr' = xr + dt * v;
```

BMC of Matlab/Simulink Model

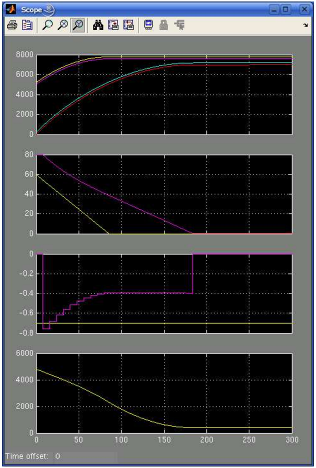
Translation to HySAT



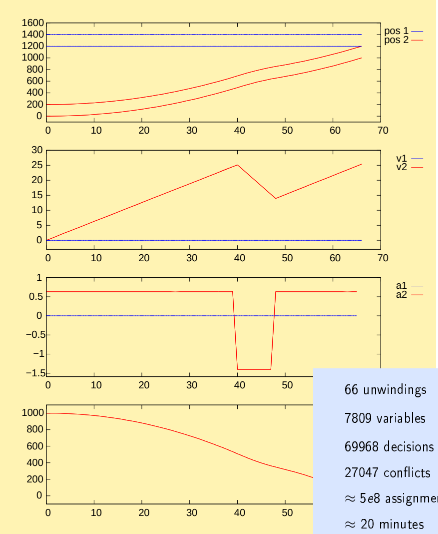

```
- Relay block: When the relay is on, it remains on until the input
- drops below the value of the switch off point parameter. When the
- relay is off, it remains off until the input exceeds the value of
- the switch on point parameter.

(!is_on and h >= param_on ) -> ( is_on' and  brake);
(!is_on and h <  param_on ) -> (!is_on' and !brake);
( is_on and h <= param_off) -> (!is_on' and !brake);
( is_on and h >  param_off) -> ( is_in' and  brake);
```

BMC of Matlab/Simulink Model

Simulation of the Model



Error Trace found by HySAT



66 unwindings

7809 variables

69968 decisions

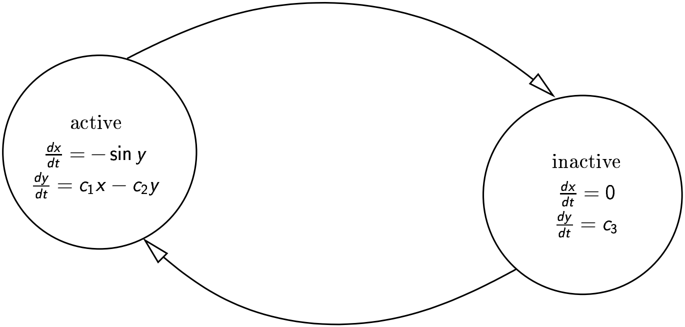
27047 conflicts

≈ 5e8 assignments

≈ 20 minutes

Direct reasoning over images and pre-images of ODEs

Motivation



- Linear and non-linear ordinary Differential Equations (ODEs) describing continous behaviour in the discrete modes of a hybrid system
- Want to do BMC on these models w/o prior hybridisation

The Problem

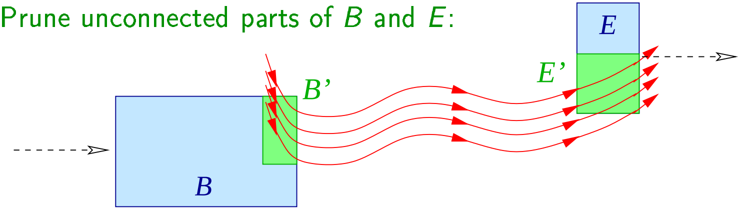
Given: a system of time-invariant ODEs

$$\frac{dx_1}{dt} = f_1(x_1, \ldots, x_n)$$
$$\vdots$$
$$\frac{dx_n}{dt} = f_n(x_1, \ldots, x_n)$$

plus three boxes $B, I, E \subset \mathbb{R}^n$.

Problem: determine whether E is reachable from B along a trajectory satisfying the ODE and not leaving I.

Added value: Prune unconnected parts of B and E:



Problem: Safely determine whether E is unreachable from B along a trajectory satisfying the ODE and not leaving I.

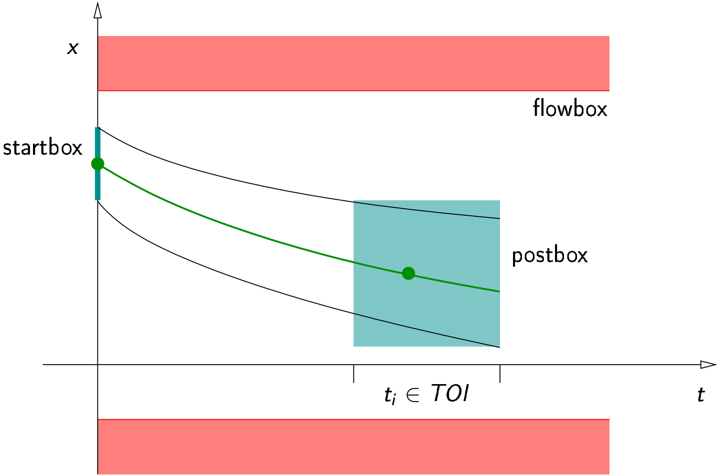
Some approaches:

1. Interval-based safe numeric approximation of ODEs [Moore 1965, Lohner 1987, Stauning 1997]

   (used in Hypertech [Henzinger, Horowitz, Majumdar, Wong-Toi 2000])

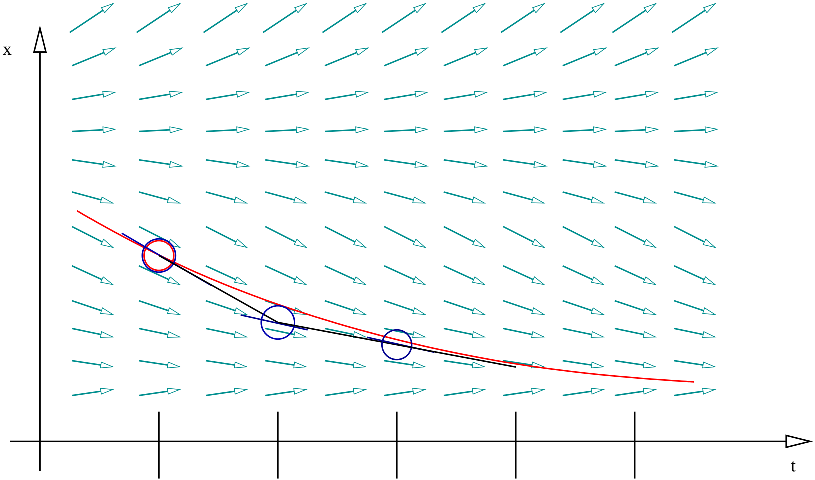
2. CLP(F): a symbolic, constraint-based technology for reasoning about ODEs grounded in (in-)equational constraints obtained from Taylor expansions [Hickey, Wittenberg 2004]

Safe Approximation



Should also be tight! And efficient to compute!

Euler's Method



Taylor Series

Exact solution $x(t)$ has slope determined by f in each point: $\frac{dx}{dt} = f(x(t))$
Taylor expansion of exact solution:

$$
\begin{aligned}
x(t_0 + h) = & x(t_0) + \frac{h^1}{1!}\frac{dx}{dt}(t_0) \\
& + \frac{h^2}{2!}\frac{d^2x}{dt^2}(t_0) + \dots \\
& + \frac{h^n}{n!}\frac{d^nx}{dt^n}(t_0) \\
& + \frac{\mathbf{h}^{\mathbf{n}+1}}{(\mathbf{n}+1)!}\frac{\mathbf{d}^{\mathbf{n}+1}\mathbf{x}}{\mathbf{dt}^{\mathbf{n}+1}}(\mathbf{t_0} + \theta\mathbf{h}), \text{ with } \mathbf{0} < \theta < \mathbf{1}
\end{aligned}
$$

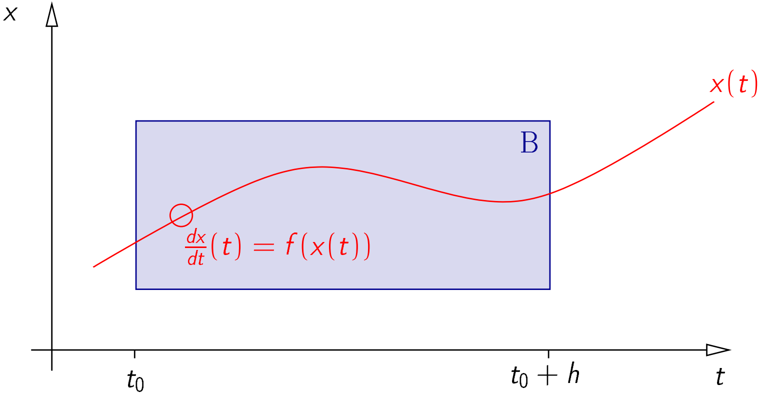
(Lagrange Remainder)

Taylor Series

$$\begin{aligned} x(t_0+h) =& x(t_0) + \frac{h^1}{1!}\underbrace{\frac{dx}{dt}(t_0)}_{f(x(t_0))} \\ &+ \frac{h^2}{2!}\underbrace{\frac{d^2x}{dt^2}(t_0)}_{\frac{\mathrm{d}f}{\mathrm{d}t}(x(t_0))\cdot f(x(t_0))} + \ldots \\ &+ \frac{h^n}{n!}\frac{d^nx}{dt^n}(t_0) \\ &+ \frac{h^{n+1}}{(n+1)!}\underbrace{\frac{d^{n+1}x}{dt^{n+1}}(t_0+\theta h)}_{\text{unknown}}, \text{ with } 0<\theta<1 \end{aligned}$$

Can use interval arithm. to evaluate $f(x(t_0))$, etc., if $x(t_0)$ is set-valued!

Bounding Box



$$\begin{aligned} \frac{dx}{dt}(t) &\le \max(f(B)) \\ \frac{dx}{dt}(t) &\ge \min(f(B)) \end{aligned} \quad \text{for all } t \in [t_0, t_0 + h]$$

If we only knew B...

Bounding Box [Lohner]

Given: Initial value problem:

$$\frac{dx}{dt} = f(x),\ x(t_0) = x_0 \text{ may also be a box}$$

Theorem (Lohner): If

$$[B^1] := x_0 + [0,h] \cdot f([B^0])$$

and

$$[B^1] \subseteq [B^0]$$

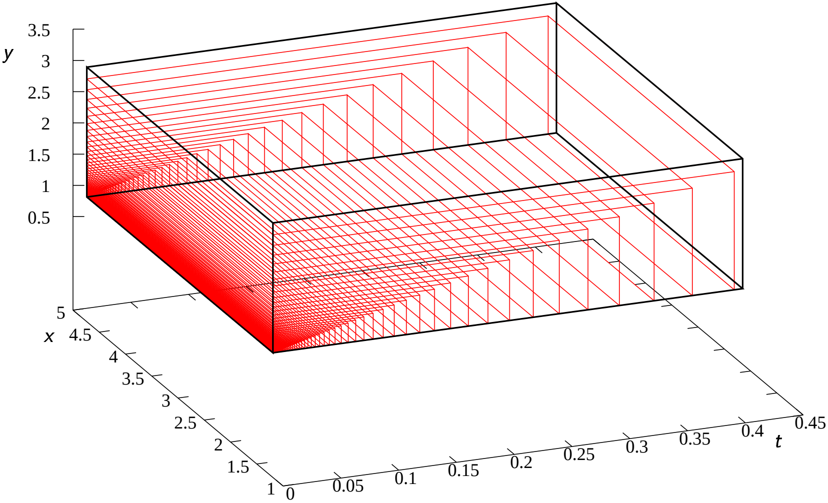
then the initial value problem above has exactly one solution over $[t_0, t_0 + h]$ which lies entirely within $[B^1]$ $\longrightarrow$ Bounding Box.

Algorithm

To get an enclosure . . .

- Determine bounding box and stepsize
- Evaluate Taylor series up to desired order over startbox
- Evaluate remainder term over bounding box

Bounding Box

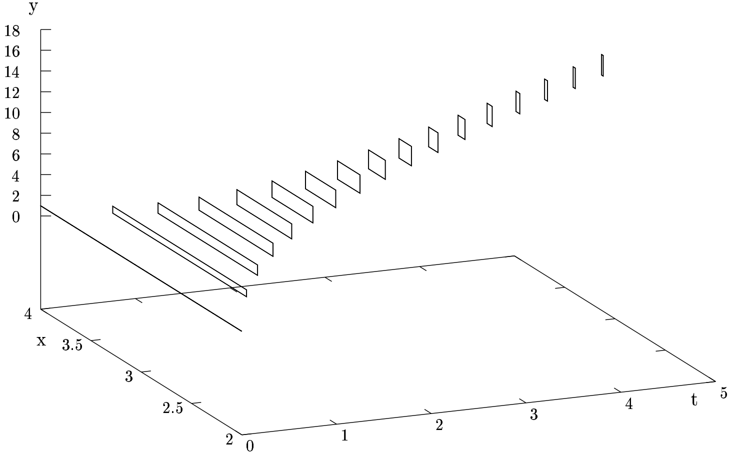


Algorithm

- Find **bounding box** with greedy algorithm
- Generate **derivatives** symbolically
- Simplify expressions to reduce alias effects on variables
- **Evaluate expressions** with interval arithmetic
  - Taylor series
  - Lagrange remainder

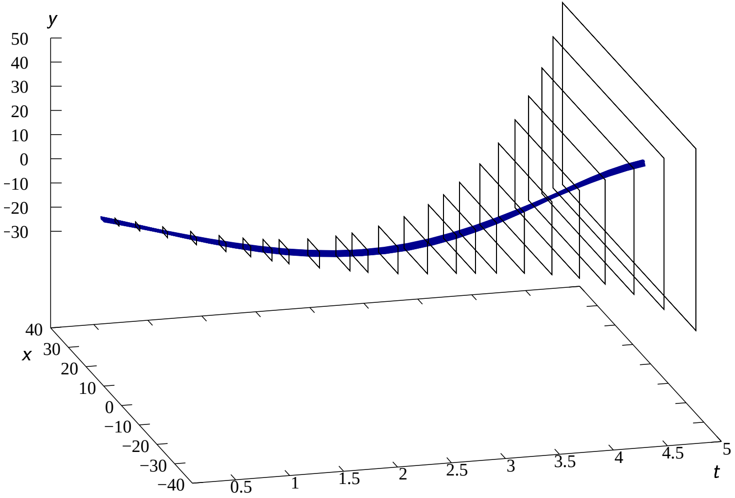
Example

$$\frac{dx}{dt} = -x + 3,\ \frac{dy}{dt} = x,\ x_0 = [2,4],\ y_0 = [1,1]$$



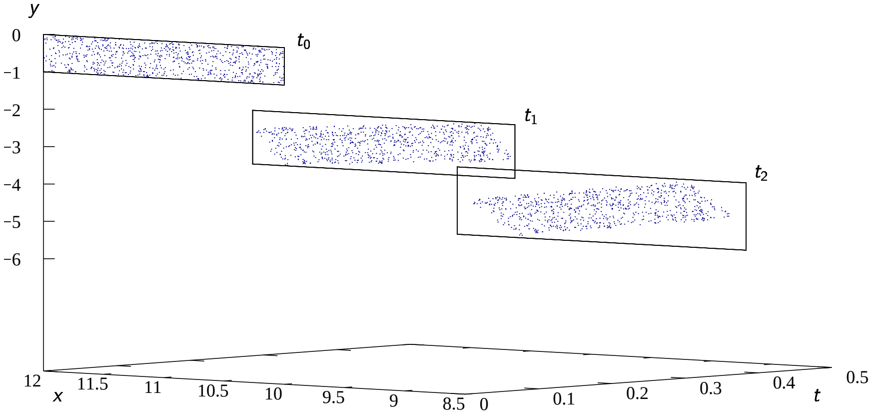
Example II: Stable Oscillator

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$$



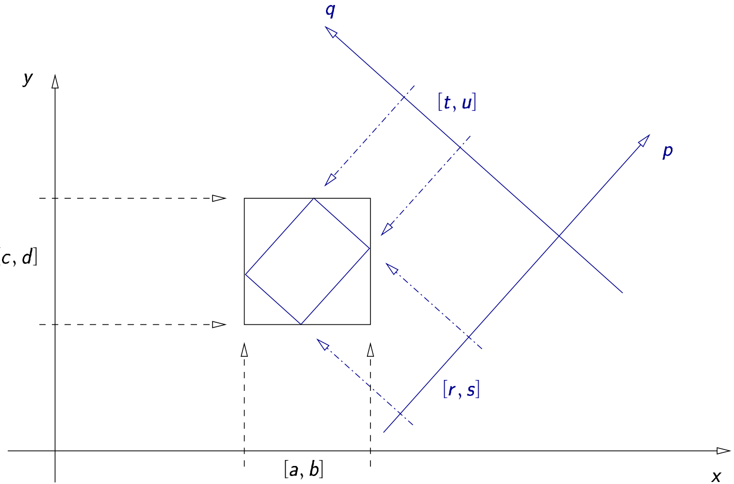
Wrapping Effect

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$$



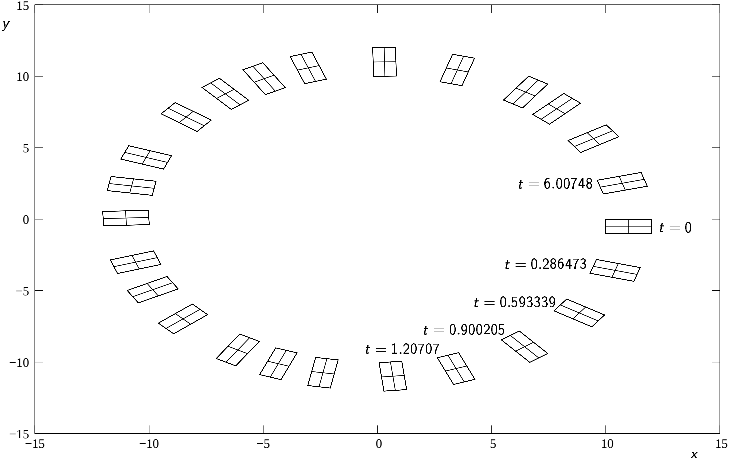
Fight Wrapping Effect

Lohner, Stauning, . . . : use **coordinate transformation**



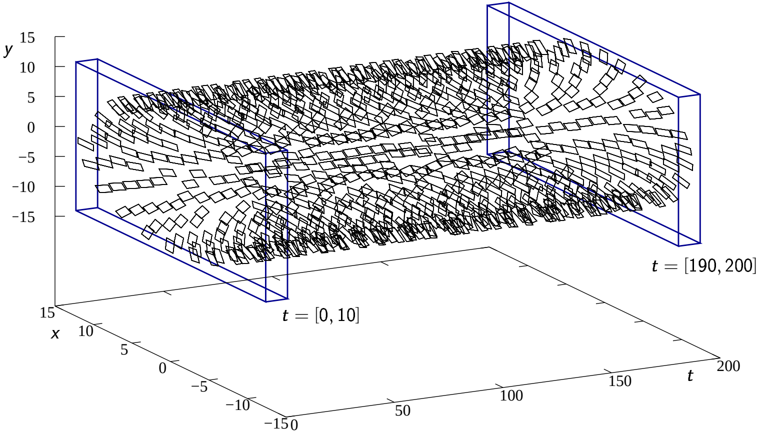
Stable Oscillator

$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$

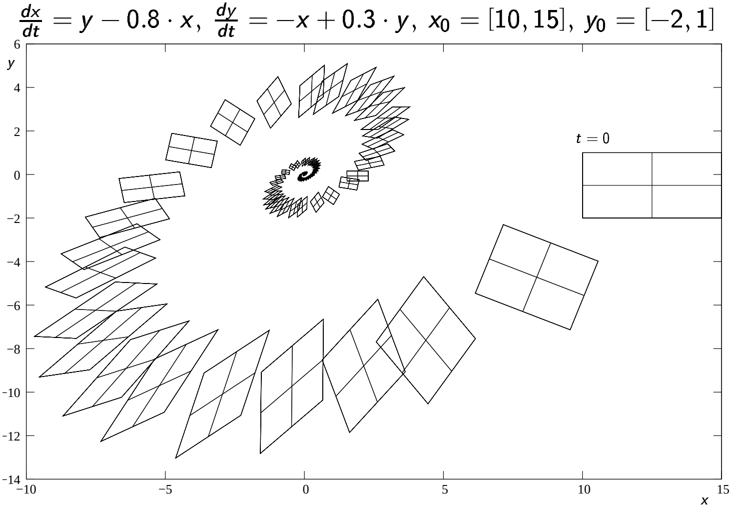


Stable Oscillator

$$\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x,\ x_0 = [10, 12],\ y_0 = [-1, 0]$$

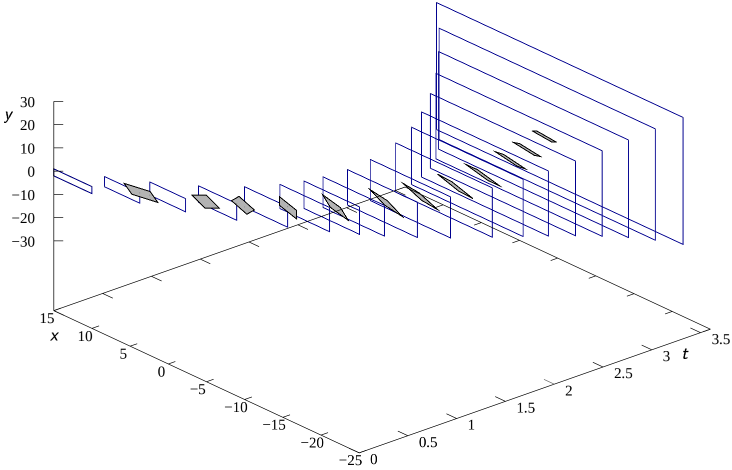


Damped Oscillator



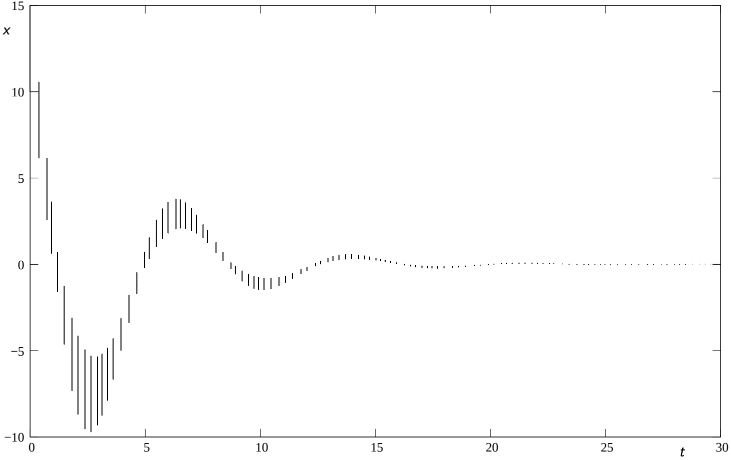
Damped Oscillator

$$\frac{dx}{dt} = y - 0.8 \cdot x,\ \frac{dy}{dt} = -x + 0.3 \cdot y,\ x_0 = [10, 15],\ y_0 = [-2, 1]$$

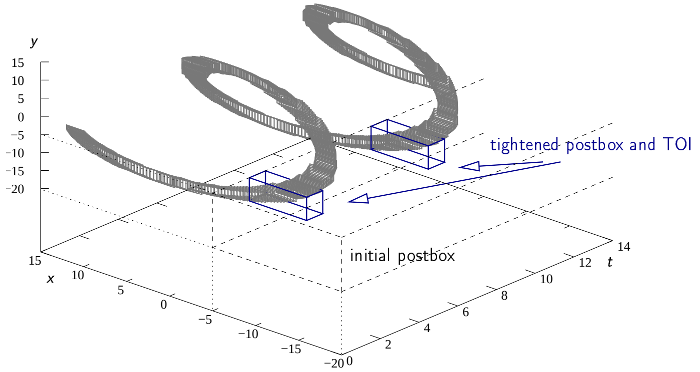


Damped Oscillator

$\frac{dx}{dt} = y - 0.8 \cdot x,\ \frac{dy}{dt} = -x + 0.3 \cdot y,\ x_0 = [10, 15],\ y_0 = [-2, 1]$



Use in ICP: Tighten Target Box



- Given target box (including phase space and time)
- Intersect target box with enclosure
- Remove elements with empty intersection
  (narrows also time-window of interest)

Backward Propagation

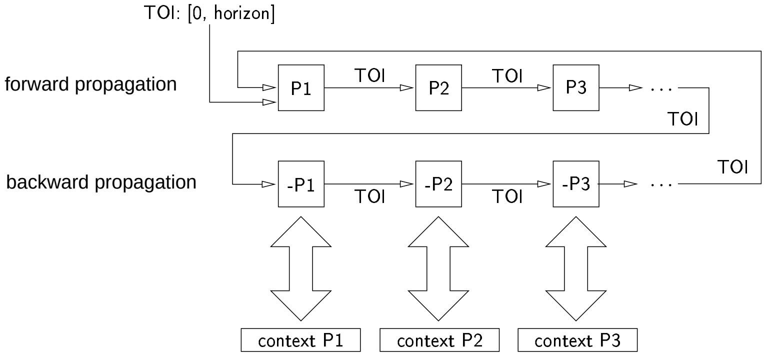
- Use temporally reversed ODEs
- Use start box as target box and do normal forward propagation
- Intersect resulting target box with original start box

Fwd. and bwd. propagation do

- narrow the start box B and target box E — also iteratively!
- narrow the time window for both B and E,
- thus give fresh meat to constraint propagation along adjacent parts of the transition sequence!

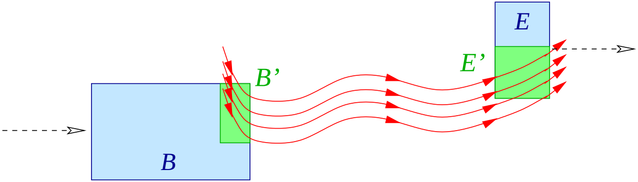
Controlling Complexity: Partitioning

- Partition ODEs: Group together ODEs with common variables
- Deduction process alternates between different partitions and between forward and backward pruning:



Summary

- Taylor-based numerical method with error enclosure
- Tightly integrated with non-linear arithmetic constraint solving:
  - provides an interval contractor, just like ICP



  - temporally symmetric (fwd. and bwd. contraction), unlike traditional image computation
  - refutes trajectory bundles based on partial knowledge
- experimental: first proof-of-concept implemented.
  [Eggers, Fränzle, Herde, ATVA 2008]

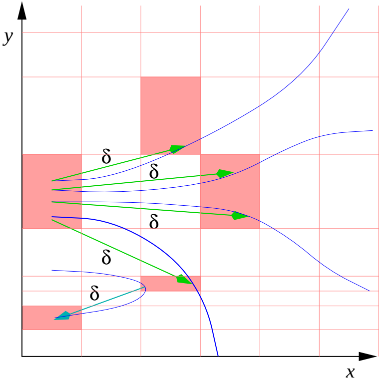
Other Approaches to ODE Analysis

Automatic derivation of
safe finite-state approximations
&
Mechanized Lyapunov-based methods

Model-checking through discretization

Idea:

Hybrid automata are mapped to finite state through overapproximation, then subjected to finite-state symbolic model-checking



Problems:

- *effective construction* of the overapproximation
- find *appropriate discretization* (avoid "false negatives")

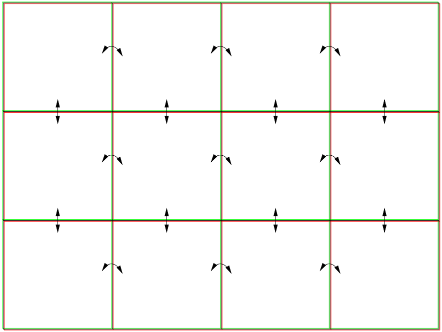
HSolver

Overapproximation via Constraint-based Reasoning

Stefan Ratschan, Czech Academy of Sciences, Prague, Czech Rep.
Shikun She, Beihang University, Beijing, China

Starting Point: Interval Grid Method

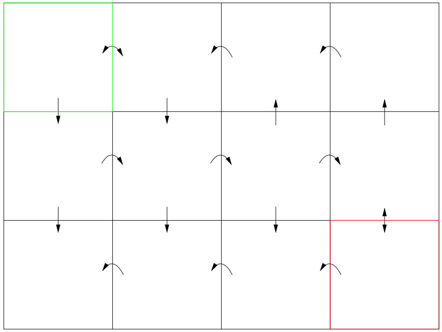
Stursberg/Kowalewski et. al., one-mode case:



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

Starting Point: Interval Grid Method

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

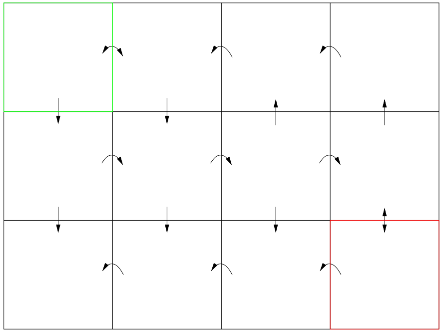


$\dot{x} \in [-5, -1]$

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

Starting Point: 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 arithmetic

Is a method for calculating an interval *covering* the possible values of a real operator if its arguments range over intervals:

$$
\begin{aligned}
[a, A] \overset{\circ}{+} [b, B] &= [a + b, A + B] \\
[a, A] \overset{\circ}{\cdot} [b, B] &= [\min\{ab, aB, Ab, AB\}, \max\{ab, aB, Ab, AB\}] \\
\overset{\circ}{\min}([a, A], [b, B]) &= [\min\{a, b\}, \min\{A, B\}] \\
\overset{\circ}{\sin}([a, A]) &= \begin{bmatrix} \min\{\sin x \mid x \in [a, A]\}, \\ \max\{\sin x \mid x \in [a, A]\} \end{bmatrix} \\
\overset{\circ}{f}([a, A], [b, B], \ldots) &= \begin{bmatrix} \min\{f(\mathbf{x}) \mid \mathbf{x} \in [a, A] \times [b, B] \times \ldots\}, \\ \max\{f(\mathbf{x}) \mid \mathbf{x} \in [a, A] \times [b, B] \times \ldots\} \end{bmatrix}
\end{aligned}
$$

Theorem: For each term t with free variables $\mathbf{v}$:
$\{t(\mathbf{v} \mapsto \mathbf{x}) \mid \mathbf{x} \in [a, A] \times [b, B] \times \ldots\} \subseteq \overset{\circ}{t}(v_1 \mapsto [a, A], v_2 \mapsto [b, B], \ldots)$

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

Properties

Advantages:

- can deal with constants that are only known up to intervals
- interval tests cheap (e.g., compare to explicit computation of continuous reach sets, or full decision procedures)

Disadvantages:

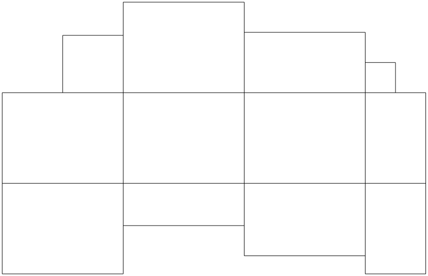
- may require a very fine grid to provide an affirmative answer (curse of dimensionality)
- ignores the continuous behavior within the grid elements

Let's remove them!

Removing Disadvantages

Objective: 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

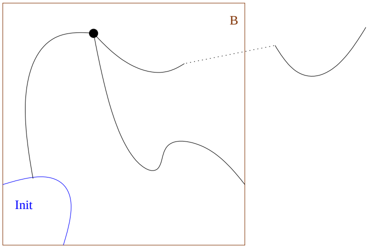


Method: formulate constraints that hold on reachable parts of state space, remove non-solutions by constraint solver.

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



$\Rightarrow$ formulate corresponding constraints, remove all points from box that do not fulfill at least one of these constraints.

Constraints in Specification

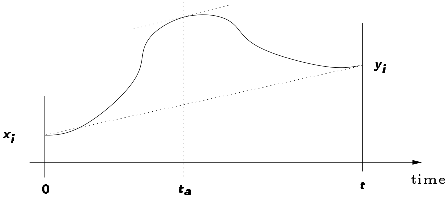
As before, we specify system using constraints involving ODEs:

- $\mathit{Flow}(s, \mathbf{x}, \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t})$
  - e.g., $s = \mathit{off} \rightarrow \frac{\mathrm{d}x}{\mathrm{d}t} = x \sin(x) + 1 \quad \ldots$
- $\mathit{Jump}(s, \mathbf{x}, s', \mathbf{x}')$
  - e.g., $(s = \mathit{off} \wedge x \geq 10) \;\rightarrow\; (s' = \mathit{on} \wedge x' = 0)$
- $\mathit{Init}(s, \mathbf{x})$
  - e.g., $s = \mathit{off} \wedge x = 0$

Lemma (n-dimensional mean value theorem): For a box B, mode s, if a point $(y_1, \ldots, y_n) \in B$ is reachable from a point $(x_1, \ldots, x_n) \in B$ via a flow in B then

$$\exists t \in \mathbb{R}_{\geq 0} \bigwedge_{1 \leq i \leq n} \exists a_1, \ldots, a_k, \overset{\bullet}{a}_1, \ldots, \overset{\bullet}{a}_k \; [(a_1, \ldots, a_k) \in B \wedge$$

$$Flow(s, (a_1, \ldots, a_k), (\overset{\bullet}{a}_1, \ldots, \overset{\bullet}{a}_k)) \wedge y_i = x_i + \overset{\bullet}{a}_i \cdot t]$$



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

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

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

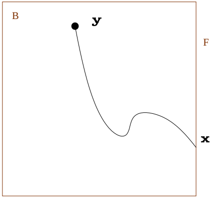
Lemma: For a box $B \subseteq \mathbb{R}^k$, mode s, if $\mathbf{y} \in B$ is reachable from a neighboring box over a face F of B and a flow in B then

$$\exists \mathbf{x} \in F \left[\mathit{incoming}_F(s, \mathbf{x}) \wedge \mathit{flow}_B(s, \mathbf{x}, \mathbf{y}) \right],$$

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

$$\exists \dot{x}_1, \ldots, \dot{x}_k \left[\mathit{Flow}(s, \mathbf{x}, (\dot{x}_1, \ldots, \dot{x}_k)) \wedge \dot{x}_j \;\; r \; 0 \right]$$

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

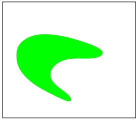


for corners etc. a little bit more involved

Using Constraints

These constraints can be used for removing definitely unreachable parts from boxes:

1. instantiate the constraints by substituting *Flow*, *Jump*, *Init* into their definition,
2. take each individual box,
3. apply interval constraint propagation wrt. the constraints to the box.



- safe overapproximation, incl. correct handling of rounding errors
- result not necessarily tight

[Ratschan & She, 2004–, `http://hsolver.sourceforge.net`]

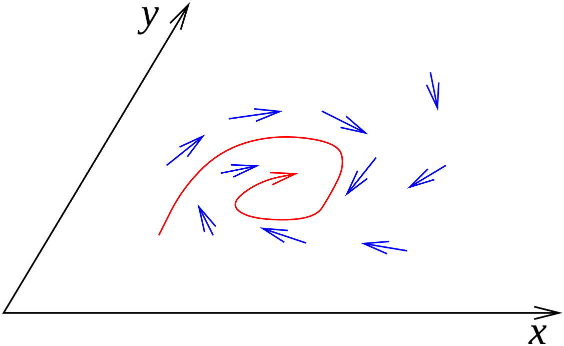
Automated Stability Proofs

Lyapunov-based Methods

Lyapunov's direct method for showing L. stability

- **Observation:** Stabilizing systems often amounts to diminishing energy in certain subsystems.
- **Idea:** Show stabilization by
  1. seeking an *appropriate "generalized energy function"*, and
  2. showing that it *decreases along the trajectories of the controlled system*.

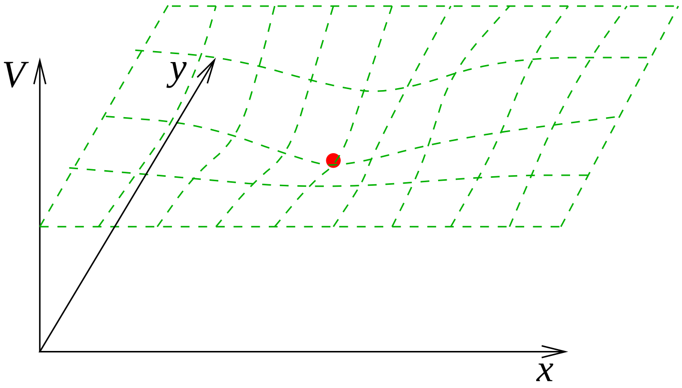
1. Model system dynamics as DE



$$\dot{\mathbf{x}} = f(\mathbf{x})$$

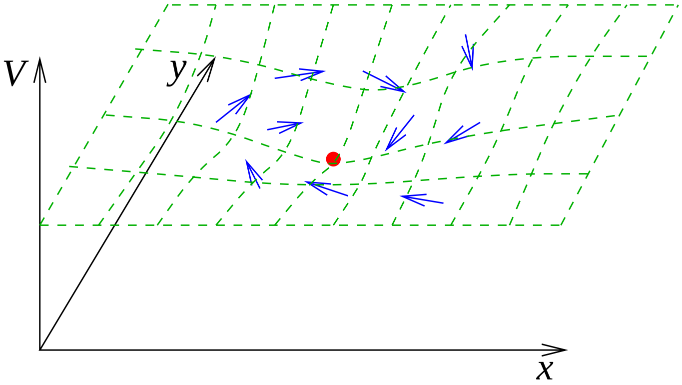
Lyapunov's direct method

2. Select witness function $V : \mathbb{R}^n \to \mathbb{R}$
   - V positive definite: $V(\mathbf{x}) \geq 0$ and $(V(\mathbf{x}) = 0 \iff \mathbf{x} = \mathbf{0})$
   - V continuously differentiable.



Lyapunov's direct method

3. Analyze growth of witness function along trajectories.
   - Non-increase of $\mathbf{x} \mapsto V(\mathbf{x} - \mathbf{x}_{eq})$ along trajectories satisfying $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = f(\mathbf{x})$ implies Lyapunov stability in $\mathbf{x}_{eq}$.



$$\left[\frac{\partial V}{\partial x_1}(\mathbf{x} - \mathbf{x}_{eq}), \ldots, \frac{\partial V}{\partial x_d}(\mathbf{x} - \mathbf{x}_{eq})\right] f(\mathbf{x}) \leq 0 \text{ for all } \mathbf{x}.$$

Automation: Idea

1. Take a parametric set of candidate Lyapunov functions
   - for example, polynomials of degree $2k$
2. Fit parameters such that Lyapunov's direct condition is satisfied

Methods for fitting functions

- Linear matrix inequalities & quadratic programming [Pettersson & Lennartson, 1996]
  - limited to polynomials of degree 2
  - problematic scalability (monolithic matrix inequality)
  - numerical stability issues
- Non-linear arithmetic constraint solving
  - uses the Lyapunov condition directly as a constraint on the parameters
  - solvable iff there exists an Lyapunov fct. in the class
  - solvability thus implies stability
  - linear ODE case: Rodriguez-Carbonell & Tiwari, 2002,
    general (incl. transcendental fct.s in ODE): Ratschan & She, 2006

How it works

- $f(\mathbf{x})$ right-hand side of ODE
- $V(\mathbf{p}, \mathbf{x})$ is a fct. of
  - parameters $\mathbf{p}$,
  - state variables $\mathbf{x}$;

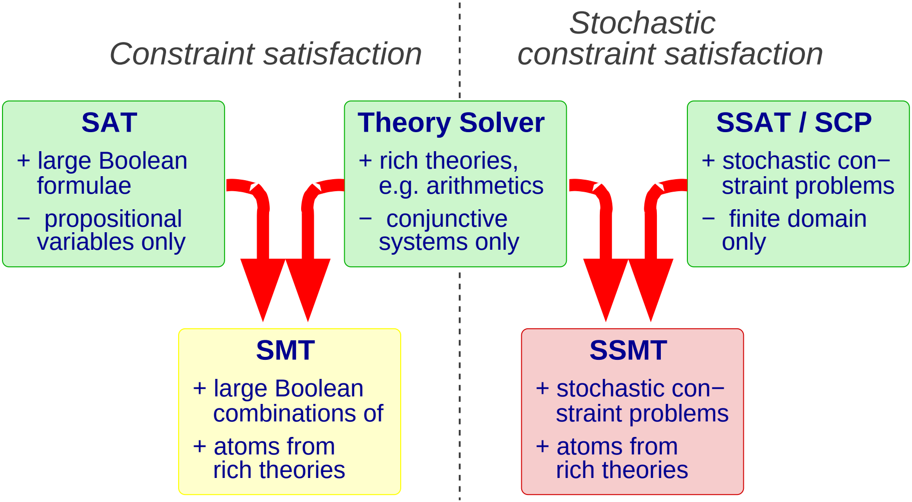
  $\frac{\partial V}{\partial x_i}(\mathbf{p}, \mathbf{x})$ its partial derivatives
- Decide whether

$$\exists \mathbf{p} \forall \mathbf{x} : \left[\frac{\partial V}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial V}{\partial x_d}(\mathbf{x}) \right] f(\mathbf{x}) \leq 0$$

  is true
- successfully pursued using the ICP-based constraint solver RSolver [Ratschan 2002-], cf. [Ratschan & She, 2006]

Extension to Probabilistic Hybrid Systems

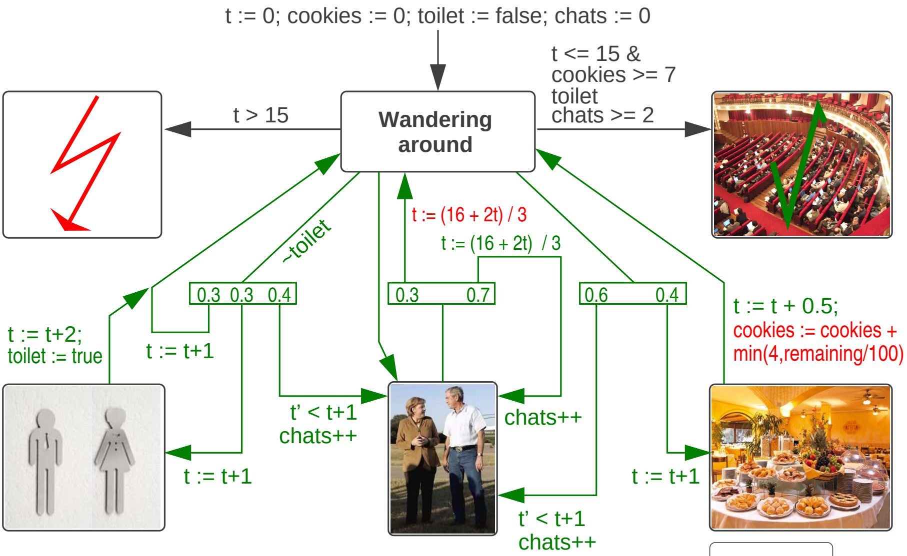
Quantifying the probability of misbehavior



BMC / stability proofs / ...
of hybrid systems

BMC / stability proofs / ...
of probabilistic hybrid systems

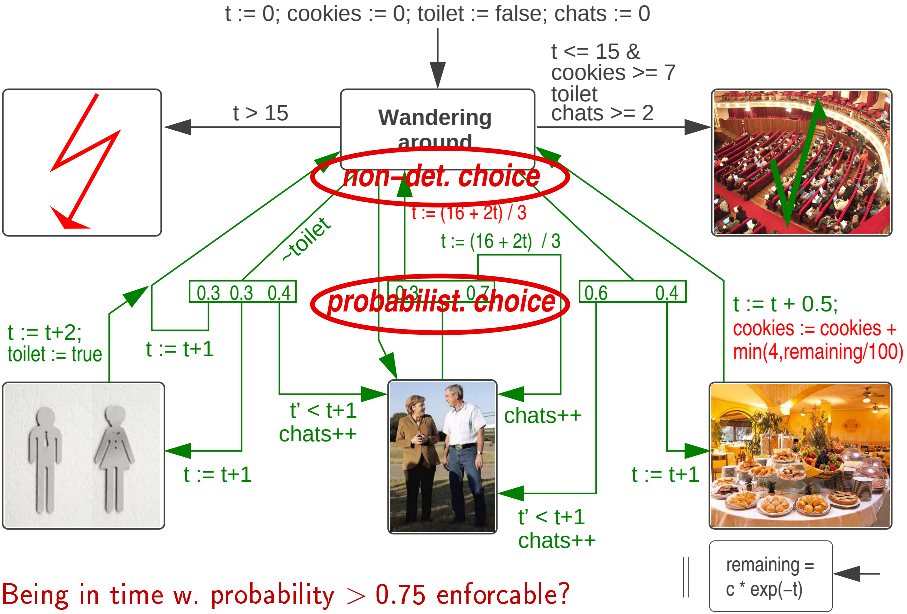
Example: The Summer School Pause Dilemma



Being in time w. probability > 0.75 enforcable?

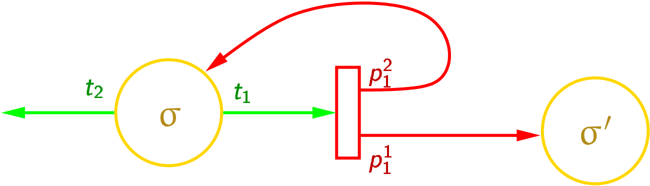
remaining = c * exp(−t)

Example: The Summer School Pause Dilemma



Being in time w. probability > 0.75 enforcable?

Worst-Case Probability of Reaching Target



Given

- a PHA A,
- a hybrid state $(\sigma, \mathbf{x})$,
- a set of target locations TL,

the **maximum probability** $\mathbf{P}^k_{(\sigma,\mathbf{x})}$ of reaching TL from $(\sigma, \mathbf{x})$ within $k \in \mathbb{N}$ steps is

$$\mathbf{P}^k_{(\sigma,\mathbf{x})} = \begin{cases} 1 & \text{if } \sigma \in TL, \\ 0 & \text{if } \sigma \notin TL \wedge k = 0, \\ \max_{i:(\sigma,\mathbf{x}) \models g(t_i)} \sum_j \left(p_i^j \cdot \mathbf{P}^{k-1}_{asgn_i^j(\sigma,\mathbf{x})} \right) & \text{if } \sigma \notin TL \wedge k > 0. \end{cases}$$

Probabilistic Bounded Reachability

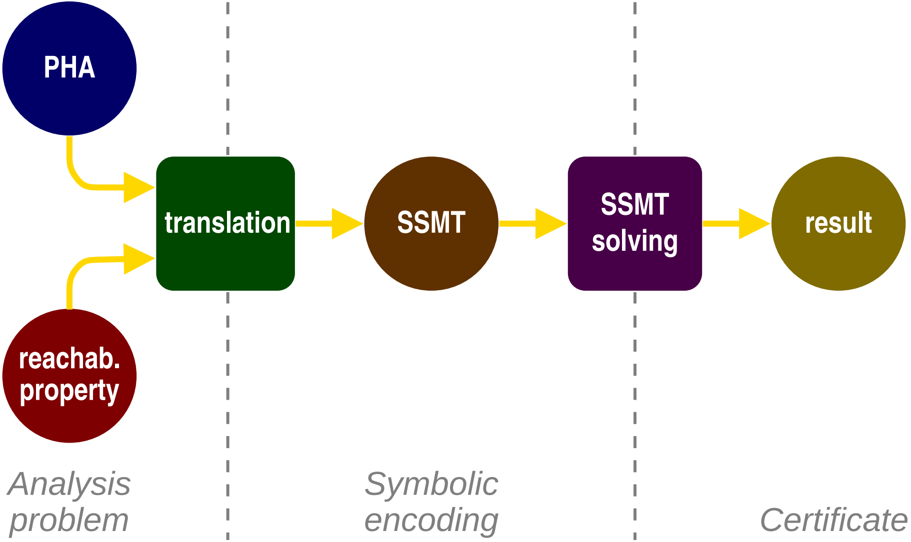
Given:

- a PHA A,
- a set of target locations TL,
- a depth bound $k \in \mathbb{N}$,
- a probability threshold $tolerable \in [0, 1]$.

Probabilistic Bounded Reachability Problem:

- Is $\max_{(\sigma,\mathbf{x})\text{ an initial state}} \mathbf{P}^k_{(\sigma,\mathbf{x})} \leq tolerable$?
- I.e., is accumulated probability *over all paths* of reaching bad state *under malicious adversary* within k steps acceptable?

Approach


PHA
reachab. property
translation
SSMT
SSMT solving
result
Analysis problem
Symbolic encoding
Certificate

Stochastic Satisfiability Modulo Theory (SSMT)

Stochastic satisfiability modulo theory (SSMT)

- Inspired by Stochastic CP and Stochastic SAT (SSAT), e.g. [Papadimitriou 85] [Tarim, Manandhar, Walsh 06] [Balafoutis, Stergiou 06] [Bordeaux, Samulowitz 07] [Littmann, Majercik 98, dto. + Pitassi 01]
- Extends it to infinite domains (for innermost existentially quantified variables).
- Extends SSAT to SSAT(T) akin to DPLL vs. DPLL(T).

An SSMT formula consists of

1. an **SMT formula** φ over some (arithmetic) theory T, e.g.

$$\varphi = (x > 0 \vee 2a \cdot \sin(4b) \geq 3) \wedge (y > 0 \vee 2a \cdot \sin(4b) < 1) \wedge \ldots$$

2. a **prefix** of **existentially** and of **randomly quantified** variables with finite domains, e.g.

$$\exists x \in \{0,1\} \; Я_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} \;\; Я \ldots \exists \ldots Я \ldots$$

Quantification in SSMT

Objective: Determine probability of satisfaction of ϕ under existential and randomized choices of quantified variables:

1) **existential** $\exists\, x \in \mathrm{dom}(x)$

   Probability corresponds to optimal choice within range $\mathrm{dom}(x)$.

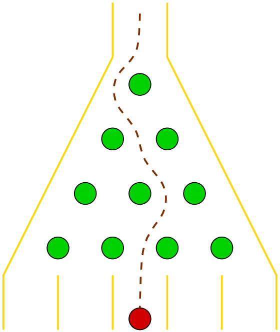
2) **randomized** $\text{Я}_{\langle (v_1, p_1), \ldots, (v_m, p_m) \rangle}\; y \in \mathrm{dom}(y)$

   Probability corresponds to random choice within range $\mathrm{dom}(y)$.

   p_i is probability of setting y to value v_i.

Randomized Quantification

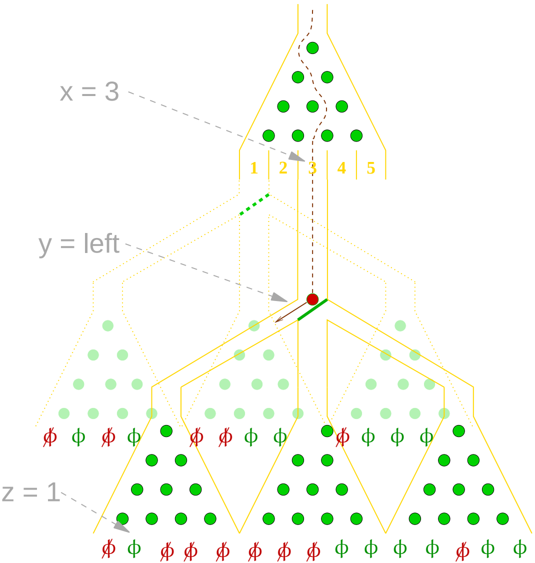
Galton Board: At each nail, ball bounces *left* or *right* with some probability p or $1 - p$, resp. (e.g. $p = 0.5$)



$k =$	**0**	**1**	**2**	**3**	**4**
$p_k =$	$\frac{1}{16}$	$\frac{4}{16}$	$\frac{6}{16}$	$\frac{4}{16}$	$\frac{1}{16}$

$$\reflectbox{R}_{\langle(0,p_0),(1,p_1),(2,p_2),(3,p_3),(4,p_4)\rangle} prob_1 \in \{0, 1, 2, 3, 4\}$$

Stochastic satisfiability modulo theory (SSMT)



$\text{Я}_{d_1} x \in \{0, 1, 2, 3, 4\}$

$\exists y \in \{left, middle, right\}$

$\text{Я}_{d_2} z \in \{0, 1, 2, 3, 4\}:$

Semantics of an SSMT formula

$$\Phi = Q_1 x_1 \in \mathrm{dom}(x_1) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi$$

Probability of satisfaction $Pr(\Phi)$:

Quantifier-free base cases:

1. $Pr(\varepsilon : \varphi) = \mathbf{0}$ if φ is **unsatisfiable.**
2. $Pr(\varepsilon : \varphi) = \mathbf{1}$ if φ is **satisfiable.**

$\exists \triangleq$ Maximum over all alternatives:

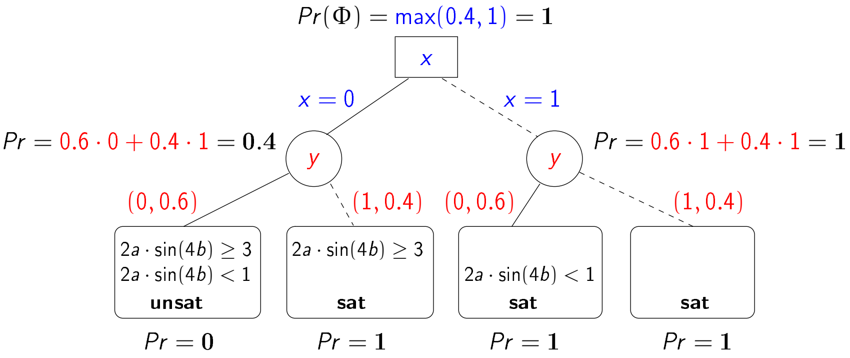
3. $Pr(\exists x \in \mathcal{D}\ \mathcal{Q} : \varphi) = \max_{v \in \mathcal{D}} Pr(\mathcal{Q} : \varphi[v/x])$.

Ⅎ $\triangleq$ Weighted sum of all alternatives:

4. $Pr(Ⅎ_d x \in \mathcal{D}\ \mathcal{Q} : \varphi) = \sum_{(v,p) \in d} p \cdot Pr(\mathcal{Q} : \varphi[v/x])$.

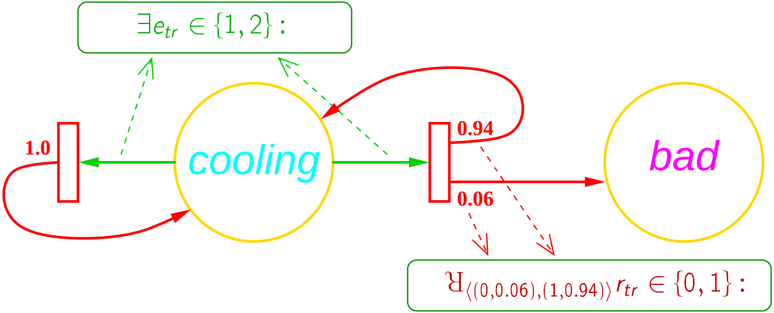
Semantics of an SSMT formula: Example

$$\Phi = \exists x \in \{0,1\}\ \text{Я}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \lor 2a \cdot \sin(4b) \geq 3) \land (y > 0 \lor 2a \cdot \sin(4b) < 1)$$



Translating PHA Problems to SSMT Problems

Translating PHA into SSMT



source $\wedge$	guard $\wedge$	trans $\wedge$	distr $\wedge$	action $\wedge$	target
$\Big($ *cooling* $\wedge$	$(T \geq 90^\circ)\ \wedge$	$(e_{tr} = 1)\wedge$	true $\wedge$	$(T' = T - \Delta t \cdot f_{cool}) \wedge (t' = t + \Delta t)$	$\wedge\, cooling' \Big) \vee$
$\Big($ *cooling* $\wedge$	$(T > 110^\circ)\wedge$	$(e_{tr} = 2)\wedge$	$(r_{tr} = 0)\wedge$	$(t' = t + \Delta t)$	$\wedge\ bad' \Big) \vee$
$\Big($ *cooling* $\wedge$	$(T > 110^\circ)\wedge$	$(e_{tr} = 2)\wedge$	$(r_{tr} = 1)\wedge$	$(T' = T - \Delta t \cdot f_{cool}) \wedge (t' = t + \Delta t)$	$\wedge\ cooling' \Big)$

Unwinding

$$\underbrace{\exists t_1 \text{Ⅎ}_d p_1 \exists t_2 \text{Ⅎ}_d p_2 \ldots \exists t_k \text{Ⅎ}_d p_k}_{\text{alternating choices}} : \underbrace{\underbrace{\begin{pmatrix} \mathit{Init}(\mathbf{x}_0) \\ \wedge\ \mathit{Trans}(\mathbf{x}_0, \mathbf{x}_1) \\ \wedge\ \mathit{Trans}(\mathbf{x}_1, \mathbf{x}_2) \\ \wedge \ldots \\ \wedge\ \mathit{Trans}(\mathbf{x}_{k-1}, \mathbf{x}_k) \end{pmatrix}}_{k\text{-bounded reach set}} \wedge \underbrace{\begin{pmatrix} \mathit{Bad}(\mathbf{x}_0) \\ \vee\ \mathit{Bad}(\mathbf{x}_1) \\ \vee\ \mathit{Bad}(\mathbf{x}_2) \\ \vee \ldots \\ \vee\ \mathit{Bad}(\mathbf{x}_k) \end{pmatrix}}_{\text{hits bad state}}}_{\text{BMC}(k)}$$

- Alternating quantifier prefix encodes alternation of
  - nondeterministic transition selection
  - probabilistic choice between transition variants
- $Pr(\Phi)$ = accumulated probability over all paths of reaching bad state under malicious adversary within k steps $= \max_{(\sigma,\mathbf{x}) \text{ initial}} \mathbf{P}^k_{(\sigma,\mathbf{x})}$.

$$\max_{(\sigma,\mathbf{x}) \text{ initial}} \mathbf{P}^k_{(\sigma,\mathbf{x})} > \mathit{tolerable} \text{ iff } Pr(\Phi) > \mathit{tolerable}$$

SSMT Solving

SSMT algorithm

Problem: Determine whether $Pr(\Phi) > tolerable$, where

- $\Phi = Pre : \varphi$ is an SSMT formula
- φ is a Boolean combination of (non-linear) arithmetic constraints
- $Pr(\Phi)$ the satisfaction probability of Φ
- *tolerable* is a constant, the probabilistic satisfaction threshold.

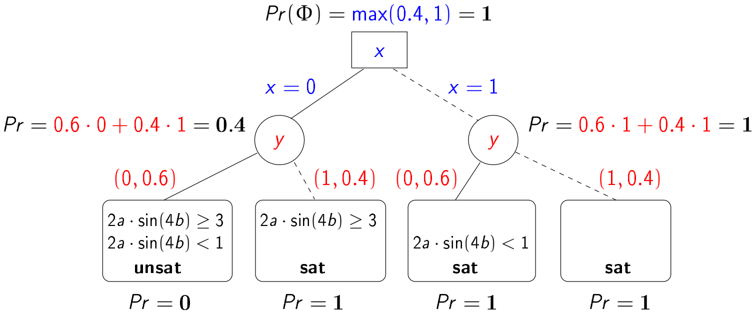
Solution: Take appropriate SMT solver, implant branching rules for quantifiers, add rigorous proof-tree pruning:

- **iSAT** solver for mixed Boolean and non-linear arithmetic problems [Fränzle, Herde, Ratschan, Schubert, Teige 2006+2007]
- **iSAT** + **branching rules** for quantifier handling + **pruning rules** $\rightsquigarrow$ **SiSAT** [Teige and Fränzle, CPAIOR 2008]

Naive SSMT solving

1. Enumerate assignments to quantified variables
2. Call subordinate SMT solver on resulting instances
3. Aggregate results accord. to SSMT semantics, compare to *tolerable*

$$\Phi = \exists x \in \{0,1\}\ \text{Я}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \lor 2a \cdot \sin(4b) \geq 3) \land (y > 0 \lor 2a \cdot \sin(4b) < 1)$$



Efficient quantifier handling

Given:

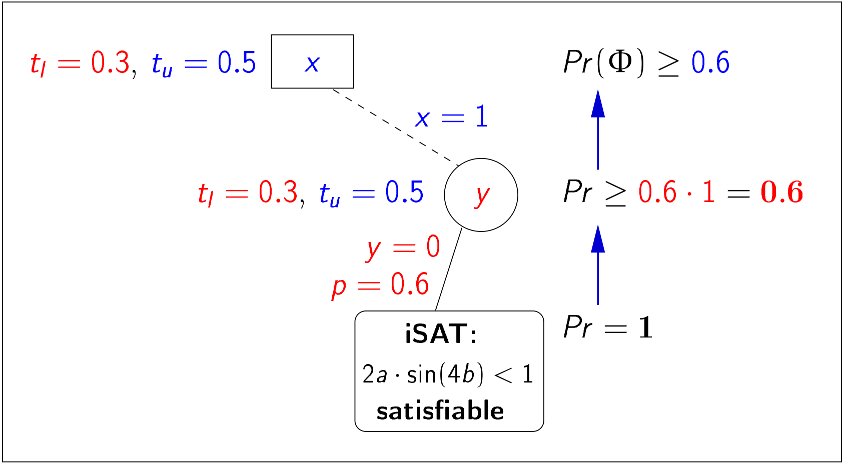
- $$\Phi = \exists x \in \{0,1\}\ \text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
  $$(x > 0 \vee 2a \cdot \sin(4b) \geq 3) \wedge (y > 0 \vee 2a \cdot \sin(4b) < 1),$$
- **lower threshold** $t_l = 0.3$,
- **upper threshold** $t_u = 0.5$.

Objective:

- $Pr(\Phi) \overset{?}{<} t_l$ or $Pr(\Phi) \overset{?}{>} t_u$ or compute $t_l \leq Pr(\Phi) \leq t_u$?

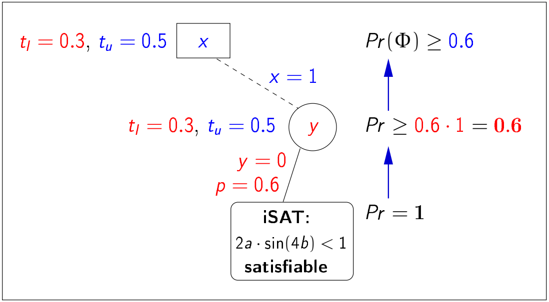
Efficient quantifier handling

$$\Phi = \exists x \in \{0,1\}\ Я_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \lor 2a \cdot \sin(4b) \geq 3) \land (y > 0 \lor 2a \cdot \sin(4b) < 1)$$



Efficient quantifier handling

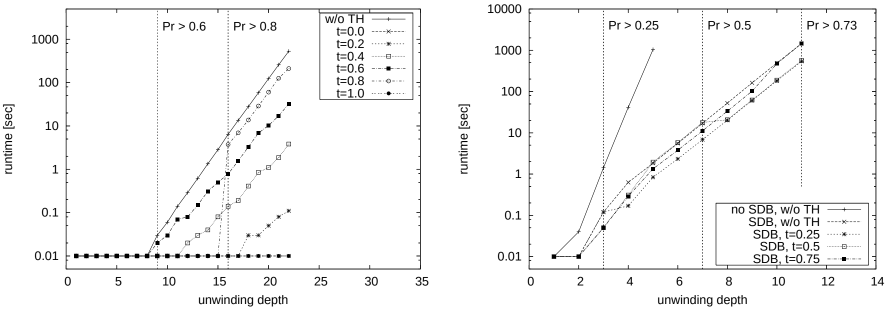
$$\Phi = \exists x \in \{0,1\}\ \text{Ʀ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \vee 2a \cdot \sin(4b) \geq 3) \wedge (y > 0 \vee 2a \cdot \sin(4b) < 1)$$



Pruning occurs

- when satisfaction probability of investigated branches $> t_u$,
- when probability mass of remaining branches $< t_l$,
- *based on inferences in SMT solving*

First experimental results



Impact of thresholding (left) and solution-directed backjumping (right)

SSMT often traverses only minor fraction of quantifier domains!

Synopsis

- Hybrid systems
  - are a reasonable formalization of the interaction of embedded control and physical environment
  - there is rapidly growing body of theory pertaining to hybrid systems
  - the theory bridges various fields of science, among them
    - control theory
    - discrete event systems
    - numerical analysis
    - arithmetic constraint solving
- Arithmetic constraint solving
  - is an enabler for fully symbolic analysis of hybrid systems
  - thus providing prospects for scalable automatic analysis procedures;
  - its solving power is progressing much more rapidly than the advances in computing hardware
  - yet still in its infancy.

Thanks

- to the many collaborators within four major projects:
  - DFG-funded Transregional Research Center 14 "AVACS" (Automatic Analysis and Verification of Complex Systems)
  - DFG-funded Graduate School 1076 "TrustSoft" (Trustworthy Software Systems)
  - Project "IMoST" (Integrated Modelling for Safe Transportation) funded by the state of Lower Saxony
  - Helmholtz Virtual Institute "DeSCAS" (Design of Safety-Critical Automotive Systems)
- and to the contributing institutions:
  - Academy of Sciences of the Czech Republic, Prague, Czech Republic
  - Albert-Ludwigs-Universität Freiburg, Germany
  - Carl von Ossietzky Universität Oldenburg, Germany
  - DLR Braunschweig, Germany
  - ETH Zurich, Switzerland
  - MPII, Saarbrücken, Germany
  - TU Braunschweig, Germany
  - Universität des Saarlandes, Saarbrücken, Germany
- and to Andreas Eggers, Christian Herde, Stefan Ratschan, and Tino Teige for contributing many slides.