Bounded Model Checking of Hybrid Systems

From Qualitative to Quantitative Certificates and from Falsification to Verification

Martin Fränzle[1]

joint work with A. Eggers, C. Herde, T. Teige (all Oldenburg), N. Kalinnik, S. Kupferschmid, T. Schubert, B. Becker (Freiburg), H. Hermanns (Saarbrücken), S. Ratschan (Prague)

SFB/TR 14 AVACS

[1] Dpt. of Computing Science · C. v. Ossietzky Universität · Oldenburg, Germany

What is a hybrid system?

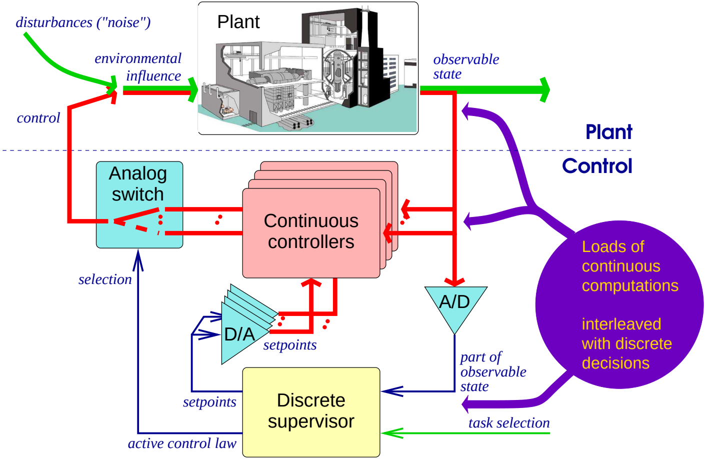
Hybrid (from Greece) means arrogant, presumptuous.

After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

Hybrid stems from Latin hybrida 'offspring of a tame sow and wild boar, child of a freeman and slave, etc.'

From the Compact Oxford English Dictionary, 2008

Hybrid Systems



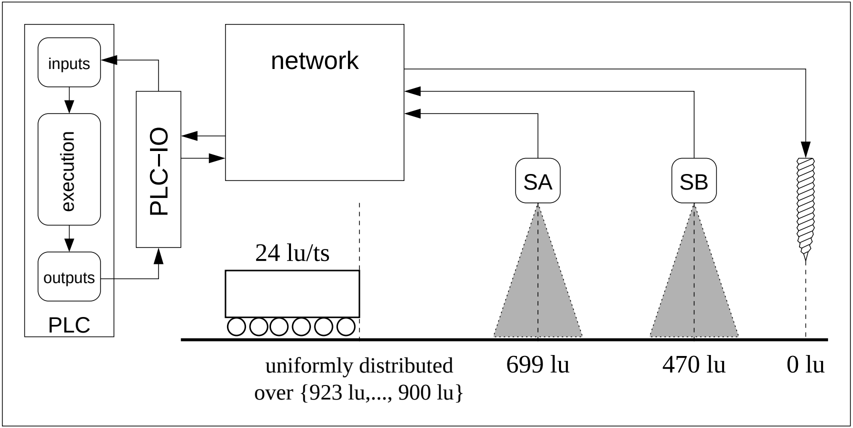
Which one is the tame sow and which the wild boar?

Hybrid systems

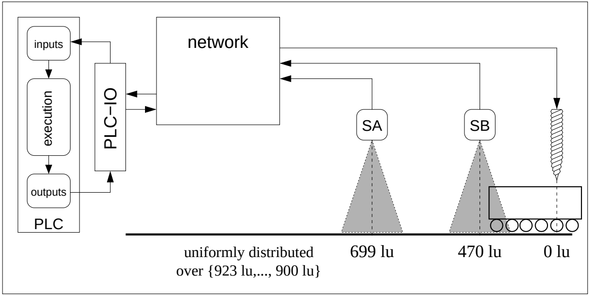
are ensembles of interacting discrete and continuous subsystems:

- **Technical systems:**
  - physical plant + multi-modal control
  - physical plant + embedded digital system
  - mixed-signal circuits
  - multi-objective scheduling problems (computers / distrib. energy management / traffic management / ...)
- **Biological systems:**
  - Delta-Notch signaling in cell differentiation
  - Blood clotting
  - ...
- **Economy:**
  - cash/good flows + decisions
  - ...
- **Medicine/health/epidemiology:**
  - infectious diseases + vaccination strategies
  - ...

A Networked Automation System
(After Greifeneder and Frey, 2006)



A Networked Automation System



Questions:

- **May** the carriage **ever** stop outside the designated range of drilling positions, or even fail to stop at all?
- **How likely** is it to stop inside the designated range of drilling positions?
- What is the **expected value** of the stopping position, etc.?

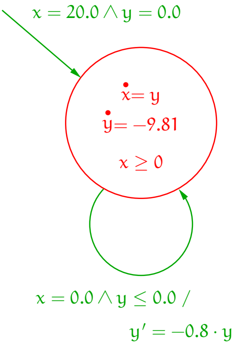
Agenda

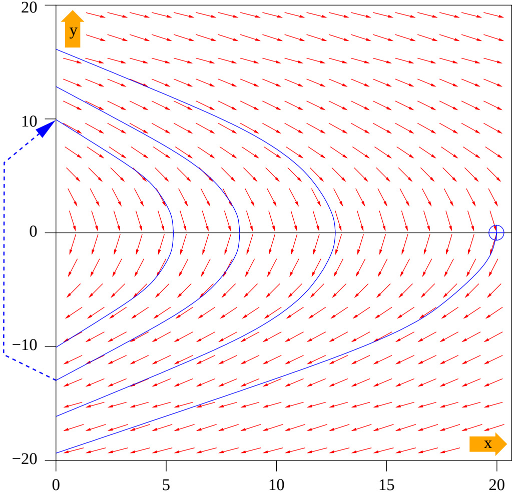
1. **Qualitative analysis:**
   1. An appropriate computational model: hybrid automata
   2. Bounded model checking of discrete-time HA:
      - reduction to arithmetic constraint formulae,
      - arithmetic constraint solving.
   3. Bounded model checking of dense-time HA:
      - constraint solving for arithmetic formulae involving ODE.
2. **Quantitative analysis:**
   1. An appropriate computational model: probabilistic hybrid automata
   2. Bounded model checking of avoid probabilities
      - falsification by reduction to quantified arithmetic constraint formulae,
      - constraint solving involving randomized quantifiers.
   3. Bounded model checking of expected avoid times
      - verification by reduction to quantified arithmetic constraint formulae.

Bounded Model Checking of Hybrid Systems

The Qualitative Case

A Formal Model: Hybrid Automata





x : vertical position of the ball

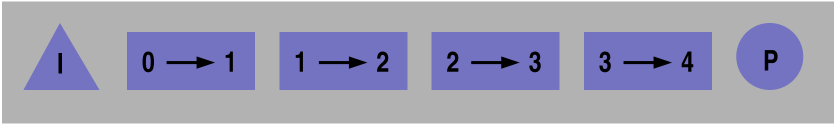
y : velocity

- $y > 0$ ball is moving up
- $y < 0$ ball is moving down

SAT Modulo Theory

An engine for bounded model checking of linear hybrid automata

Bounded Model Checking (BMC)

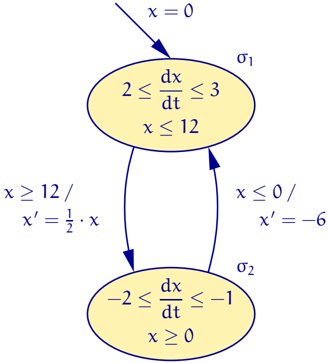


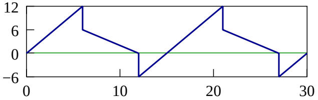
- construct formula that is satisfiable iff error trace of length k exists
- formula is a k–fold unwinding of the system's transition relation, concatenated with a characterization of the initial state(s) and the (unsafe) state to be reached

$$\neg \left(\begin{array}{ll} & init(x_0) \wedge trans(x_0, x_1) \wedge \ldots \wedge trans(x_{i-1}, x_i) \\ \Rightarrow & \phi(x_0) \wedge \ldots \wedge \phi(x_i) \end{array} \right)$$

- 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

BMC of Linear Hybrid Automata





Initial state:

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

Jumps:

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

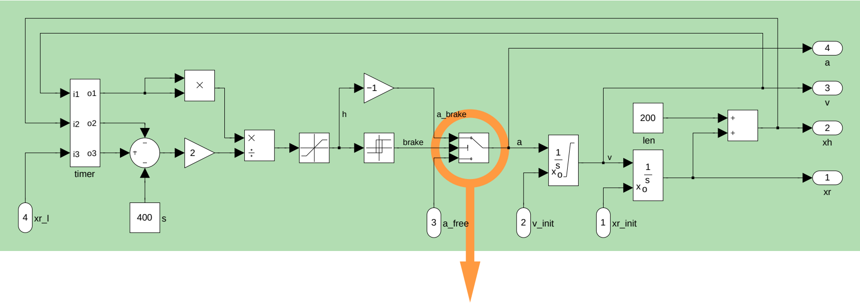
Flows:

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

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

Parallel composition corresponds to conjunction of formulae
⟶ No need to build product automaton

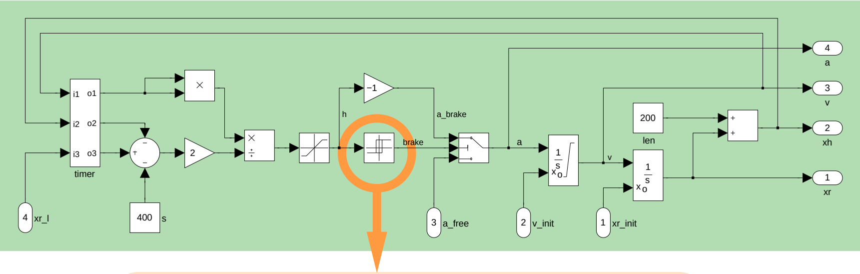
Translation to HySAT




```
- 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;
```

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);
```

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:

- ☺ efficient handling of CNFs and thus (by definitional translation) arbitrarily structured Boolean formulae
- ☹ propositional variables only

Linear Programming Solver:

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

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

(Simplified) SAT Modulo Theory Scheme: LinSAT

Davis Putnam

$2e + C + D \geq 2$

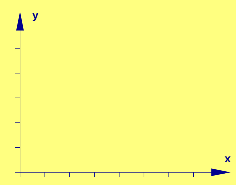
$2f + A + B \geq 2$

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

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

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

Linear Programming



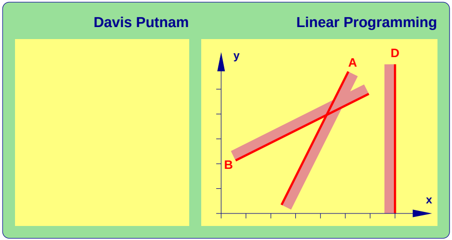
Input formula:

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

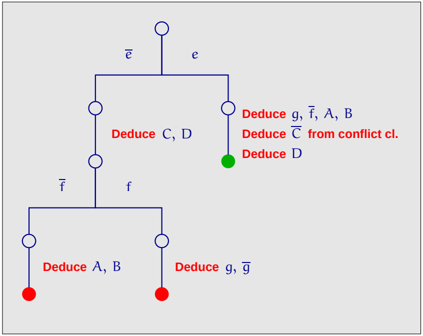
DPLL search

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



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.

SAT modulo theory for LinSAT

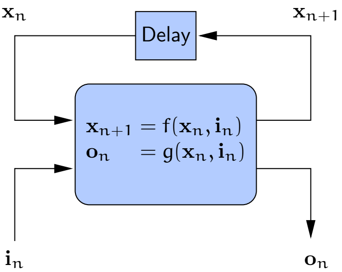
- SAT modulo theory solvers reasoning over linear arithmetic as a theory are readily available: E.g.,
  - LPSAT [Wolfman & Weld, 1999]
  - ICS [Filliatre, Owre, Rueß, Shankar 2001], Simplics [de Moura, Dutertre 2005], Yices [Dutertre, de Moura 2006]
  - MathSAT [Audemard, Bertoli, Cimatti, Kornilowicz, Sebastiani, Bozzano, Juntilla, van Rossum, Schulz 2002–]
  - CVC [Stump, Barrett, Dill 2002], CVC Lite [Barrett, Berezin 2004], CVC3 [Barrett, Fuchs, Ge, Hagen, Jovanovic 2006]
  - HySAT I [Herde & Fränzle, 2004]
  - Z3 [Bjørner, de Moura, 2006-]
  - ...
- 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.

SAT + Interval Constraint Propagation

An engine for BMC of
non-linear discrete-time HA

Bounded Model Checking of Nonlinear Discrete-Time Hybrid Systems (1)

Given:



Nonlinear 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 nonlinear.

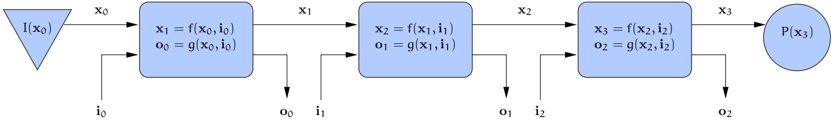
Goal:

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

Bounded Model Checking of Nonlinear Discrete-Time 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

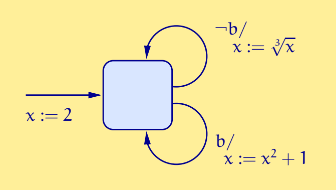


- Use appropriate procedure to "decide" satisfiability of the formula

Needed:

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

Bounded Model Checking with HySAT / iSAT



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]
```

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

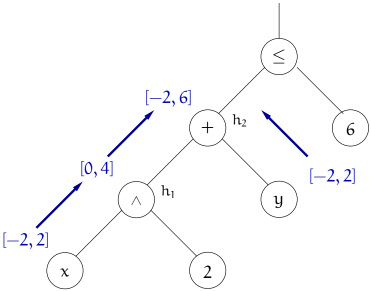
- do either address much smaller fragments of arithmetic
  - decidable theories: no transcendental fct.s, no ODEs
- or tackle only small formulae
  - some dozens of Boolean connectives.

Interval Constraint Propagation (1)

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

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

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



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

$\Downarrow$

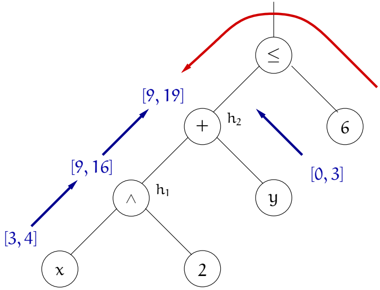
$h_2 \leq 6$ is satisfied in box

Interval Constraint Propagation (1)

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

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

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



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

$\Downarrow$

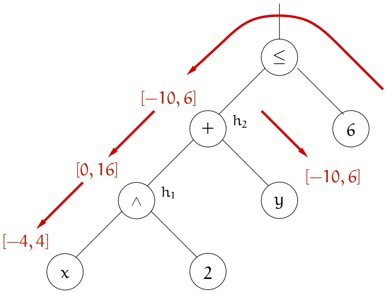
$h_2 \leq 6$ is unsat. in box

Interval Constraint Propagation (1)

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

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

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



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

$$\Downarrow$$

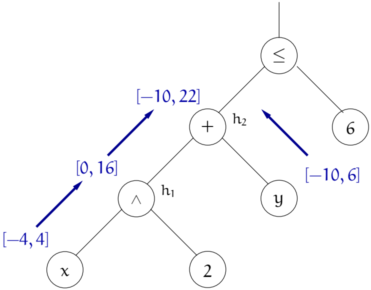
$$x \in [-4, 4] \wedge y \in [-10, 6]$$

Interval Constraint Propagation (1)

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

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

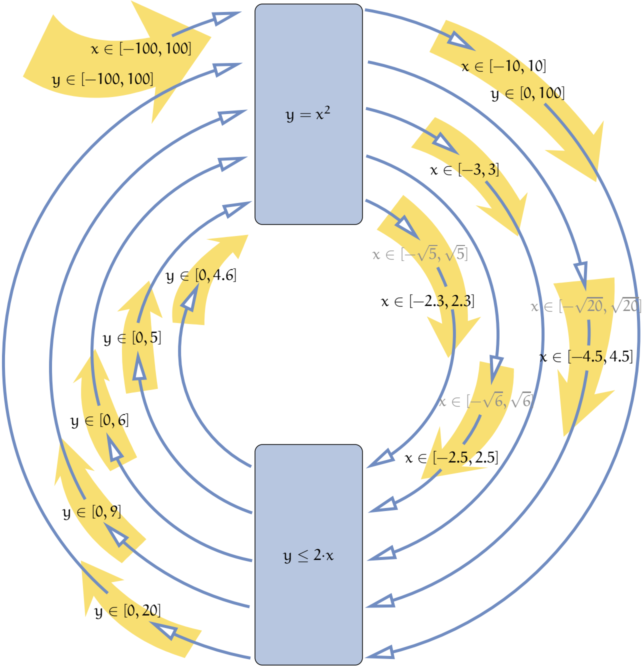
- Interval prop. (fwd & bwd until fixpoint is reached) yields contraction of box:



Constraint is not satisfied by the contracted box!

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

(details & alternatives: see Benhamou in Handbook of Constraint Progr.)


x ∈ [−100, 100]
y ∈ [−100, 100]
y = x²
x ∈ [−10, 10]
y ∈ [0, 100]
x ∈ [−3, 3]
x ∈ [−√5, √5]
y ∈ [0, 4.6]
x ∈ [−2.3, 2.3]
x ∈ [−√20, √20]
y ∈ [0, 5]
x ∈ [−4.5, 4.5]
x ∈ [−√6, √6]
y ∈ [0, 6]
x ∈ [−2.5, 2.5]
y ∈ [0, 9]
y ≤ 2·x
y ∈ [0, 20]

Interval contraction

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

Thus, interval contraction provides a highly incomplete deduction system:

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

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

iSAT: Non-linear Arithmetic Constraint Solving

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

$c_2:\quad \wedge\ (\neg a \vee \neg b \vee c)$

$c_3:\quad \wedge\ (\neg c \vee \neg d)$

$c_4:\quad \wedge\ (b \vee x \geq -2)$

$c_5:\quad \wedge\ (x \geq 4 \vee y \leq 0 \vee h_3 \geq 6.2)$

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

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

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

- 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 symb

- Boolean variables are regarded as 0-1 integer variables. Allows identification of literals with bounds on Booleans:

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

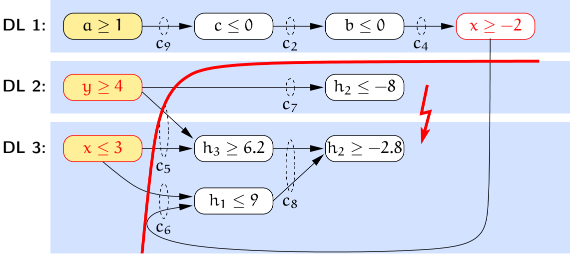
- Float variables h_1, h_2, h_3 are used for decomposition of complex constraint $x^2 - 2y \geq 6.2$.

iSAT: Non-linear Arithmetic Constraint Solving

$c_1: \quad (\neg a \lor \neg c \lor d)$
$c_2: \land (\neg a \lor \neg b \lor c)$
$c_3: \land (\neg c \lor \neg d)$
$c_4: \land (b \lor x \geq -2)$
$c_5: \land (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

$c_6: \land h_1 = x^2$
$c_7: \land h_2 = -2 \cdot y$
$c_8: \land h_3 = h_1 + h_2$

$c_9: \land (\neg a \lor \neg c)$
$c_{10}: \land (x < -2 \lor y < 3 \lor x > 3)$



← conflict clause = symbolic description

of a rectangular region of the search space

which is excluded from future search

iSAT: Non-linear Arithmetic Constraint Solving

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

$c_2: \wedge (\neg a \vee \neg b \vee c)$

$c_3: \wedge (\neg c \vee \neg d)$

$c_4: \wedge (b \vee x \geq -2)$

$c_5: \wedge (x \geq 4 \vee y \leq 0 \vee h_3 \geq 6.2)$

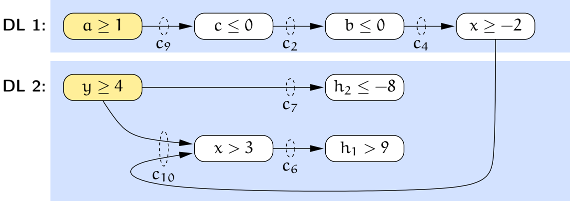
$c_6: \wedge h_1 = x^2$

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

$c_8: \wedge h_3 = h_1 + h_2$

$c_9: \wedge (\neg a \vee \neg c)$

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

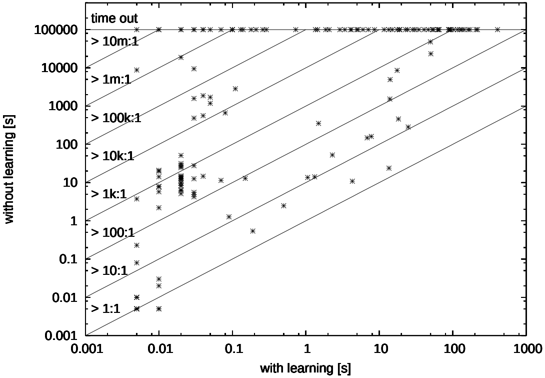


- Continue do split and deduce until either
  - ▷ formula turns out to be UNSAT (unresolvable conflict)
  - ▷ solver is left with 'sufficiently small' portion of the search space for which it cannot derive any contradiction

Results can be verified by sorting to "single assignment form".

Essentially, a tight integration of interval constraint propagation with recent propositional SAT-solving techniques.
[Fränzle, Herde, Ratschan, Schubert, Teige: J. on Satisfiability..., 2007]

The Impact of Learning: Runtime



Examples:
BMC of

- platoon control
- bouncing ball
- gingerbread map
- oscillatory logistic map

Intersection of geometric bodies

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

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

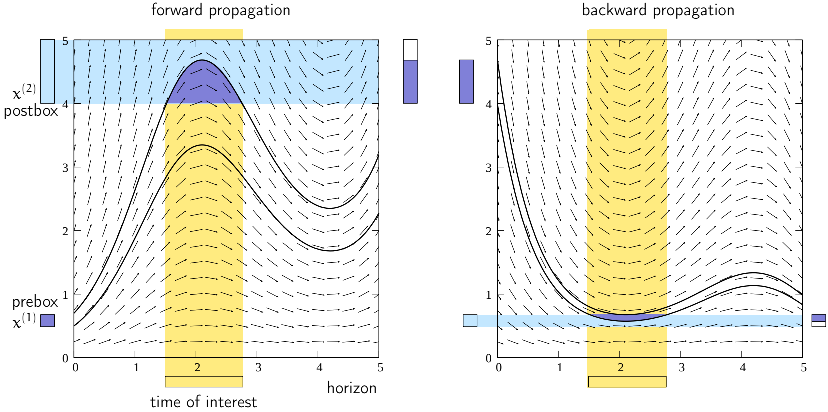
SAT + ICP + Numeric ODE Enclosure

An engine for BMC of non-linear continuous-time HA

1. Continuous flows, described by ODEs, define pre-post-constraints on continuous states:
   - Given an ODE $\frac{dx}{dt} = f(x)$ and a (convex) invariant $I \subset \mathrm{dom}(x)$,
   - $[\![\frac{dx}{dt}]\!] = \{(f(0), f(t)) \mid$ f solution of $\frac{dx}{dt} = f(x), \forall t' \leq t : f(t') \in I\}$
2. Adding direct support for such "ODE constraints" in arithmetic constraint solving facilitates BMC of continuous-time hybrid systems

   [Eggers & Fränzle: ATVA'08; Ishii, Ueda, Hosobe, Goldsztejn: ADHS'09]

odeSAT: Adding Forward and Backward Propagation for ODE Constraints

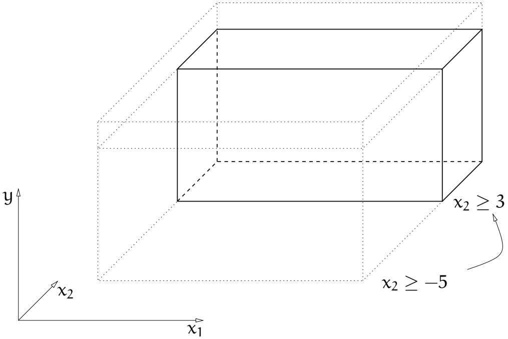


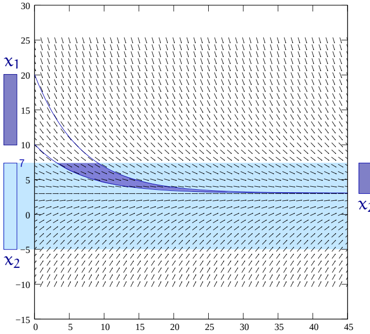
...yields a classical interval propagator!

iSAT+ODE: Integrated Algorithm (Example)

$$(x_1 + x_2 > y) \wedge (y \geq 28 \vee a) \wedge (\neg a \vee \frac{dx}{dt} = \frac{3}{20} \cdot (3 - x))$$

$$a \in \{\quad 1\}, x_1 \in [10, 20], x_2 \in [3, 7], y \in [0, 27]$$

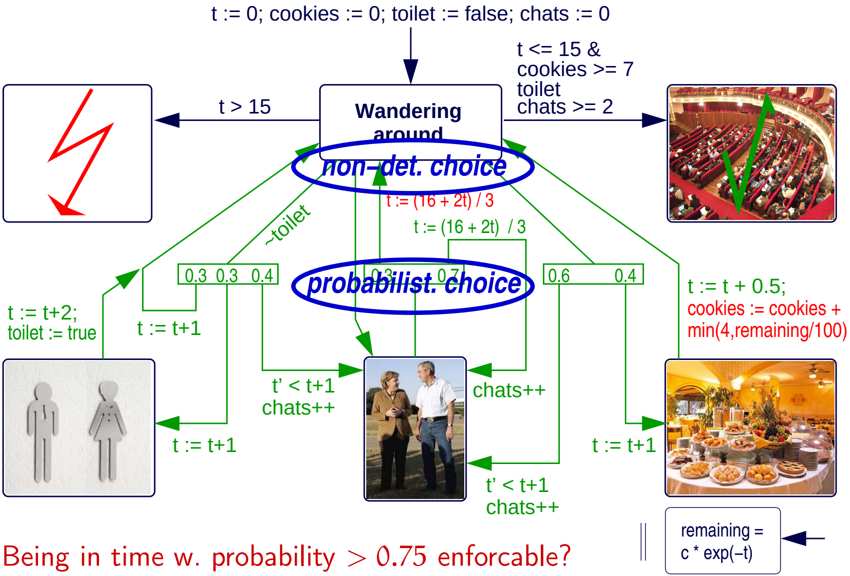




Bounded Model Checking of Hybrid Systems

The Quantitative Case

Example: The MoVeP Coffee-Break Dilemma

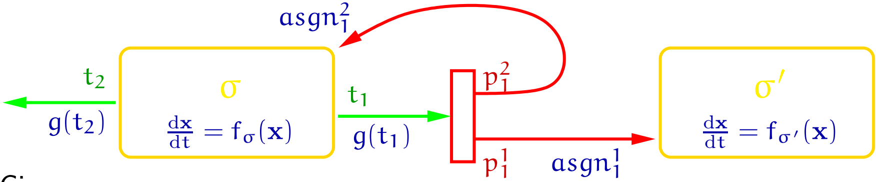


Being in time w. probability > 0.75 enforcable?

Quantitative Analysis 1

Probabilistic Bounded Reachability in Probabilistic Hybrid Automata

Worst-Case Probability of Reaching a Target Loc.



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

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

where F is the solution to the IVP $\frac{d\mathbf{y}}{dt} = f_\sigma(\mathbf{y})$, $\mathbf{y}_0 = \mathbf{x}$.

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?

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, which may include ODE, e.g.

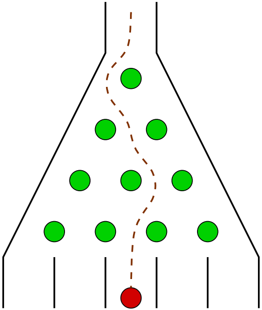
$$\varphi = (x > 0 \vee 2a \cdot \sin(4b) \geq 3) \wedge (y > 0 \vee 2a \cdot \sin(4b) < 1) \wedge \ldots$$

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

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

Randomized Quantification

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

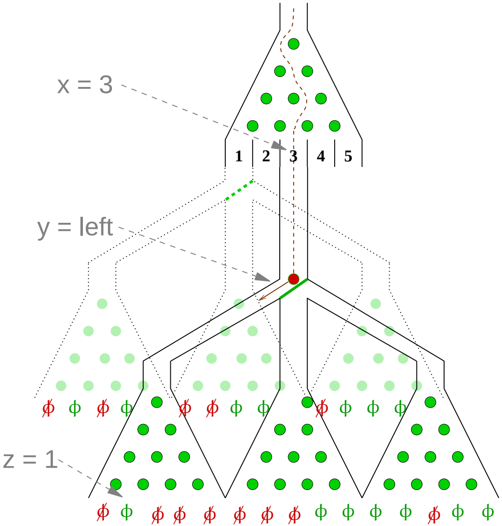


$$k = \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{2} \quad \mathbf{3} \quad \mathbf{4}$$

$$p_k = \quad \frac{1}{16} \quad \frac{4}{16} \quad \frac{6}{16} \quad \frac{4}{16} \quad \frac{1}{16}$$

$$\reflectbox{R}_{\langle(0,p_0),(1,p_1),(2,p_2),(3,p_3),(4,p_4)\rangle} \mathrm{prob}_1 \in \{0, 1, 2, 3, 4\}$$

Stochastic satisfiability modulo theory (SSMT)



$Я_{d_1} x \in \{1, 2, 3, 4, 5\}$

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

$Я_{d_2} z \in \{0, 1, 2, 3, 4\}:$

ϕ

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:

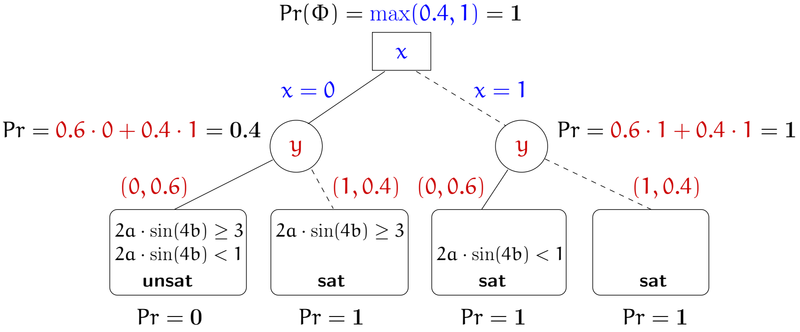
3. $\Pr(\exists x \in \mathcal{D}\ \mathcal{Q} : \varphi) = \max_{v \in \mathcal{D}} \Pr(\mathcal{Q} : \varphi[v/x])$.

Ɐ $\triangleq$ Weighted sum of all alternatives:

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

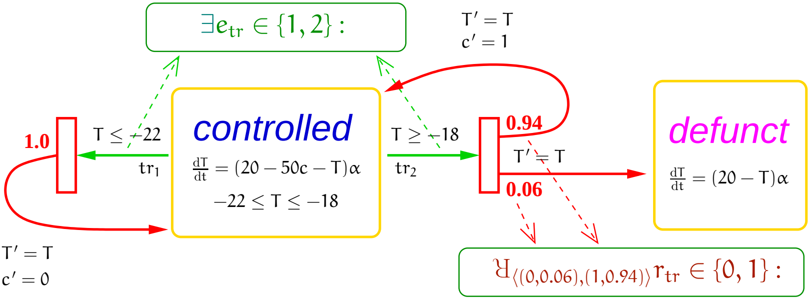
Semantics of an SSMT formula: Example

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



Translating PHA Problems to SSMT Problems

Translating *continuous-time* PHA into SSMT



source $\wedge$	guard $\wedge$	trans $\wedge$	distr $\wedge$	action $\wedge$	target	
(controlled $\wedge$	$(T \leq -22) \wedge$	$(e_{tr} = 1)$ $\wedge$	true $\wedge$	$(T' = T \wedge c' = 0) \wedge$	controlled$'$)	$\vee$
(controlled $\wedge$	$(T \geq -18) \wedge$	$(e_{tr} = 2)$ $\wedge$	$(r_{tr} = 0) \wedge$	$(T' = T)$ $\wedge$	defunct$'$)	$\vee$
(controlled $\wedge$	$(T \geq -18) \wedge$	$(e_{tr} = 2)$ $\wedge$	$(r_{tr} = 1) \wedge$	$(T' = T \wedge c' = 1) \wedge$	controlled$'$)	$\vee$

source $\wedge$	flow $\wedge$	invariant $\wedge$	target	
(controlled $\wedge$	$\left(\frac{dT}{dt} = (20 - 50c - T)\alpha\right) \wedge$	$(-22 \leq T \leq -18)$ $\wedge$	controlled$'$)	$\vee$
(defunct $\wedge$	$\left(\frac{dT}{dt} = (20 - T)\alpha\right)$ $\wedge$	true $\wedge$	defunct$'$)	

Unwinding

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

- 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
  $= \max_{(\sigma,\mathbf{x})\text{ initial}} \mathbf{P}^k_{(\sigma,\mathbf{x})}$.

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

SSMT Solving

SSMT algorithm

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

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

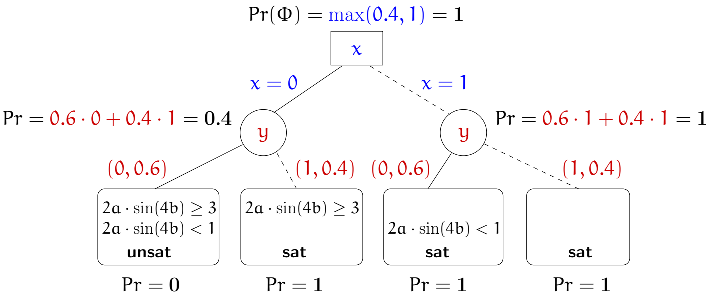
Solution: Take appropriate SMT solver, implant branching rules for quantifiers, add rigorous proof-tree pruning:

- **iSAT** solver for mixed Boolean and non-linear arithmetic problems [Fränzle, Herde, Ratschan, Schubert, Teige: 2006–]
- **odeSAT**: iSAT + ODE constraints [Eggers, Fränzle: 2008–]
- **iSAT/odeSAT** + **branching rules** for quantifier handling + **pruning rules** $\rightsquigarrow$ **SiSAT** [Eggers, Fränzle, Hermanns, Teige: QAPL 2008, HSCC 2008, CPAIOR 2008, ADHS 2009, JLAP 2010]

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*

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



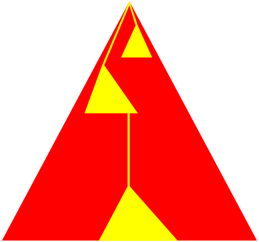
SSMT algorithm: Pruning rules

Scalability: Naive algorithm must traverse **whole quantifier tree** of size **exponential** in number of quantified variables

Goal: **Skip major parts** based on semantic inferences

Measures:

- Domain reduction by logical and numerical deductions
- Excluding conflicting (partial) assignments (conflict clauses)
- Thresholding [Littman 1999]
- Solution-directed backjumping [Majercik 2004]
- Probability-based value decision heuristics
- Probability learning (akin to memoization [Majercik, Littman 1998])
- Exploit desired accuracy of result
- For iterative BMC: Solution caching



Efficient quantifier handling: Thresholding

Given:

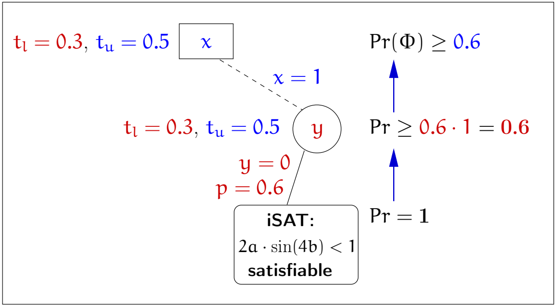
- $\Phi = \exists x \in \{0,1\} \,\text{Я}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}:$
  $(x > 0 \lor 2a \cdot \sin(4b) \geq 3) \land (y > 0 \lor 2a \cdot \sin(4b) < 1),$
- **lower threshold** $t_l = 0.3$,
- **upper threshold** $t_u = 0.5$.

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: Thresholding

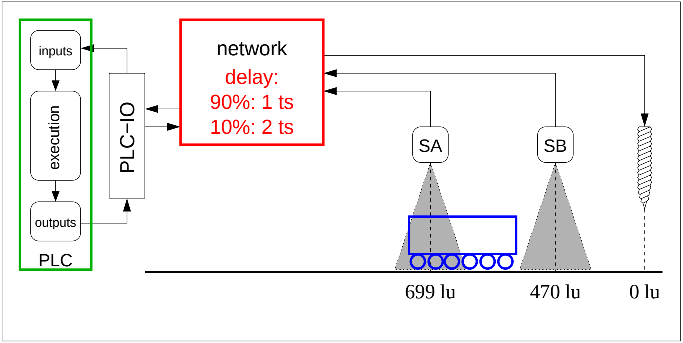
$$\Phi = \exists x \in \{0,1\}\ \text{Я}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \lor 2a \cdot \sin(4b) \geq 3) \land (y > 0 \lor 2a \cdot \sin(4b) < 1)$$



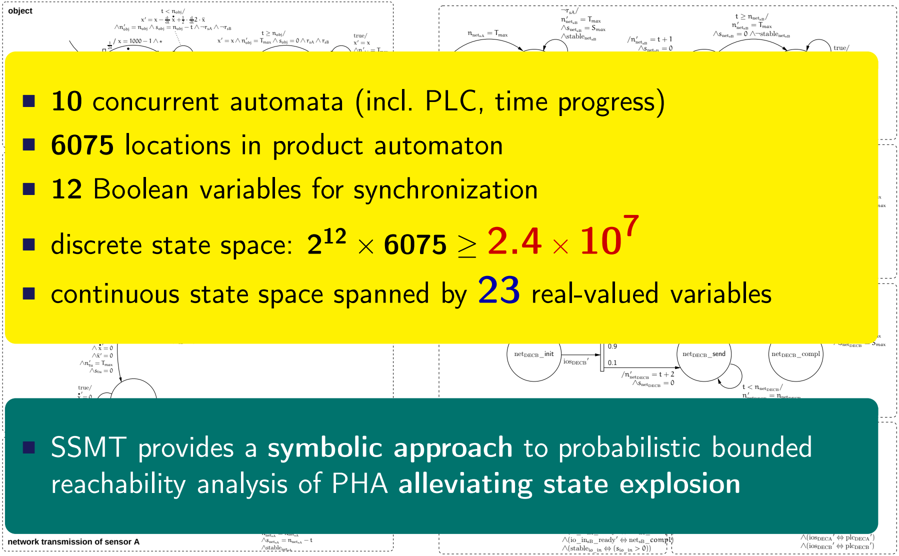
Pruning occurs

- when satisfaction probability of investigated branches $> t_u$,
- when probability mass of remaining branches $< t_l$,

Case study: Discrete-time system model



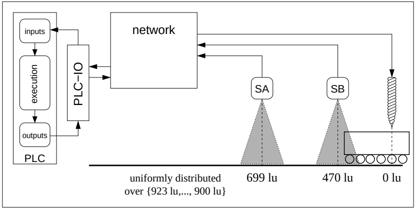
- **continuous dynamics** of conveyor: $\frac{ds}{dt} = v$, $\frac{dv}{dt} = a$
  $\leadsto s' = s + v \cdot \Delta t + \frac{1}{2} \cdot a \cdot \Delta t^2$, $v' = v + a \cdot \Delta t$
- **discrete computations** updating decel. a, communicating, ...
- **discrete probabilistic choices**: network delays
- **parallel composition** of subsystems: Sensors, network, PLC, PLC-IO, conveyor

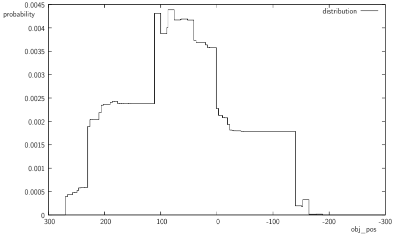


- **10** concurrent automata (incl. PLC, time progress)
- **6075** locations in product automaton
- **12** Boolean variables for synchronization
- discrete state space: $\mathbf{2^{12}} \times \mathbf{6075} \geq 2.4 \times 10^7$
- continuous state space spanned by 23 real-valued variables

- SSMT provides a **symbolic approach** to probabilistic bounded reachability analysis of PHA **alleviating state explosion**

• • •





Goal: Determine wh. probab. of stopping close to drilling pos. sufficient

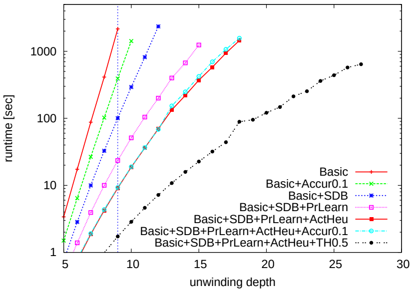
❶
- find BMC unwinding depth k s.t. object has stopped
- i.e., find k s.t. $\Pr(PBMC(k)) = \mathbf{1}$ with $TARGET(\mathbf{x}) :=$ tu_stop

$\rightsquigarrow$ holds for $k = \mathbf{44}$, total runtime 134 min (with thresholding)

❷

$TARGET(\mathbf{x})$	probability	runtime
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$=$ **0.397345[16,29]**	**71 min**
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$\geq$ **0.9**	**13 min**
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$\geq$ **0.95**	**11 min**

SSMT algorithm: Recent experimental results



Accuracy reduction far less effective than accuracy-preserving optimizations!

depth 9	Basic	B+Accur0.1	B+SDB	+PrLearn	+ActHeu	+TH0.5
runtime [sec]	2160.99	392.65	100.64	23.53	9.12	1.73
speed-up wrt. basic	1	**5.5**	**21**	**92**	**237**	**1249**
Result	exact	safe approx.	exact			

Quantitative Analysis 2:
From Falsification to Verification

Verifying Requirements on Expected Values

Rationale for Conditional Expectations

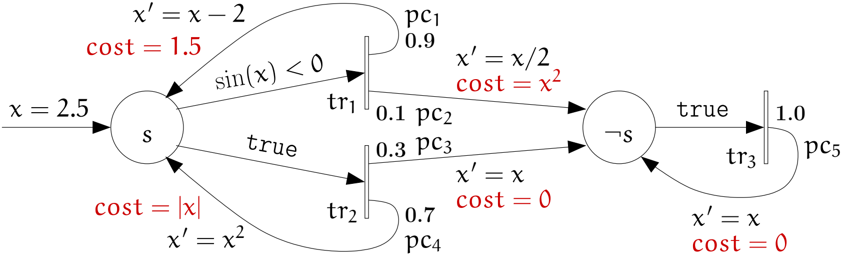
Observation:
- Reachability probabilities tend to 1 in the long run, thus are not a sufficiently discriminative measure in practice.
- Reliability engineers prefer other measures, like MTTF.

Question:
- Could we use BMC to compute MTTFs, etc., of PHA?

Result:
- Yes, with only minor adaptations to previous procedure.
- And this converts BMC into a **verification** procedure!

Sometimes, it suffices to just pose the right questions!

Expected Cost Values of Weighted PHA



Semantics: Step costs accumulate along runs.

Quest: Determine whether minimum (wrt. possible adversaries) expected cost for reaching a given set of target states is acceptably high, i.e. exceeds a threshold.

Example:

- Cost is step duration, target states = failures $\rightsquigarrow$ Expectation = MTTF
- Want to verify that MTTF exceeds requirements, irrespective of actual use case / adversary.

Can BMC verify that expectation on *monotonic* costs exceeds bound?

Expected Cost

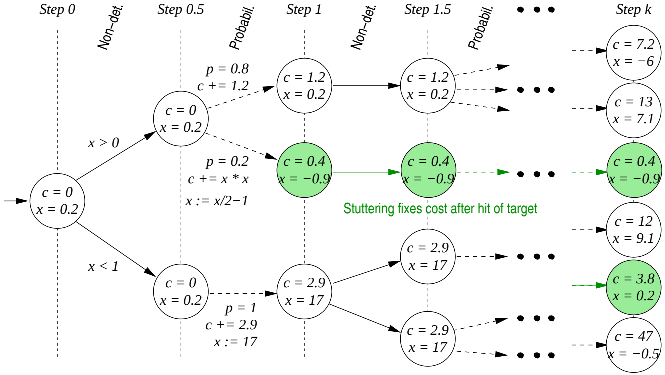
1. The cost expectation under adversary $adv : States \rightarrow \mathsf{Tr}$ is the least (wrt. the product order) solution of the equation system

$$\left(\mathrm{CE}_{adv}(z) = \begin{cases} 0 & \text{if } z \models target \\ \sum_{p \in PC_t} \underbrace{P(t)(p)}_{\substack{\text{probability}\\\text{of transition}\\\text{variant}}} \cdot \left(\begin{array}{c} \overbrace{cost(t,p,z)}^{\text{cost of transition}} \\ + \underbrace{\mathrm{CE}_{adv}(z')}_{\substack{\text{cost expect.}\\\text{of successor}}} \end{array} \right) & \text{if } z \not\models target \end{cases} \right)_{z \in States}$$

with $t = adv(z)$, and $(z, z') \models trans(t, p)$.

2. The minimum (maximum, resp.) cost expectation for reaching $target$ from state s is $\inf_{adv:States \rightarrow \mathsf{Tr}} \mathrm{CE}_{adv}(s)$ ($\sup_{adv:States \rightarrow \mathsf{Tr}} \mathrm{CE}_{adv}(s)$, resp.).

Unravelling the Probabilistic Transition Tree



- Costs on branches which have hit the target are known.
- Costs on "open" branches can be safely estimated from below by cost accumulated at the horizon.

⇝ Yields bounded cost expectation CE_k which converges monotonically against unbounded cost expectation when $k \longrightarrow \infty$.

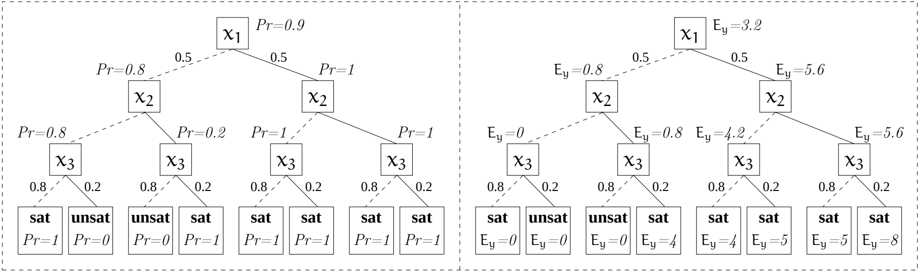
- CE_k is easy to encode in (suitably enhanced) SSMT

Empowering SSMT

$$Я_{[0\to 0.5, 1\to 0.5]} x_1 \in \{0,1\}\ \exists x_2 \in \{0,1\}\ Я_{[0\to 0.8, 1\to 0.2]} x_3 \in \{0,1\} :$$

$$(x_1 = 1 \vee x_2 = 1 \vee x_3 = 0) \wedge (x_1 = 1 \vee x_2 = 0 \vee x_3 = 1) \wedge (y = 4 \cdot x_1 + (x_2 + x_3)^2)$$

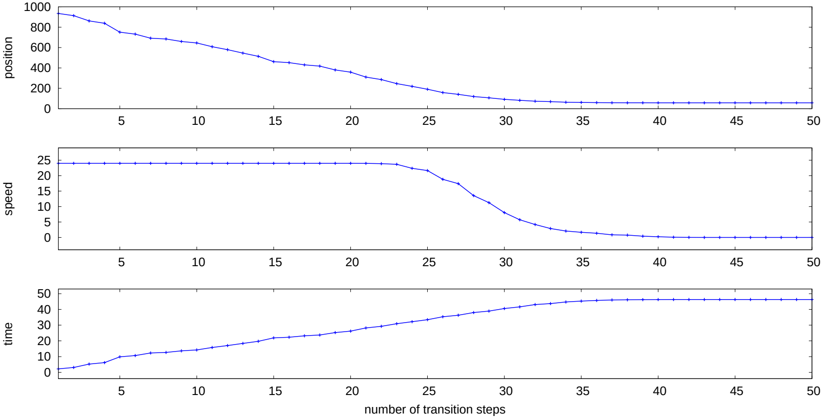
maximum probability of satisfaction — *maximum conditional expectation of* $y \in [0, 8]$



Caution: Pruning rules are substantially different with cost expectations!

Can thus compute $\min CE_k$ (with universal quantifiers) and $\max CE_k$ (with existential quantifiers) by SSMT.

Expectations vs. BMC Unwinding Depth
Benchmark Results from NAS Case Study



Monotonically decreasing costs have been normalized by multiplication with -1.

BMC-Based Verification

Observations:

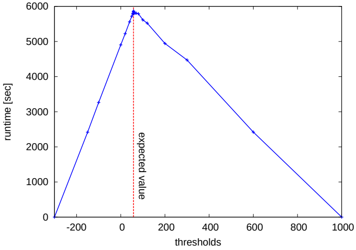
1. $\min \mathrm{CE}_k$ and $\max \mathrm{CE}_k$ can be determined by SSMT.
2. For $k \to \infty$, $\min \mathrm{CE}_k$ / $\max \mathrm{CE}_k$ converges
   - monotonically from below against the minimum / maximum cost expectation if step cost is non-negative,
   - monotonically from above against the minimum /maximum cost expectation if step cost is non-positive.

Consequence: Can employ the SSMT-encoding of CE_k together with SSMT-Solving for verification of the following proof obligations:

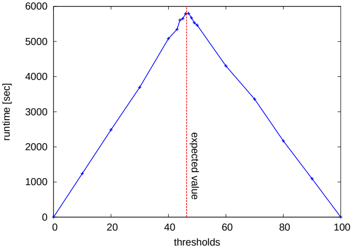
- Given a *non-negatively* weighted PHA A and $\theta \in \mathbb{Q}$, determine whether the minimum / maximum *unbounded* cost expectation $\mathrm{CE} > \theta$.
- Given a *non-positively* weighted PHA A and $\theta \in \mathbb{Q}$, determine whether the minimum / maximum *unbounded* cost expectation $\mathrm{CE} < \theta$.

Impact of Pruning
Benchmark Results from NAS Case Study

Drilling position



Time to stop



- Maximum runtime $\approx$ runtime for computing exact reach probability, no genuine overhead due to computing expectations.
- Pruning effective when deciding excess of expectation threshold.

Discussion

Ultimate Goal:

- Symbolic (wrt. both discr. and contin. state components) analysis of HA and PHA wrt. qualitative and quantitative requirements

Approach:

- Symbolic encoding of depth-bounded unwindings of the transition system as (stochastic) constraint problems involving contin. arithm. and ODEs;
- Extension of SAT-modulo-theory solving to non-linear constraints, ODEs, and randomized quantification problems.

Current results:

- SMT solver supporting non-linear (in)-equational constraints over the reals as theory, plus pre-post-relations mediated by ODEs
- SSMT solvers for the above, supporting alternating $\forall, \exists, \text{Я}$ quantifiers
- A symbolic procedure for bounded reachability of systems of discrete-time as well as dense-time HA and PHA
- A symbolic procedure for computing (in the limit exact) lower bounds for expected values of monotonic costs in PHA
- Largest probabilistic instance solved: Prob. reachability for dense-time model of NCS w. message loss, 12 parallel automata, yielding $2.008 \cdot 10^6$ discr. locations, 6 integers, 4 cont. variables, 2 governed by ODEs, unwinding depth 500 $\hat{=}$ 500 Я quantifiers (no non-determinism)

Future work:

- Quantitative verification by probabilistic interpolation