Automated Analysis of Hybrid Systems

A Constraint-Solving Perspective

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] Research Group Hybrid Systems
Department of Computing Science
Universität Oldenburg, Germany

What is a hybrid system?

Hybrid (griech.) bedeutet überheblich, hochmütig, vermessen.

Hybrid (from Greece) means arrogant, presumptuous.

After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

What is a hybrid system?

Hybrid (griech.) bedeutet überheblich, hochmütig, vermessen. Weitere Inhalte [insbes. im wiss. Sprachgebrauch] sind später hinein interpretiert worden.

Hybrid (from Greece) means arrogant, presumptuous. Other interpretations [in particular, in scientific jargon] have been added later.

After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

What is a hybrid system?

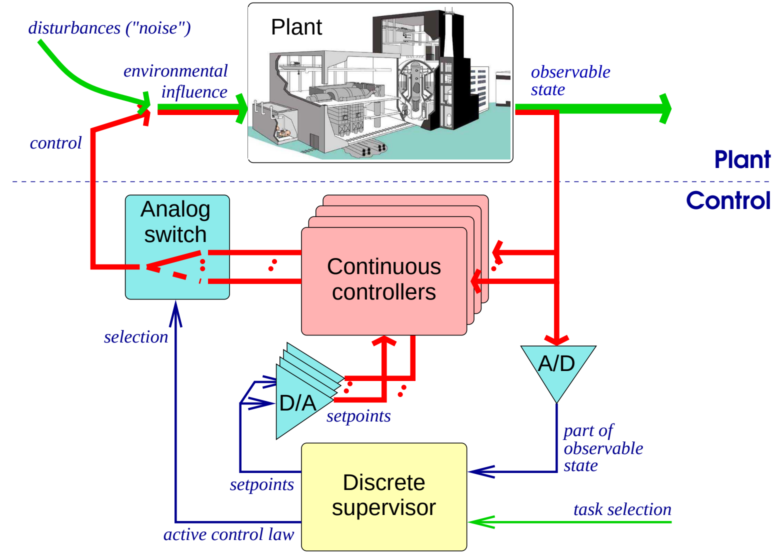
Hybrid (griech.) bedeutet überheblich, hochmütig, vermessen. Weitere Inhalte [insbes. im wiss. Sprachgebrauch] sind später hinein interpretiert worden.

Hybrid (from Greece) means arrogant, presumptuous. Other interpretations [in particular, in scientific jargon] have been added later.

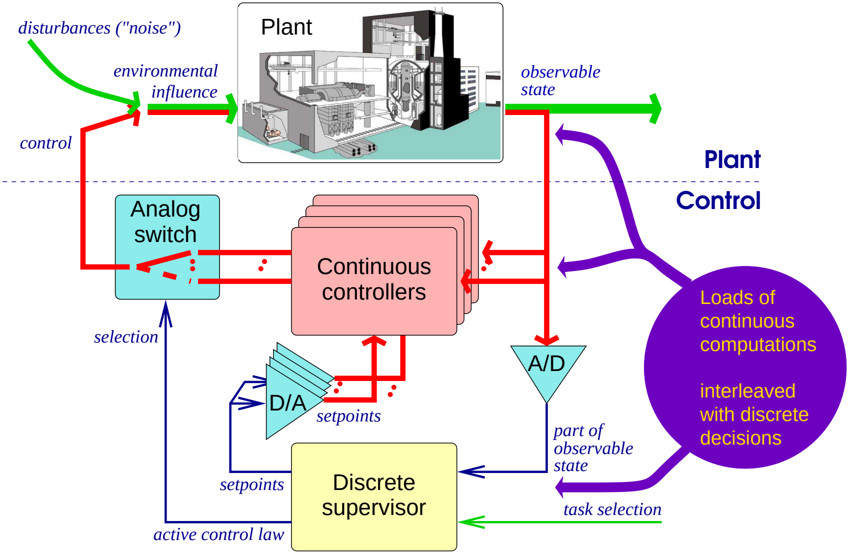
After H. Menge: Griechisch/Deutsch, Langenscheidt 1984

$\Rightarrow$ when you try to verify hybrid systems,
be prepared that they may act like a prima donna!

Hybrid Systems



Hybrid Systems

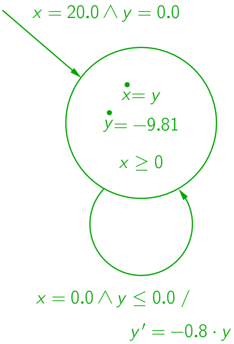


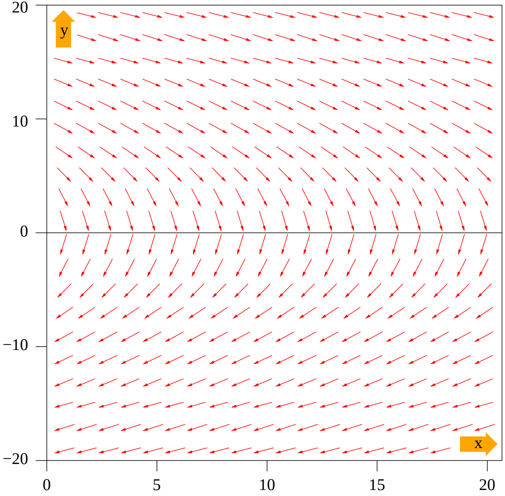
Hybrid systems

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 Formal Model: Hybrid Automata



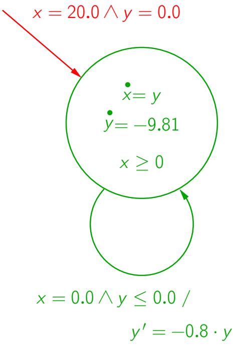


x : vertical position of the ball

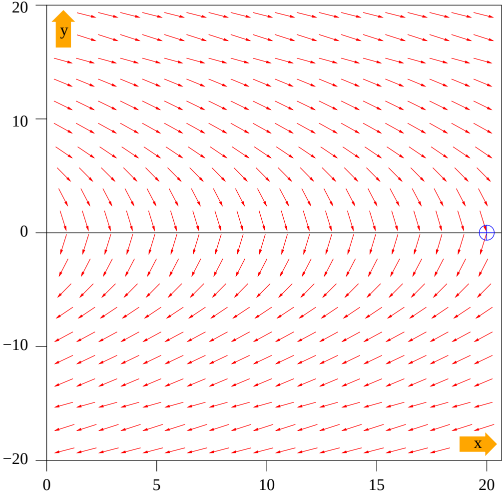
y : velocity

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

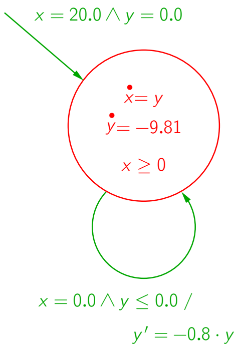
A Formal Model: Hybrid Automata

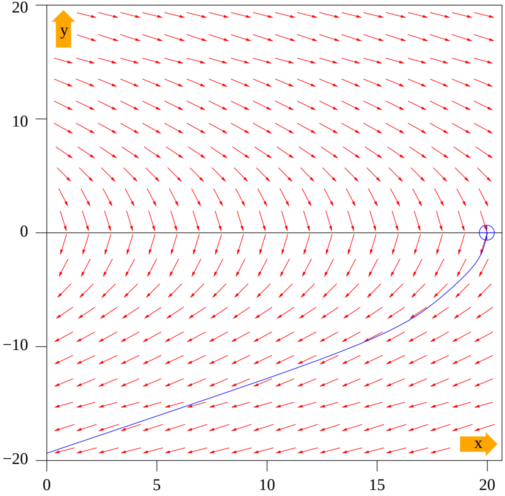


- x : vertical position of the ball
- y : velocity
  - $y > 0$ ball is moving up
  - $y < 0$ ball is moving down



A Formal Model: Hybrid Automata



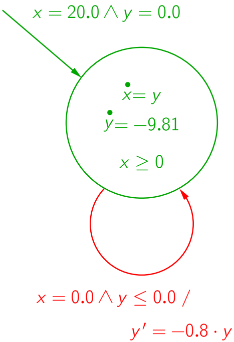


x : vertical position of the ball

y : velocity

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

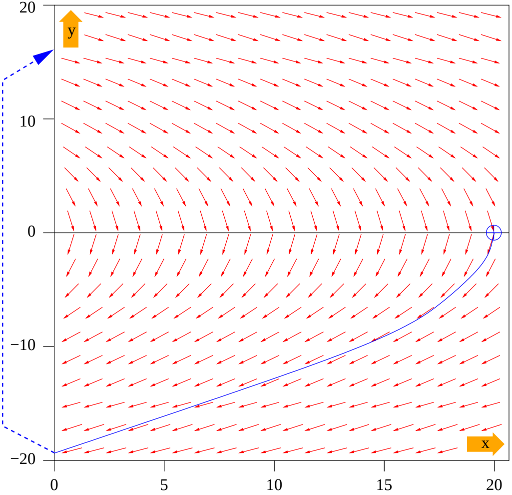
A Formal Model: Hybrid Automata



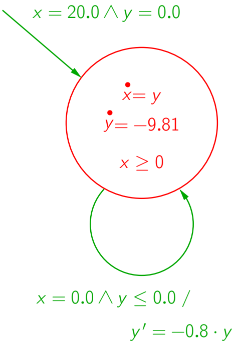
x : vertical position of the ball

y : velocity

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



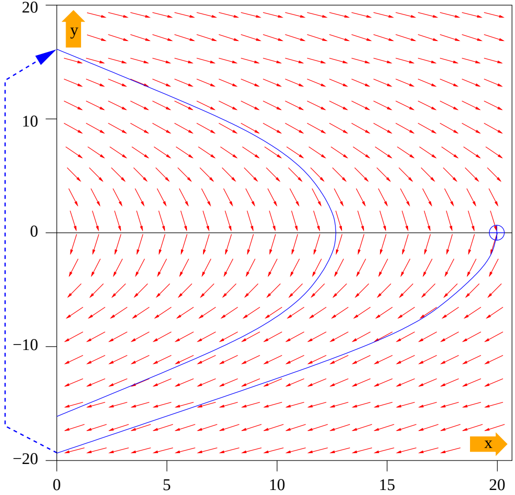
A Formal Model: Hybrid Automata



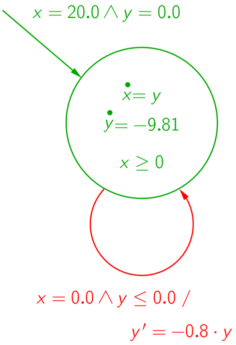
x : vertical position of the ball

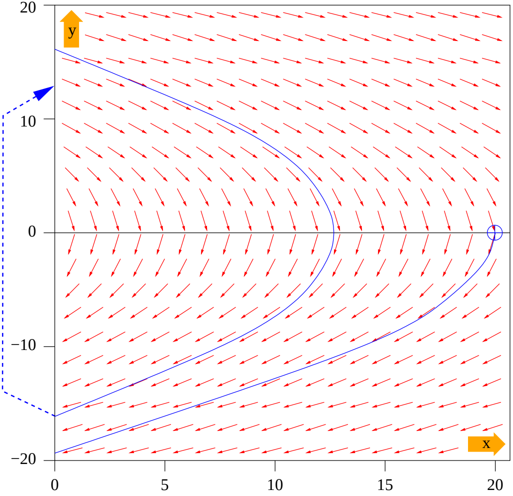
y : velocity

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



A Formal Model: Hybrid Automata



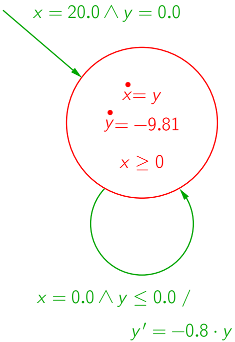


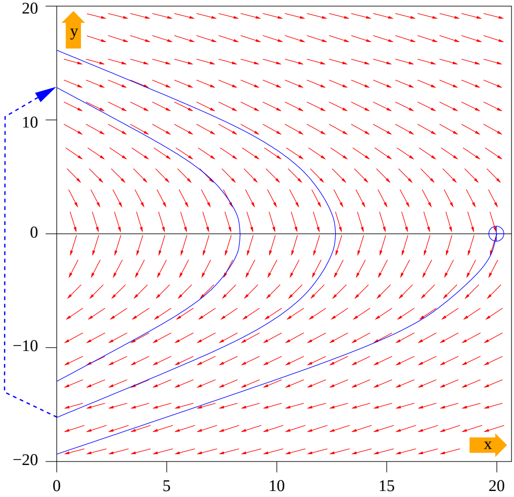
x : vertical position of the ball

y : velocity

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

A Formal Model: Hybrid Automata



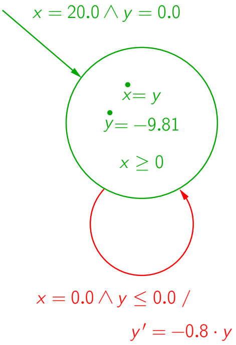


x : vertical position of the ball

y : velocity

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

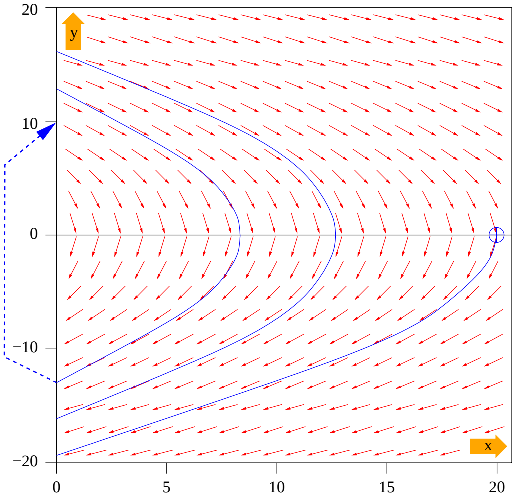
A Formal Model: Hybrid Automata



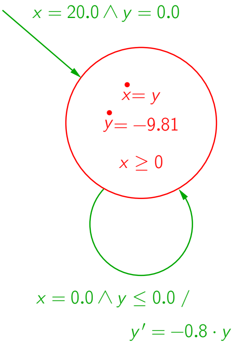
x : vertical position of the ball

y : velocity

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



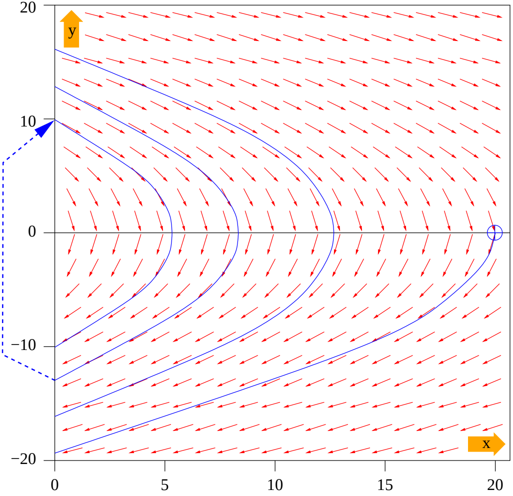
A Formal Model: Hybrid Automata



x : vertical position of the ball

y : velocity

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



State and Dimension Explosion

Number of continuous variables linear in number of cars

- Positions, speeds, accelerations,
- torque, slip, ...

Number of discrete states exponential in number of cars

- Operational modes, control modes,
- state of communication subsystem, ...

Size-dependent dynamics

- Latency in ctrl. loop depends on number of cars due to communication subsystem.
- Coupled dynamics yields long hidden channels chaining signal transducers.

State and Dimension Explosion

Number of continuous variables linear in number of cars

- Positions, speeds, accelerations,
- torque, slip, ...

Number of discrete states exponential in number of cars

- Operational modes, control modes,
- state of communication subsystem, ...

Size-dependent dynamics

- Latency in ctrl. loop depends on number of cars due to communication subsystem.
- Coupled dynamics yields long hidden channels chaining signal transducers.

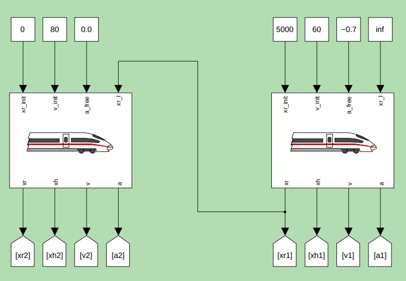
$\Rightarrow$ Need a scalable approach
$\Rightarrow$ Let's try to achieve this through strictly symbolic methods.

Industrial Modelling Paradigms by Example

Train Separation in ETCS Level 3

ETCS Movement Authorization

Example: Train Separation in Absolute Braking Distance

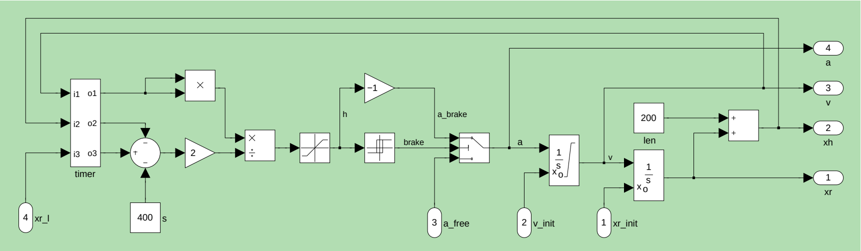


Minimal admissible distance d between two successive trains equals braking distance d_b of the second train plus a safety distance S.

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.

Analysis of Matlab/Simulink Model

Model of Controller & Train Dynamics



Property to be checked: Does the controller guarantee that collisions are averted in any possible scenario of use?

Worst-Case Analysis: Running at top speed...

- With $v_{\mathrm{max}} = 83.4\frac{\mathrm{m}}{\mathrm{s}}$ and $a_{\mathrm{on}} = -0.7\frac{\mathrm{m}}{\mathrm{s}^2}$, due to $s = \frac{1}{2}\frac{v^2}{a}$, automatic braking should commence at distance

$$s_{\mathrm{on}} = \frac{1}{2}\frac{\left(83.4\frac{\mathrm{m}}{\mathrm{s}}\right)^2}{-0.7\frac{\mathrm{m}}{\mathrm{s}^2}} = -4968\,\mathrm{m}$$

Worst-Case Analysis: Running at top speed...

- With $v_{\mathrm{max}} = 83.4\frac{\mathrm{m}}{\mathrm{s}}$ and $a_{\mathrm{on}} = -0.7\frac{\mathrm{m}}{\mathrm{s}^2}$, due to $s = \frac{1}{2}\frac{v^2}{a}$, automatic braking should commence at distance

$$s_{\mathrm{on}} = \frac{1}{2}\frac{\left(83.4\frac{\mathrm{m}}{\mathrm{s}}\right)^2}{-0.7\frac{\mathrm{m}}{\mathrm{s}^2}} = -4968\,\mathrm{m}$$

- In the worst case, initiating braking $8\,\mathrm{s}$ late, we may have travelled $8\,\mathrm{s} \cdot 83.4\frac{\mathrm{m}}{\mathrm{s}} = 667\mathrm{m}$ beyond that horizon, thus commencing deceleration at

$$s_{\mathrm{on,act}} = -4968\,\mathrm{m} + 667\,\mathrm{m} = -4301\,\mathrm{m}$$

Worst-Case Analysis: Running at top speed...

- With $v_{\mathrm{max}} = 83.4\frac{\mathrm{m}}{\mathrm{s}}$ and $a_{\mathrm{on}} = -0.7\frac{\mathrm{m}}{\mathrm{s}^2}$, due to $s = \frac{1}{2}\frac{v^2}{a}$, automatic braking should commence at distance

$$s_{\mathrm{on}} = \frac{1}{2}\frac{\left(83.4\frac{\mathrm{m}}{\mathrm{s}}\right)^2}{-0.7\frac{\mathrm{m}}{\mathrm{s}^2}} = -4968\,\mathrm{m}$$

- In the worst case, initiating braking $8\,\mathrm{s}$ late, we may have travelled $8\,\mathrm{s} \cdot 83.4\frac{\mathrm{m}}{\mathrm{s}} = 667\mathrm{m}$ beyond that horizon, thus commencing deceleration at

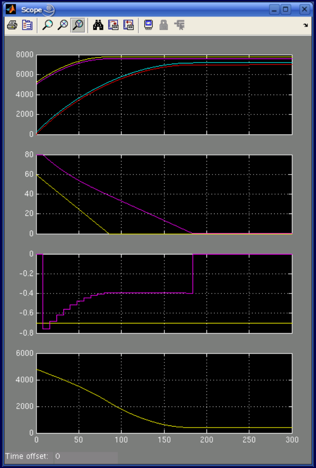
$$s_{\mathrm{on,act}} = -4968\,\mathrm{m} + 667\,\mathrm{m} = -4301\,\mathrm{m}$$

- Due to $a = \frac{1}{2}\frac{v^2}{s}$, the corresponding deceleration is

$$a_{\mathrm{act}} = \frac{1}{2}\frac{\left(83.4\frac{\mathrm{m}}{\mathrm{s}}\right)^2}{-4301\,\mathrm{m}} = -0.8\frac{\mathrm{m}}{\mathrm{s}^2} \gg -1.4\frac{\mathrm{m}}{\mathrm{s}^2}$$

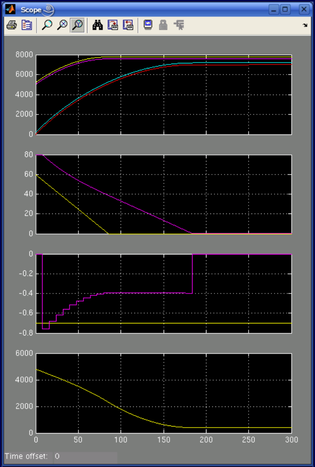
Analysis of Matlab/Simulink Model

Simulation of the Model

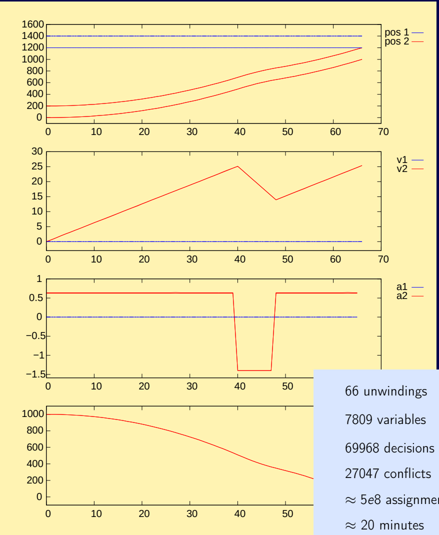


Analysis of Matlab/Simulink Model

Simulation of the Model



Error Trace found by HySAT / iSAT



66 unwindings

7809 variables

69968 decisions

27047 conflicts

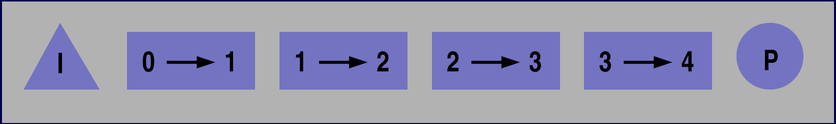
≈ 5e8 assignments

≈ 20 minutes

SAT Modulo Theory

An engine for bounded model checking of linear hybrid automata

Bounded Model Checking (BMC)



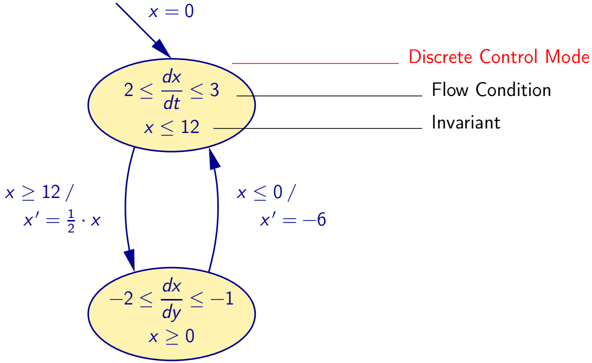
Method:

- 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 $k = 0, 1, 2, \dots$ until an error trace is found or tired

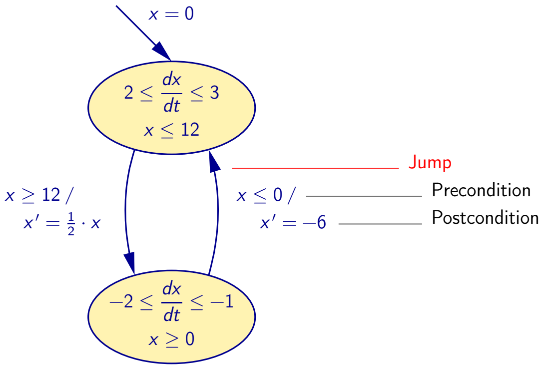
Bounded Model Checking (BMC) algorithm

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.

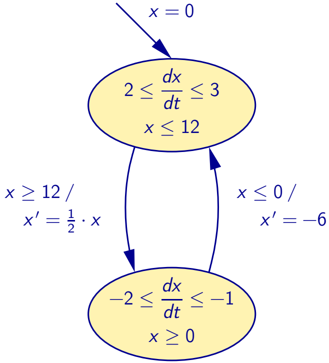
Linear Hybrid Automata (LHA)

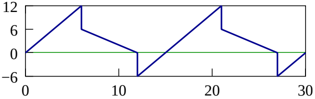


Linear Hybrid Automata (LHA)

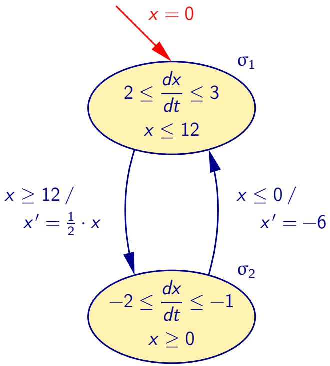


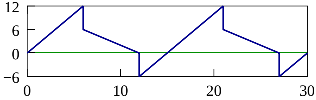
Linear Hybrid Automata (LHA)





BMC of Linear Hybrid Automata

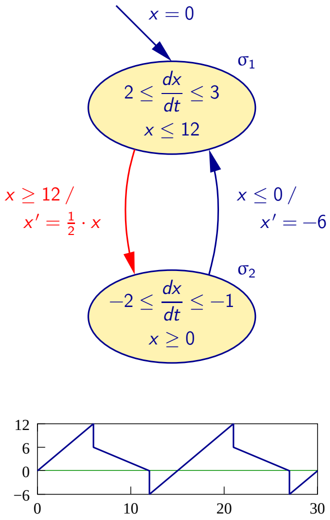




Initial state:

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

BMC of Linear Hybrid Automata



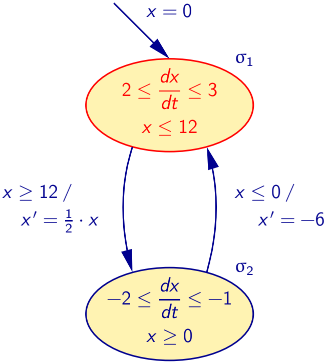
Initial state:

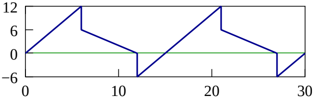
$$\sigma_1^0 \;\wedge\; \neg\sigma_2^0 \;\wedge\; x^0 = 0.0$$

Jumps:

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

BMC of Linear Hybrid Automata





Initial state:

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

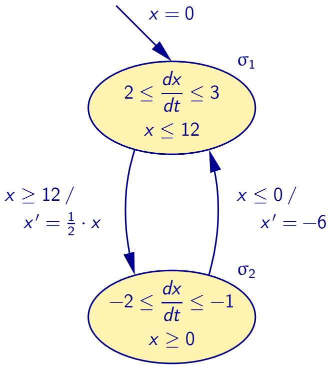
Jumps:

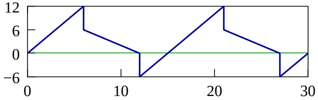
$$\sigma_1^i \wedge \sigma_2^{i+1} \;\rightarrow (x^i \geq 12) \;\wedge\; (x^{i+1} = 0.5 \cdot x^i) \;\wedge\; t^i = 0$$

Flows:

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

BMC of Linear Hybrid Automata





Initial state:

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

Jumps:

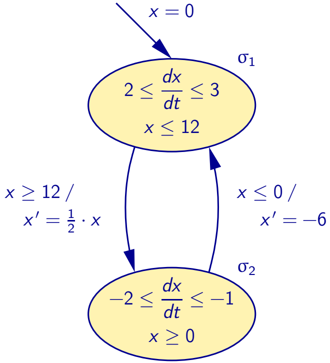
$$\sigma_1^i \wedge \sigma_2^{i+1} \;\rightarrow (x^i \ge 12) \;\wedge\; (x^{i+1} = 0.5 \cdot x^i) \;\wedge\; t^i = 0$$

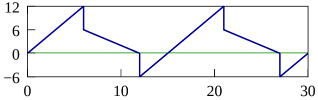
Flows:

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

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

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.
$\rightsquigarrow$ **SAT modulo theory**

(Old-fashioned) DPLL Procedure

$$\begin{aligned} & (x \vee y \vee z) \\ \wedge\ & (\bar{x} \vee y) \\ \wedge\ & (\bar{y} \vee z) \\ \wedge\ & (\bar{x} \vee \bar{y} \vee \bar{z}) \\ \wedge\ & (x \vee \bar{y} \vee \bar{z}) \end{aligned}$$

(Old-fashioned) DPLL Procedure

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

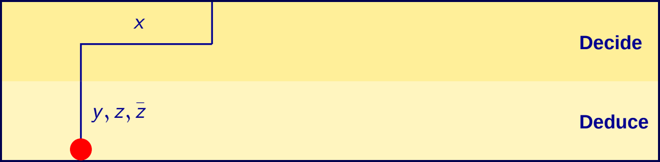
(Old-fashioned) DPLL Procedure

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

x	**Decide**
$y, z, \bar{z}$	**Deduce**

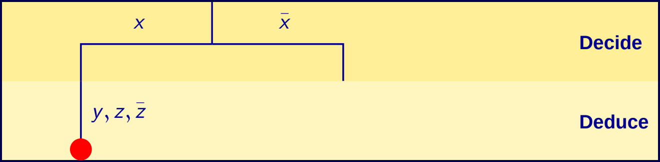
(Old-fashioned) DPLL Procedure

$$
\begin{aligned}
& (x \vee y \vee z) \\
\wedge\ & (\bar{x} \vee y) \\
\wedge\ & (\bar{y} \vee z) \\
\wedge\ & (\bar{x} \vee \bar{y} \vee \bar{z}) \\
\wedge\ & (x \vee \bar{y} \vee \bar{z})
\end{aligned}
$$



(Old-fashioned) DPLL Procedure

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



(Old-fashioned) DPLL Procedure

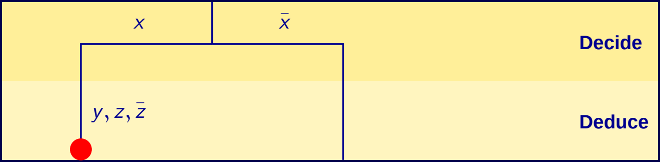
$(x \vee y \vee z)$

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

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

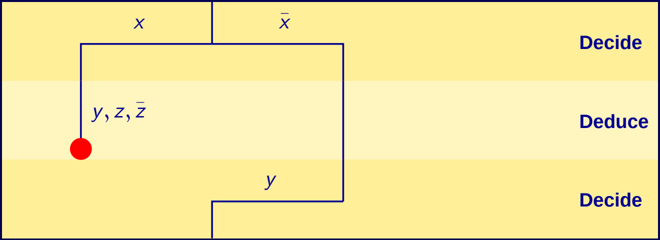
$\wedge \; (\bar{x} \vee \bar{y} \vee \bar{z})$

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



(Old-fashioned) DPLL Procedure

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



(Old-fashioned) DPLL Procedure

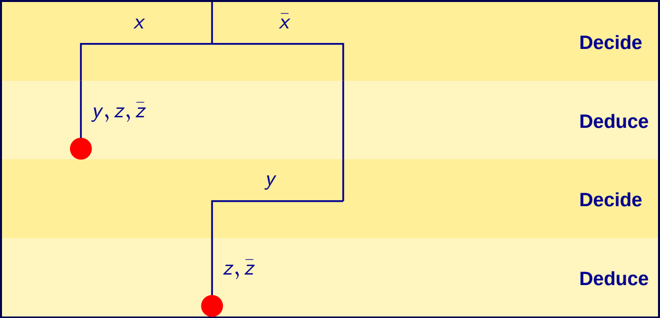
$(x \vee y \vee z)$

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

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

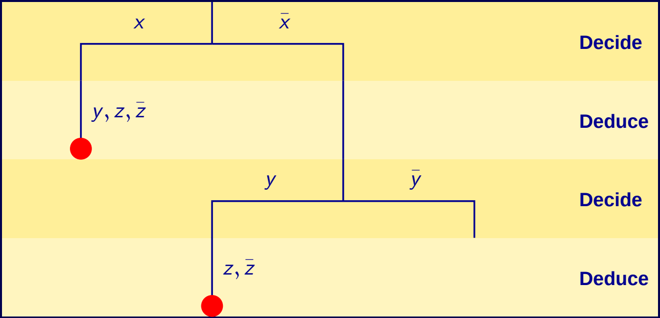
$\wedge\ (\bar{x} \vee \bar{y} \vee \bar{z})$

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



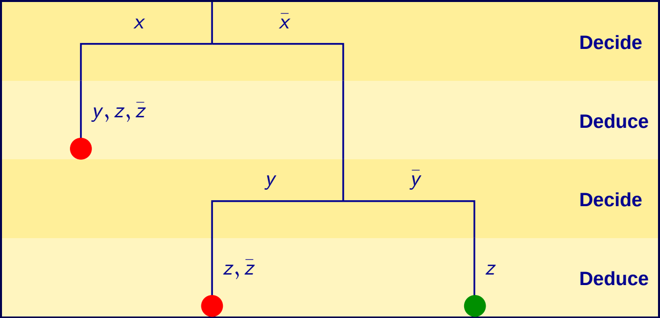
(Old-fashioned) DPLL Procedure

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

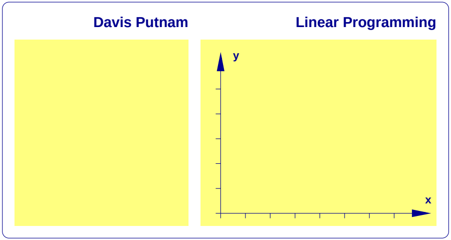


(Old-fashioned) DPLL Procedure

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



(Simplified) SAT Modulo Theory Scheme: LinSAT



Input formula:

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

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

$2e + C + D \geq 2$

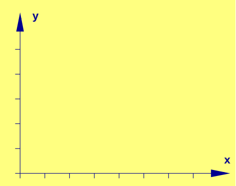
$2f + A + B \geq 2$

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

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

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

Linear Programming



Input formula:

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

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

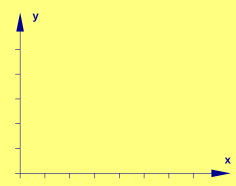
$C + D \geq 2$

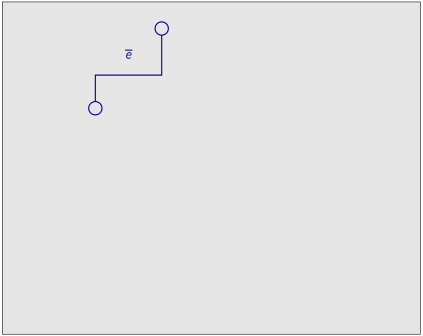
$2f + A + B \geq 2$

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

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

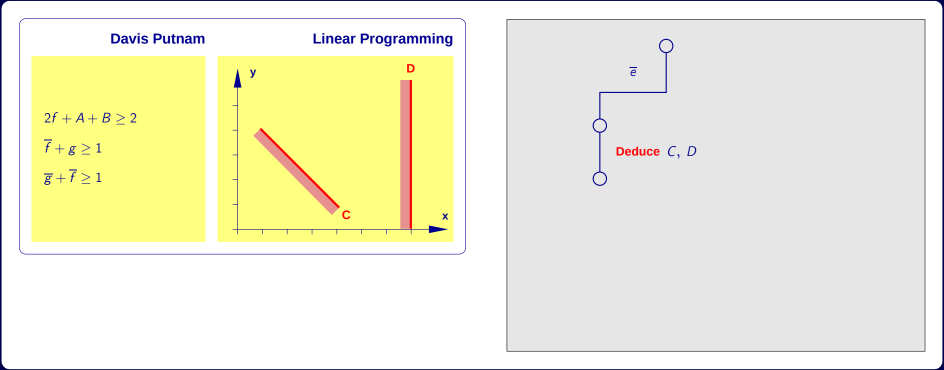
Linear Programming





DPLL search

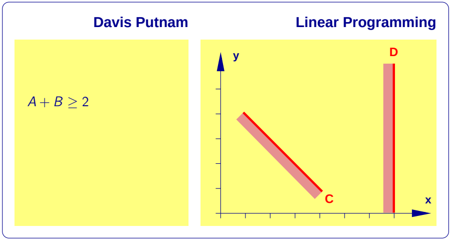
1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

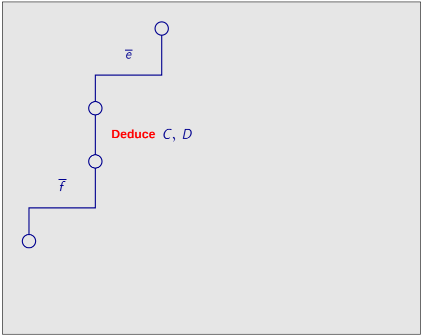


DPLL search

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

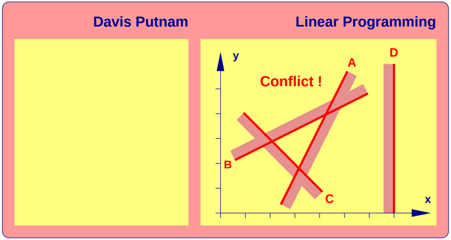




DPLL search

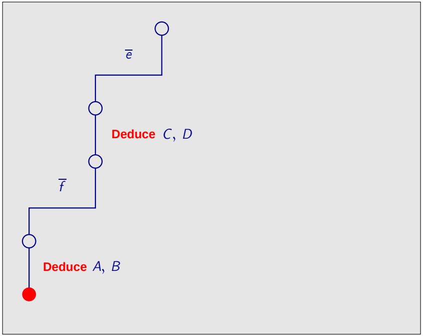
1. traversing possible truth-value assignments of Boolean part
2. incrementally (de-)constructing a *conjunctive* arithmetic constraint system
3. querying external solver to determine consistency of arithm. constr. syst.

(Simplified) SAT Modulo Theory Scheme: LinSAT



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

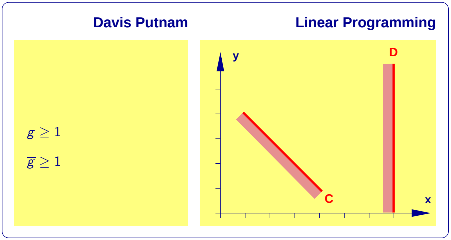
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



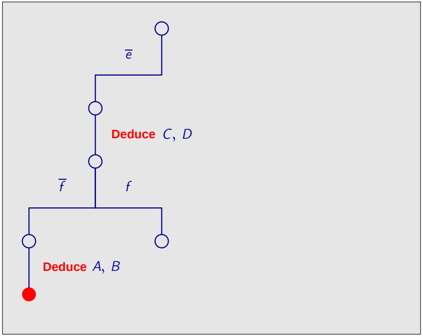
DPLL search

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



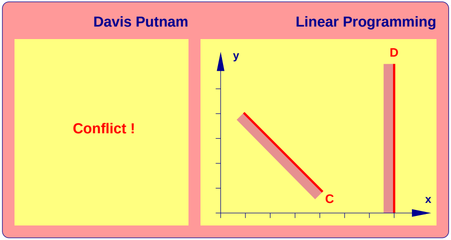
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



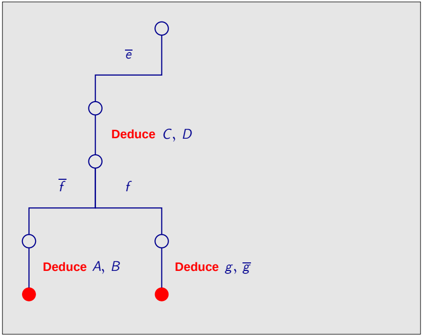
DPLL search

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

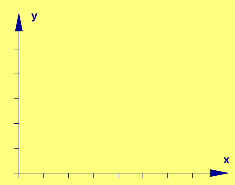
Davis Putnam

$2f + A + B \geq 2$

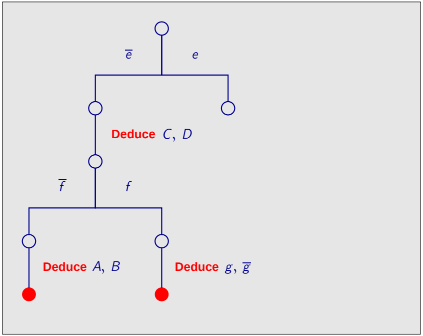
$\overline{g} + \overline{f} \geq 1$

$2g + C + D \geq 3$

Linear Programming



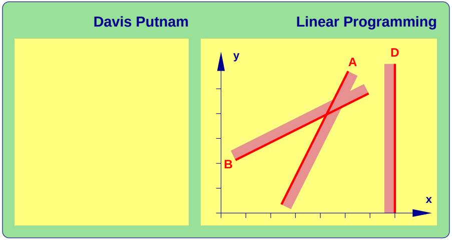
Learned conflict clause: $\overline{A} + \overline{B} + \overline{C} \geq 1$



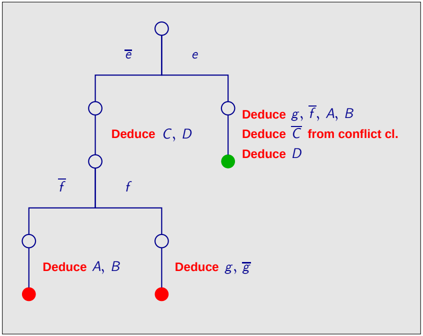
DPLL search

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.

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

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

is infeasible, we want a subset $I \subseteq \{1, \dots, n\}$ 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.

Extensions & Optimizations

DPLL(T): If the T solver can itself do fwd. inference, it cannot only prune the search tree through conflict detection, but also through constraint propagation:

1. SAT solver assigns truth values to subset $C \subset A$ of the set A of constraints occurring in the input formula,
2. T solver finds them to be consistent *and* to imply a truth value assignment to further T constraints $D \subseteq A \setminus C$,
3. these truth-value assignments are performed in the SAT solver store before resuming SAT solving.

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-]
  - ...

SAT modulo theory for LinSAT

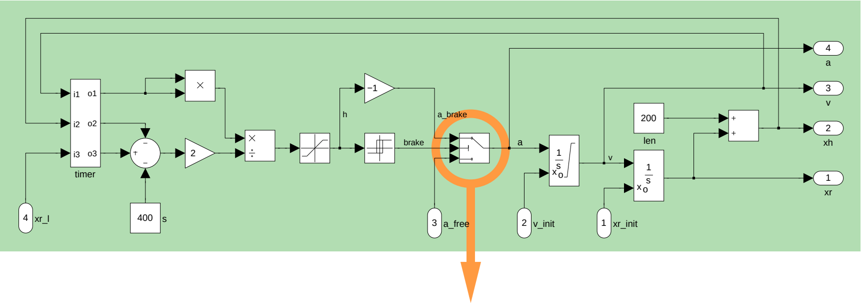
- SAT modulo theory solvers reasoning over linear arithmetic as a theory are readily available: E.g.,
  - LPSAT [Wolfman & Weld, 1999]
  - ICS [Filliatre, Owre, Rueß, Shankar 2001], Simplics [de Moura, Dutertre 2005], Yices [Dutertre, de Moura 2006]
  - MathSAT [Audemard, Bertoli, Cimatti, Kornilowicz, 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.

Hybrid BMC in Practice

ETCS Train separation in HySAT II

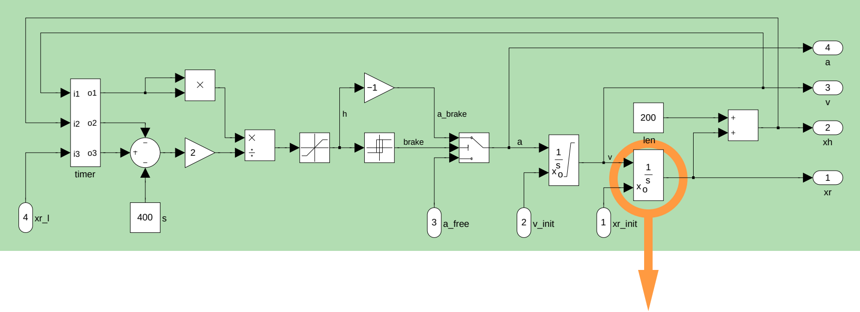
Reduction of Matlab/Simulink to Constraints

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;
```

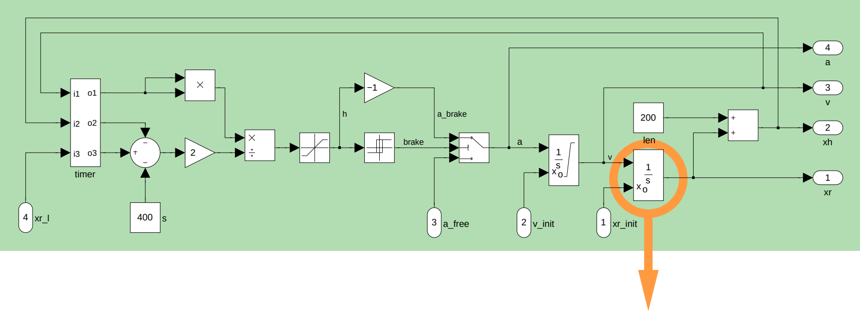
Translation to HySAT




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

Reduction of Matlab/Simulink to Constraints

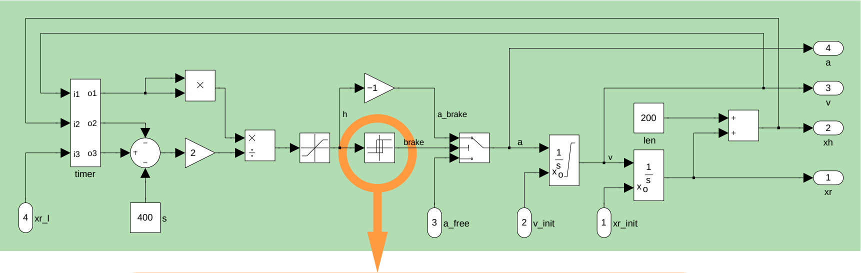
Translation to HySAT




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

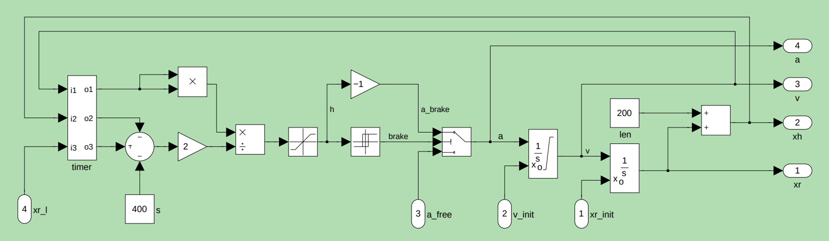
... could also be higher-order Taylor approximation with safe remainder.

Translation to HySAT



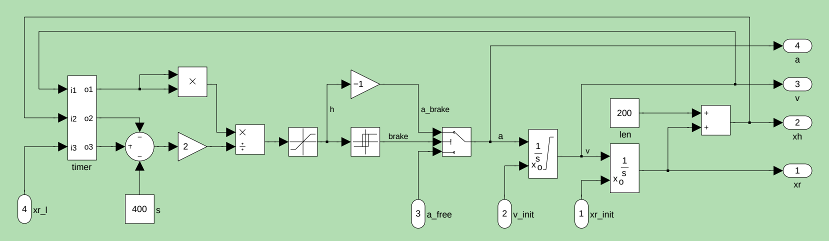

```
- Relay block: When the relay is on, it remains on until the input
- drops below the value of the switch off point parameter. When the
- relay is off, it remains off until the input exceeds the value of
- the switch on point parameter.

(!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);
```

Reduction of Matlab/Simulink to Constraints



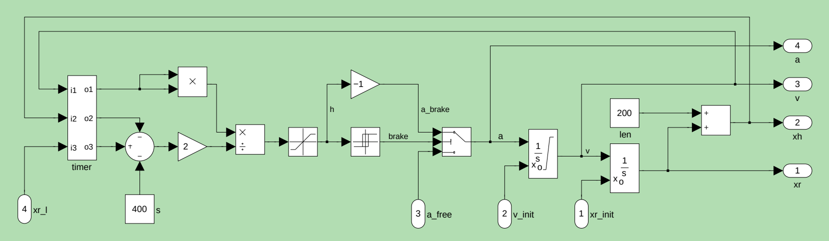
- The model contains non-linearities due to $a = \frac{1}{2}\frac{v^2}{s}$

Reduction of Matlab/Simulink to Constraints



- The model contains non-linearities due to $a = \frac{1}{2}\frac{v^2}{s}$
- Thus not expressible in LinSAT

Reduction of Matlab/Simulink to Constraints

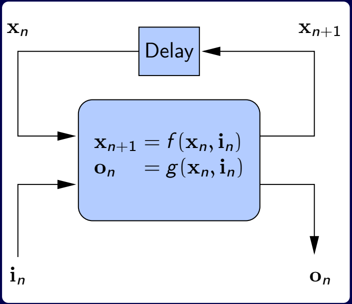


- The model contains non-linearities due to $a = \frac{1}{2}\frac{v^2}{s}$
- Thus not expressible in LinSAT

$\Rightarrow$ Need a more comprehensive solving technology than DPLL(LP), able to deal with non-linear constraints

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.

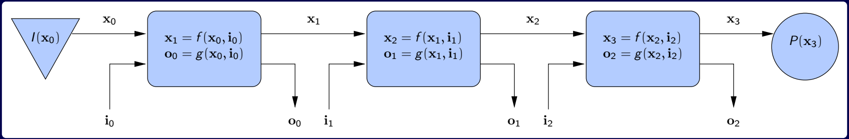
Goal:

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

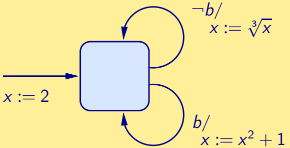


- Use appropriate procedure to "decide" satisfiability of the formula

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

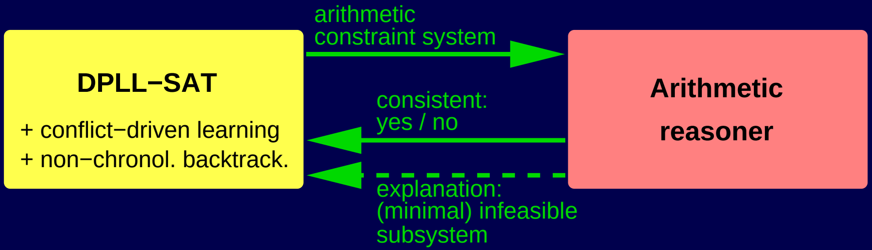
COUNTEREXAMPLE

```
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]
```

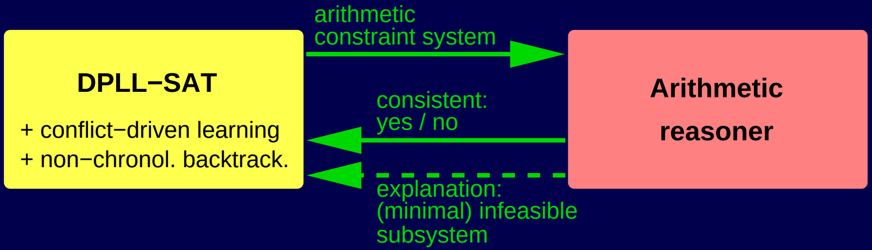
Satisfiability solving in undecidable arithmetic domains

iSAT algorithm

Classical Lazy TP Layout



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

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

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 Propagation (1)

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

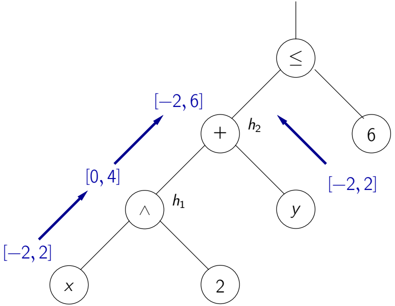
$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{lcl} c_1: & & h_1 \triangleq x \wedge 2 \\ c_2: & \wedge & h_2 \triangleq h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

Interval Constraint Propagation (1)

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

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

- “Forward” interval propagation yields justification for constraint satisfaction:



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

$$\Downarrow$$

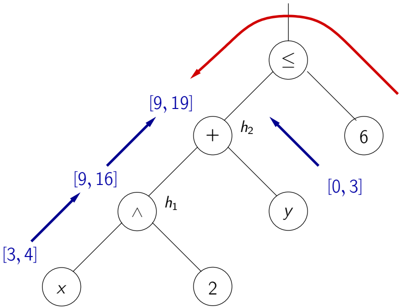
$h_2 \le 6$ is satisfied in box

Interval Constraint Propagation (1)

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

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

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



$$\begin{array}{ll} & x \in [3, 4] \\ \wedge & y \in [0, 3] \end{array}$$

$$\Downarrow$$

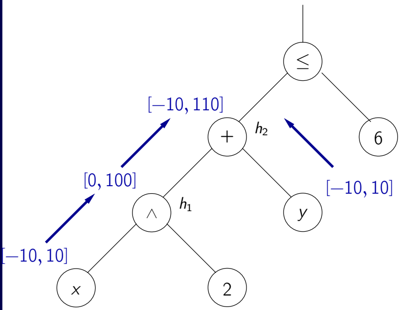
$h_2 \leq 6$ is unsat. in box

Interval Constraint Propagation (1)

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

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

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



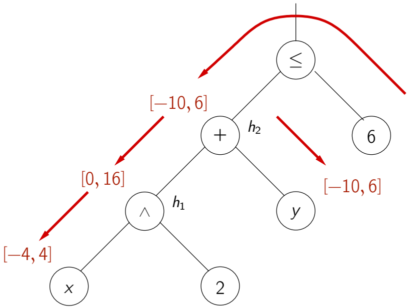
$$\begin{array}{rl} & x \in [-10, 10] \\ \wedge & y \in [-10, 10] \end{array}$$

Interval Constraint Propagation (1)

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

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

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



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

$$\Downarrow$$

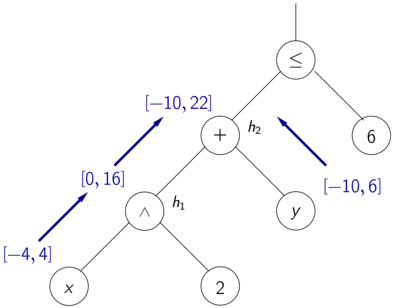
$$x \in [-4, 4] \wedge y \in [-10, 6]$$

Interval Constraint Propagation (1)

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

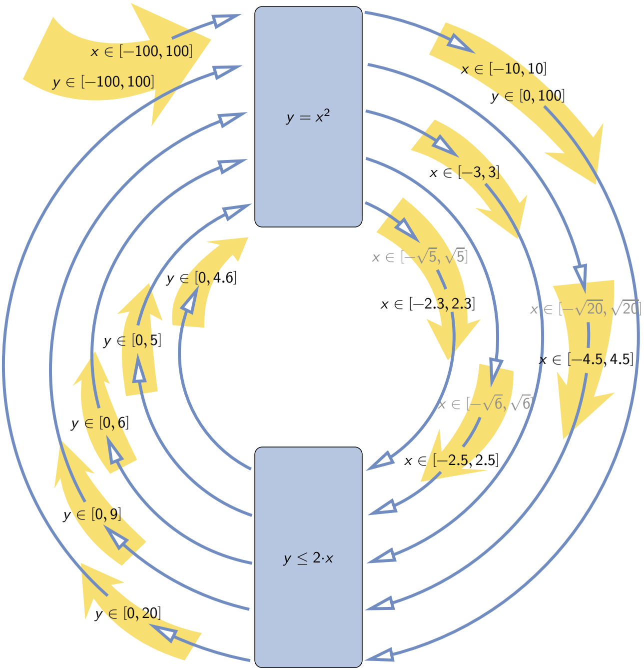
$$x^2 + y \leq 6 \quad \rightsquigarrow \quad \begin{array}{rcl} c_1: & & h_1 \mathrel{\widehat{=}} x \mathbin{\wedge} 2 \\ c_2: & \wedge & h_2 \mathrel{\widehat{=}} h_1 + y \\ & \wedge & h_2 \leq 6 \end{array}$$

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



Constraint is not satisfied by the contracted box!

$$\begin{array}{rl} & x \in [-4, 4] \\ \wedge & y \in [-10, 6] \end{array}$$



$x \in [-100, 100]$
$y \in [-100, 100]$
$y = x^2$
$x \in [-10, 10]$
$y \in [0, 100]$
$x \in [-3, 3]$
$x \in [-\sqrt{5}, \sqrt{5}]$
$x \in [-2.3, 2.3]$
$x \in [-\sqrt{20}, \sqrt{20}]$
$x \in [-4.5, 4.5]$
$x \in [-\sqrt{6}, \sqrt{6}]$
$x \in [-2.5, 2.5]$
$y \in [0, 4.6]$
$y \in [0, 5]$
$y \in [0, 6]$
$y \in [0, 9]$
$y \in [0, 20]$
$y \leq 2 \cdot x$

Interval contraction

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

Thus, interval contraction provides a highly incomplete deduction system:

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

Interval contraction

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

Thus, interval contraction provides a highly incomplete deduction system:

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

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

How HySAT works

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

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 : $\wedge\ h_3 = h_1 + h_2$

- 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 symbol
- 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 HySAT 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 HySAT works

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

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

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

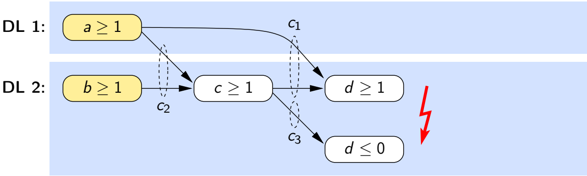
c_4 : $\land\ (b \lor x \geq -2)$

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

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

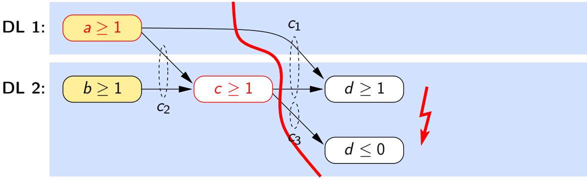
c_7 : $\land\ h_2 = -2 \cdot y$

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



How HySAT 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 HySAT 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: $\boxed{a \geq 1} \xrightarrow[c_9]{} \boxed{c \leq 0} \xrightarrow[c_2]{} \boxed{b \leq 0} \xrightarrow[c_4]{} \boxed{x \geq -2}$

How HySAT works

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

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

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

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

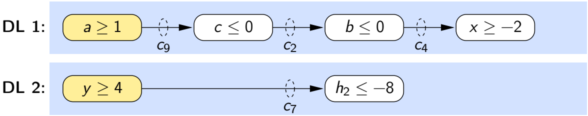
$c_5: \land (x \geq 4 \lor y \leq 0 \lor h_3 \geq 6.2)$

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

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

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

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



How HySAT works

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

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

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

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

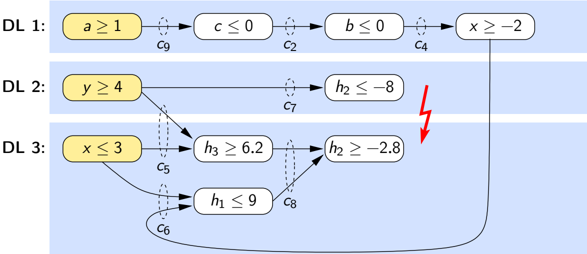
$c_5:\quad \wedge\ (x \geq 4 \vee y \leq 0 \vee h_3 \geq 6.2)$

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

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

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

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



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

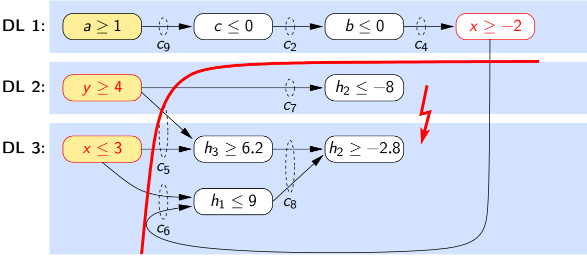
c_6 : $\land\ h_1 = x^2$

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



$\leftarrow$ conflict clause = symbolic description

of a rectangular region of the search space

which is excluded from future search

How HySAT works

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

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

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

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

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

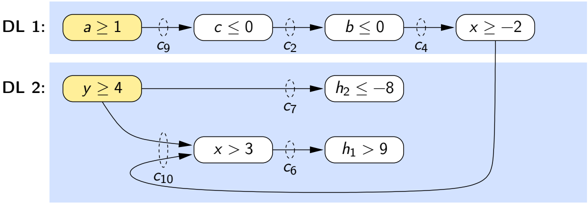
$c_6:\quad \land\ h_1 = x^2$

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

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

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

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



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

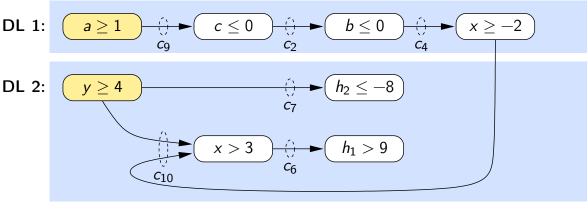
c_6 : $\land\ h_1 = x^2$

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



- 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

How HySAT works

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

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

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

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

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

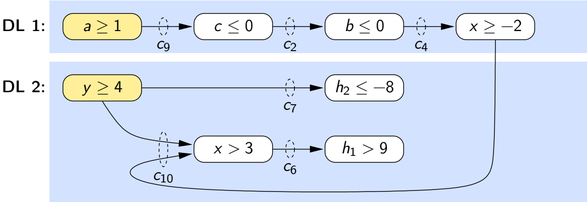
$c_6 : \land\ h_1 = x^2$

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

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

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

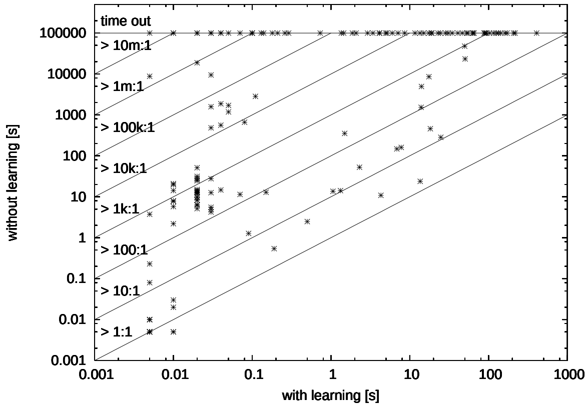
$c_{10} : \land\ (x < -2 \lor y < 3 \lor x > 3)$



- 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

Essentially, a tight integration of interval constraint propagation with recent propositional SAT-solving techniques.

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]

Extension to Probabilistic Hybrid Systems

Quantifying the probability of misbehavior

Example: The QMC Pause Dilemma

t := 0; cookies := 0; toilet := false; chats := 0

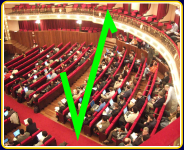
Wandering around

Example: The QMC Pause Dilemma

t := 0; cookies := 0; toilet := false; chats := 0

Wandering around

t <= 15 &
cookies >= 7
toilet
chats >= 2

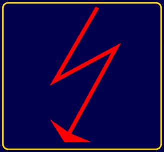


Example: The QMC Pause Dilemma

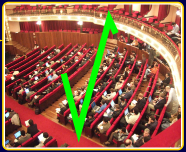
t := 0; cookies := 0; toilet := false; chats := 0

Wandering around

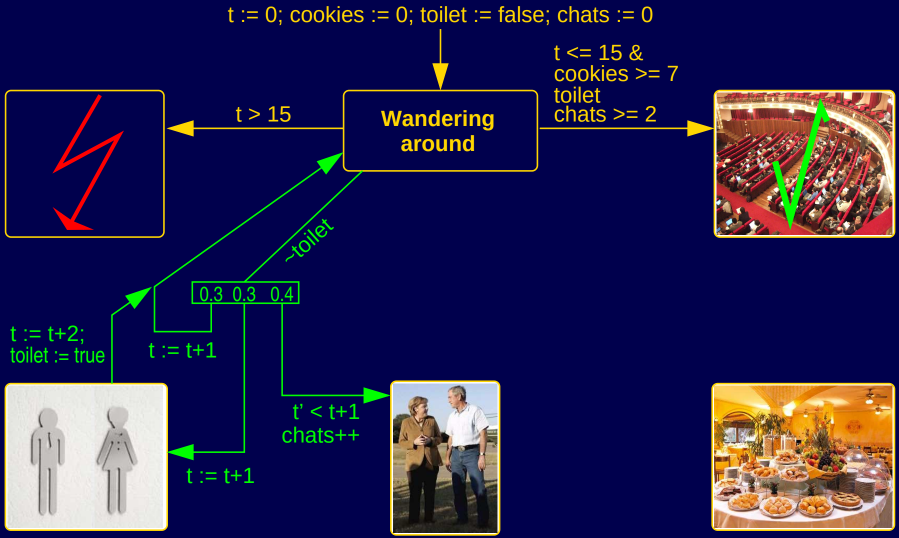
t > 15



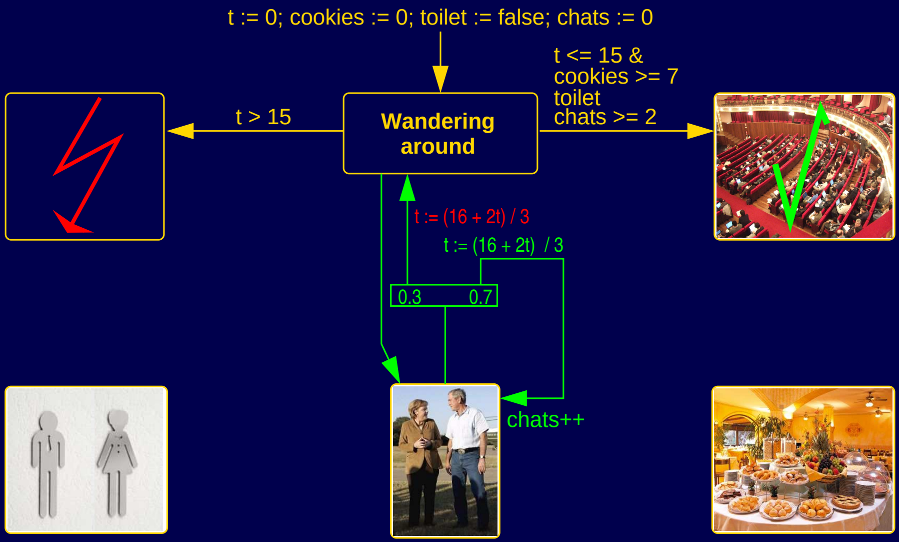
t <= 15 &
cookies >= 7
toilet
chats >= 2



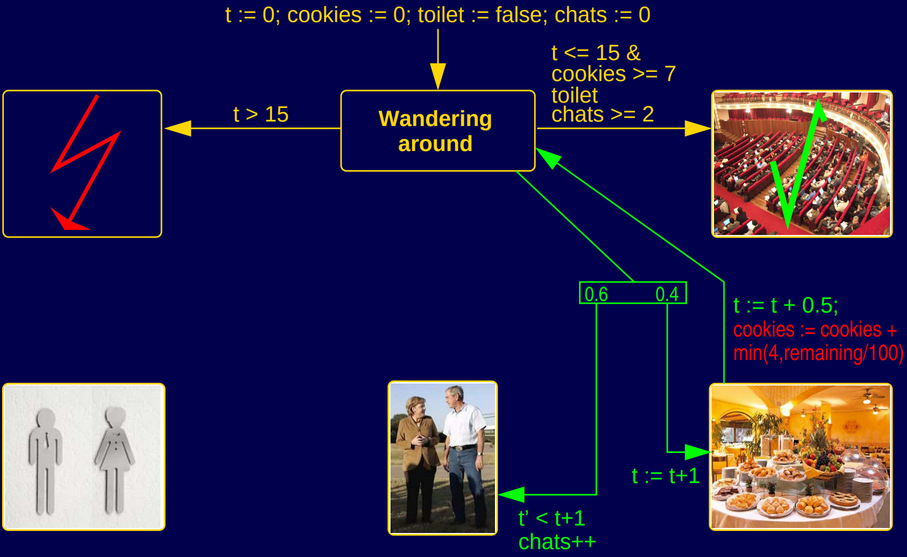
Example: The QMC Pause Dilemma



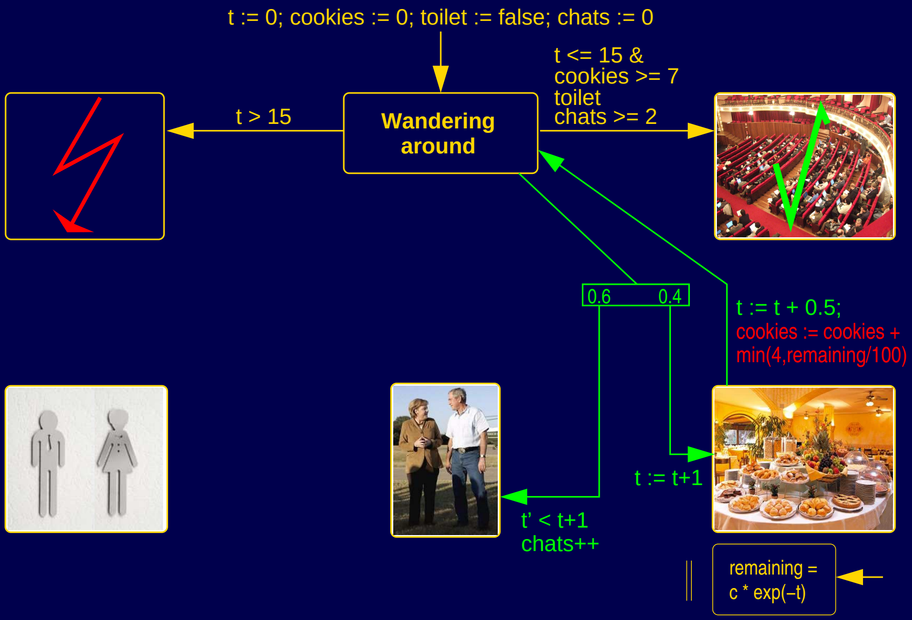
Example: The QMC Pause Dilemma



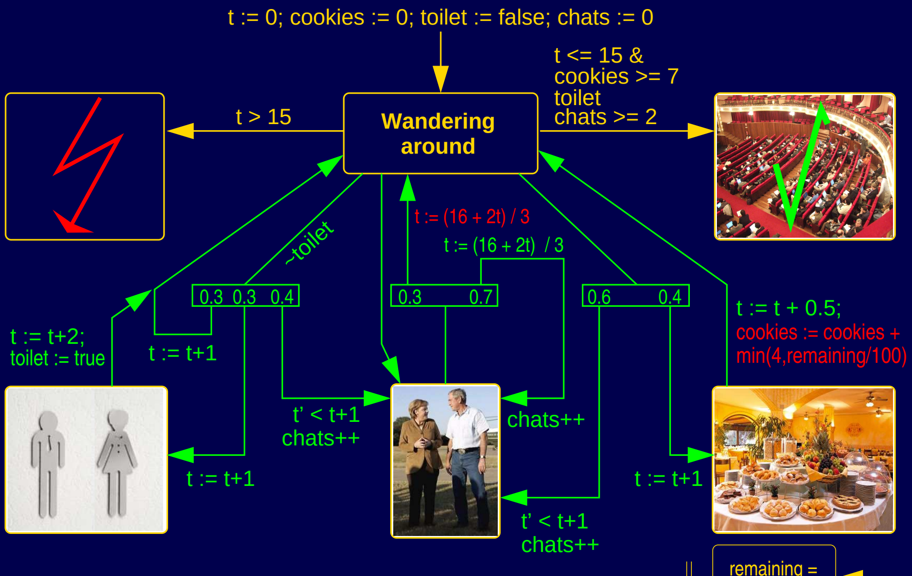
Example: The QMC Pause Dilemma



Example: The QMC Pause Dilemma

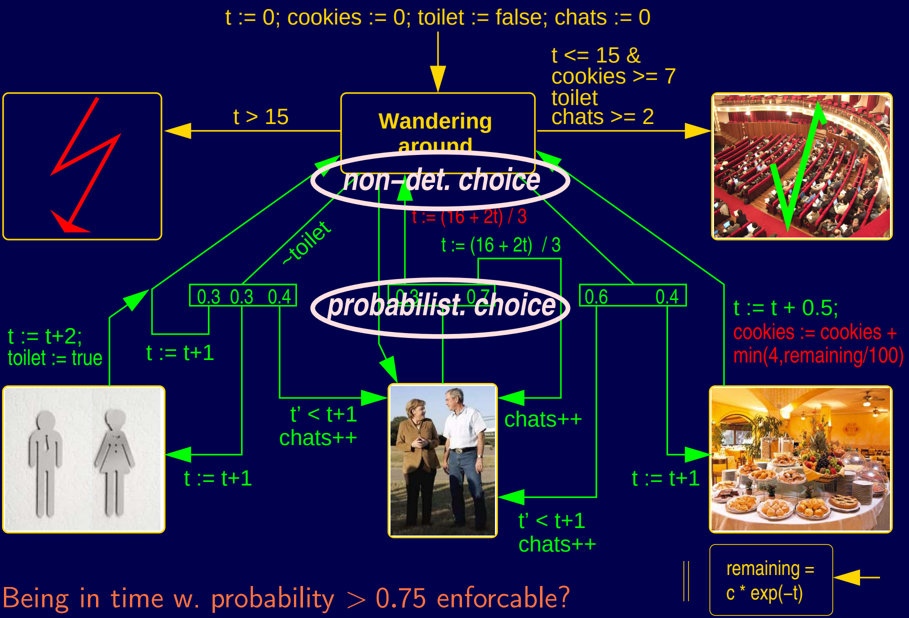


Example: The QMC Pause Dilemma



Being in time w. probability > 0.75 enforcable?

Example: The QMC Pause Dilemma



Being in time w. probability > 0.75 enforcable?

Constraint satisfaction

SAT

+ large Boolean formulae
− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics
− conjunctive systems only

Constraint satisfaction

SAT

+ large Boolean formulae
− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics
− conjunctive systems only

SMT

+ large Boolean combinations of
+ atoms from rich theories

Constraint satisfaction

SAT

+ large Boolean formulae
− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics
− conjunctive systems only

SMT

+ large Boolean combinations of
+ atoms from rich theories

BMC / stability proofs / ...
of hybrid systems

Constraint satisfaction

Stochastic constraint satisfaction

SAT

+ large Boolean formulae
− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics
− conjunctive systems only

SMT

+ large Boolean combinations of
+ atoms from rich theories

BMC / stability proofs / ...
of hybrid systems

Constraint satisfaction

Stochastic constraint satisfaction

SAT

+ large Boolean formulae
− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics
− conjunctive systems only

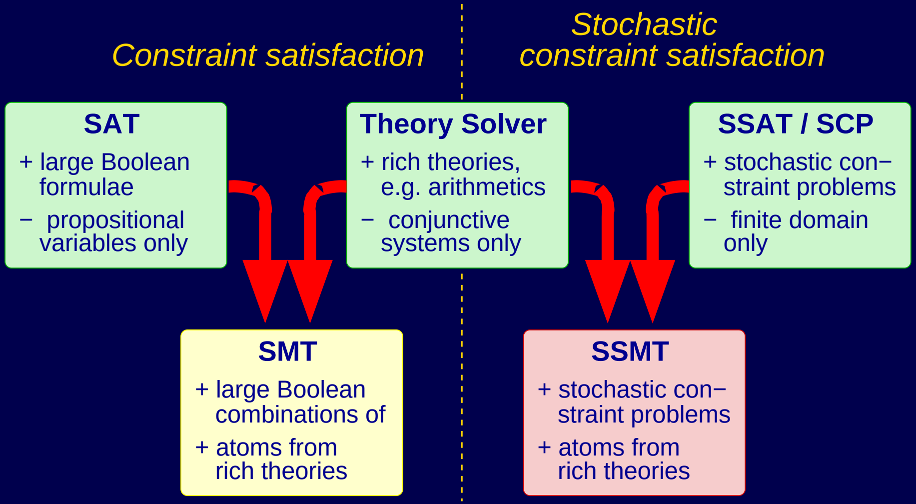
SSAT / SCP

+ stochastic con−straint problems
− finite domain only

SMT

+ large Boolean combinations of
+ atoms from rich theories

BMC / stability proofs / ...
of hybrid systems



BMC / stability proofs / . . .
of hybrid systems

Constraint satisfaction

Stochastic constraint satisfaction

SAT

+ large Boolean formulae

− propositional variables only

Theory Solver

+ rich theories, e.g. arithmetics

− conjunctive systems only

SSAT / SCP

+ stochastic con− straint problems

− finite domain only

SMT

+ large Boolean combinations of

+ atoms from rich theories

SSMT

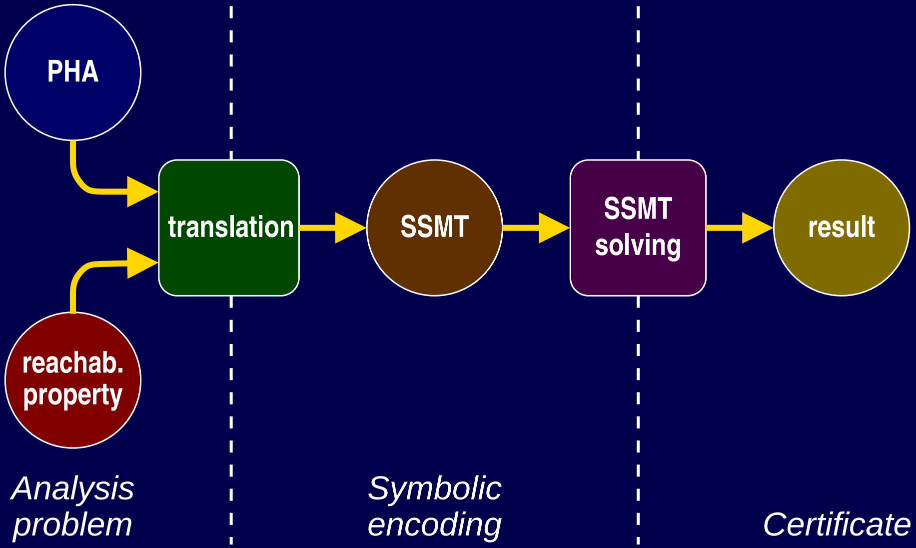
+ stochastic con− straint problems

+ atoms from rich theories

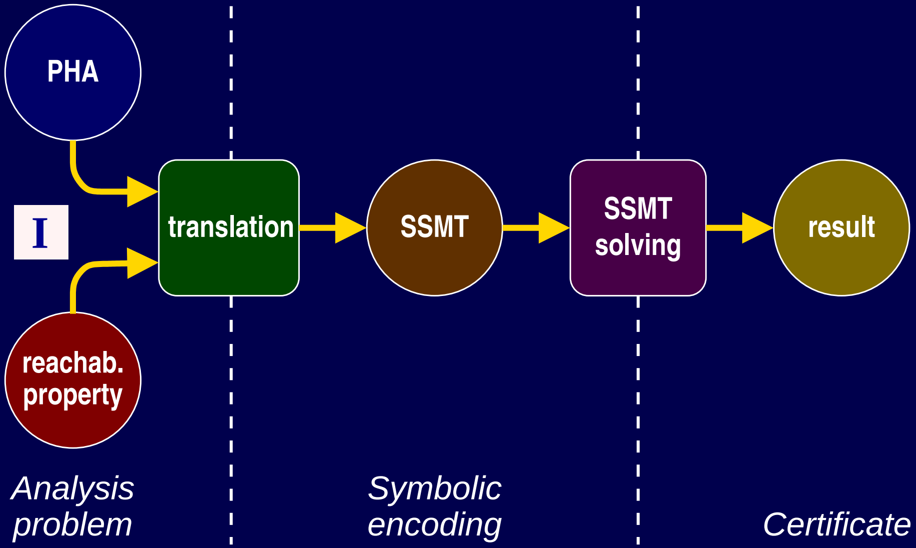
BMC / stability proofs / . . .
of hybrid systems

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

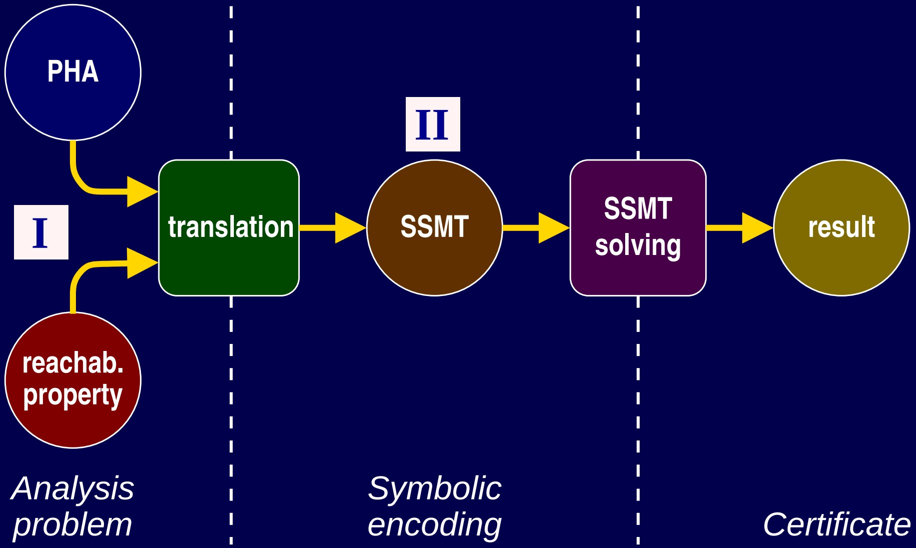
Approach


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

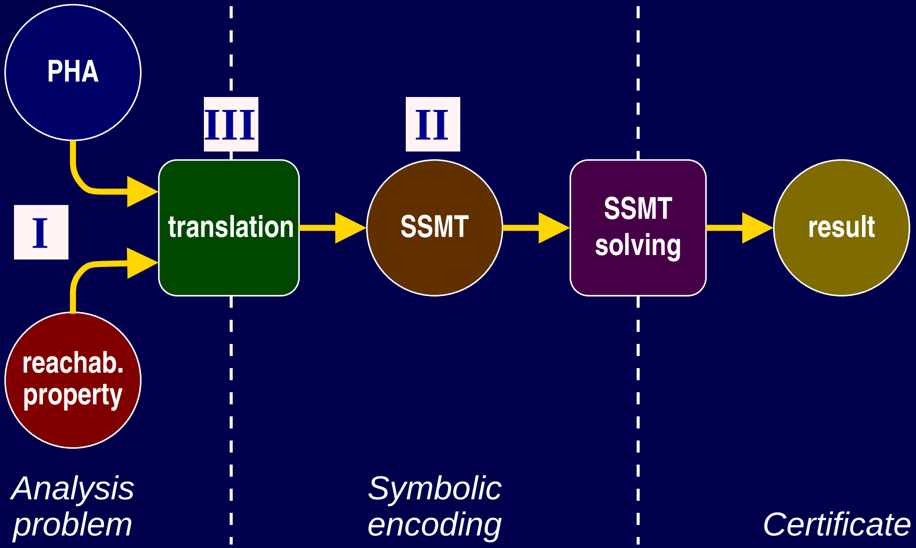
Approach


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

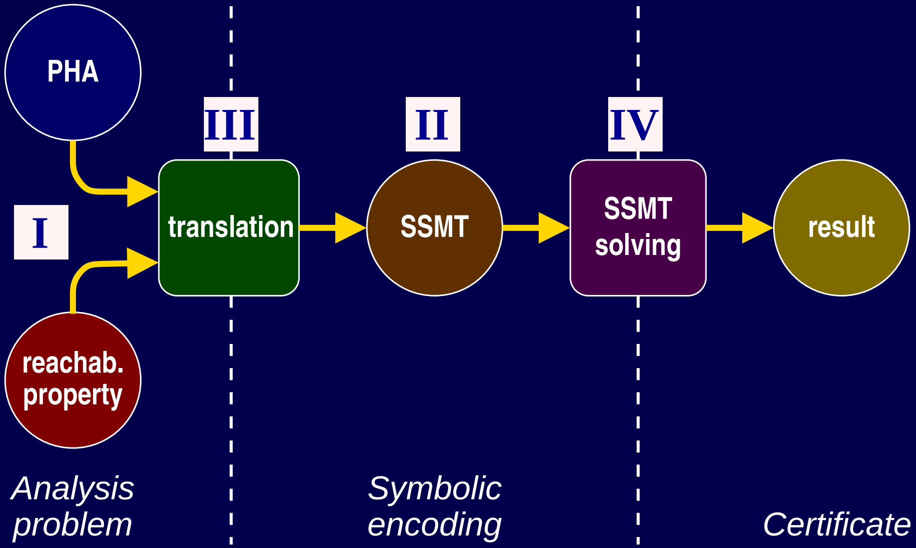
Approach


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

Approach

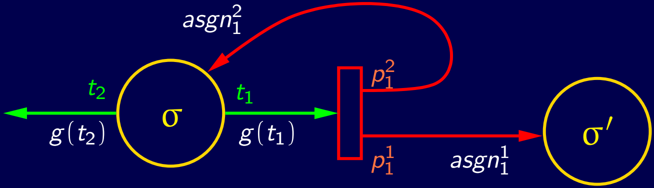

PHA
III
II
I
translation
SSMT
SSMT solving
result
reachab. property
Analysis problem
Symbolic encoding
Certificate

Approach


PHA
III
II
IV
I
translation
SSMT
SSMT solving
result
reachab. property
Analysis problem
Symbolic encoding
Certificate

Probabilistic Bounded Reachability in Probabilistic Hybrid Automata

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 $\max_{(\sigma,\mathbf{x})\text{ an initial state}} \mathbf{P}^k_{(\sigma,\mathbf{x})} \leq tolerable$?

Probabilistic Bounded Reachability

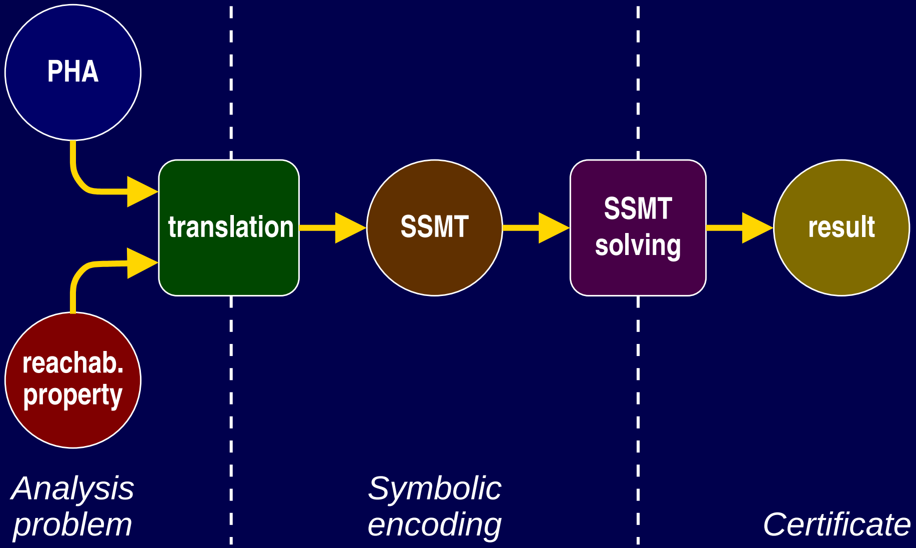
Given:

- 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).

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.

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

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.

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

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 $\mathrm{dom}(x)$.

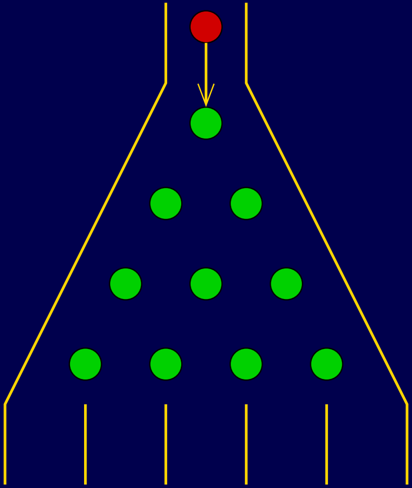
2) **randomized** $Ꙟ_{\langle(v_1,p_1),\ldots,(v_m,p_m)\rangle}\, y \in \mathrm{dom}(y)$

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

   p_i is probability of setting y to value v_i.

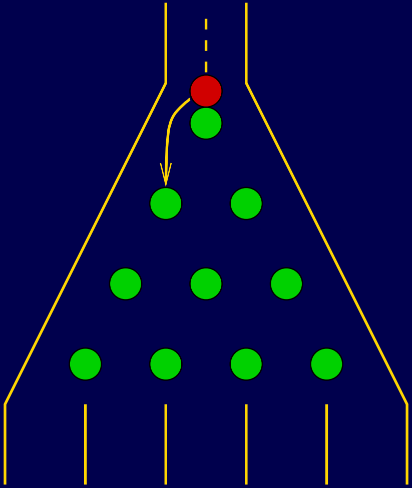
Randomized Quantification

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



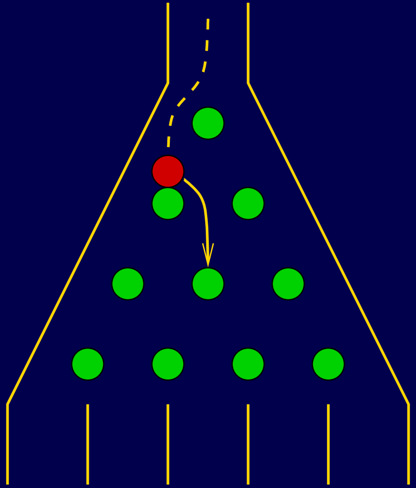
Randomized Quantification

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



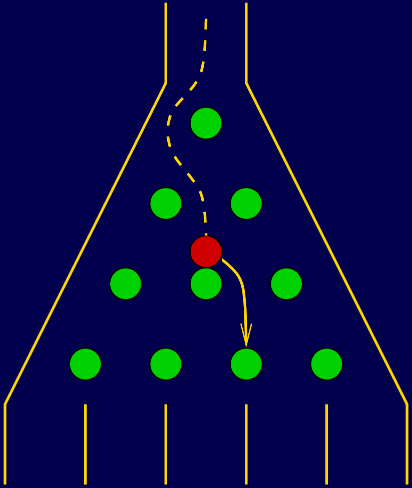
Randomized Quantification

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



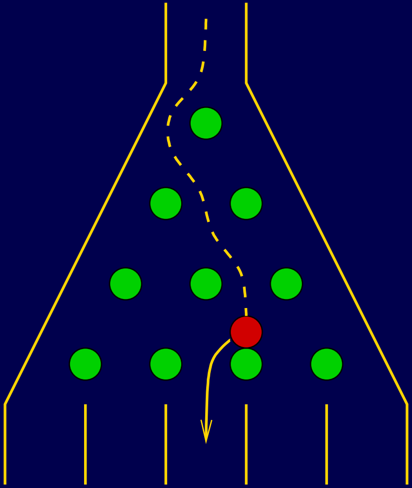
Randomized Quantification

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



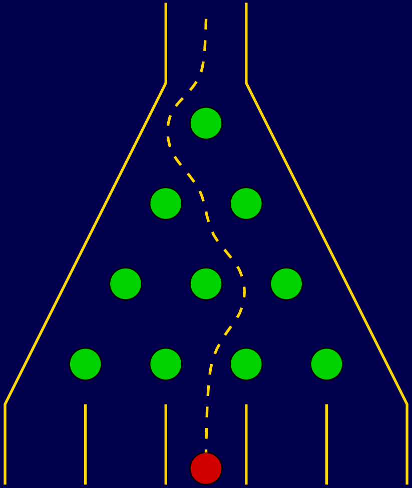
Randomized Quantification

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



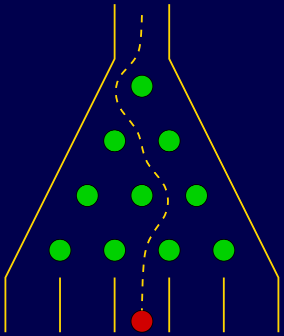
Randomized Quantification

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



Randomized Quantification

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

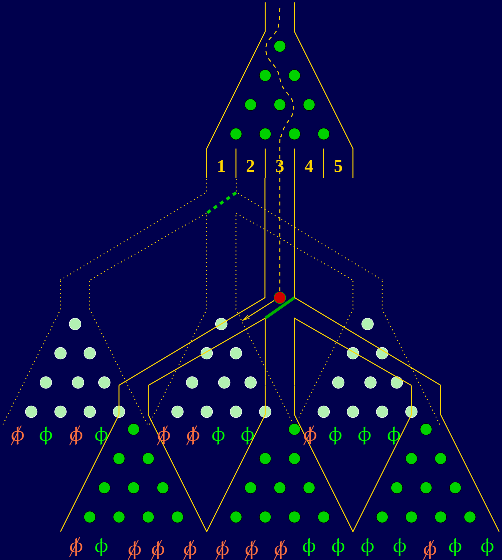


$$k = \quad \mathbf{0} \quad \mathbf{1} \quad \mathbf{2} \quad \mathbf{3} \quad \mathbf{4}$$

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

$$\rotatebox{180}{R}_{\langle(0,p_0),(1,p_1),(2,p_2),(3,p_3),(4,p_4)\rangle} prob_1 \in \{0, 1, 2, 3, 4\}$$

Stochastic satisfiability modulo theory (SSMT)



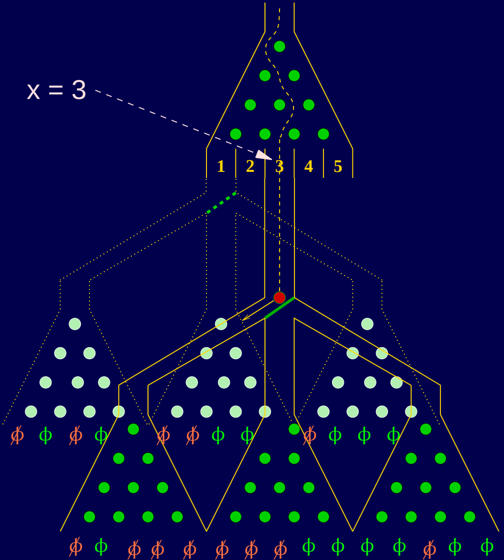
$\reflectbox{R}_{d_1} x \in \{0, 1, 2, 3, 4\}$

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

$\reflectbox{R}_{d_2} z \in \{0, 1, 2, 3, 4\}:$

ϕ

Stochastic satisfiability modulo theory (SSMT)



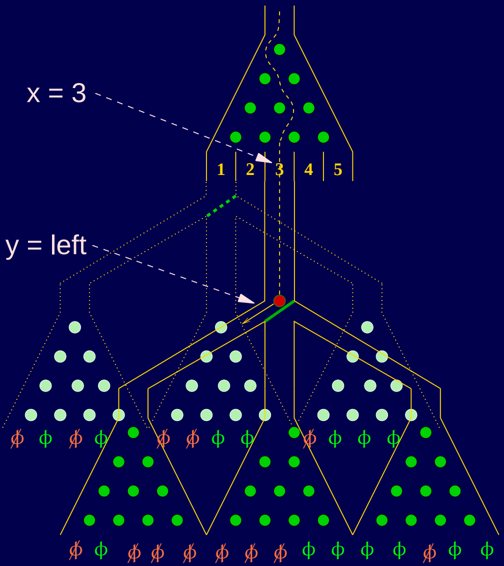
$\reflectbox{R}_{d_1} x \in \{0, 1, 2, 3, 4\}$

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

$\reflectbox{R}_{d_2} z \in \{0, 1, 2, 3, 4\}:$

ϕ

Stochastic satisfiability modulo theory (SSMT)



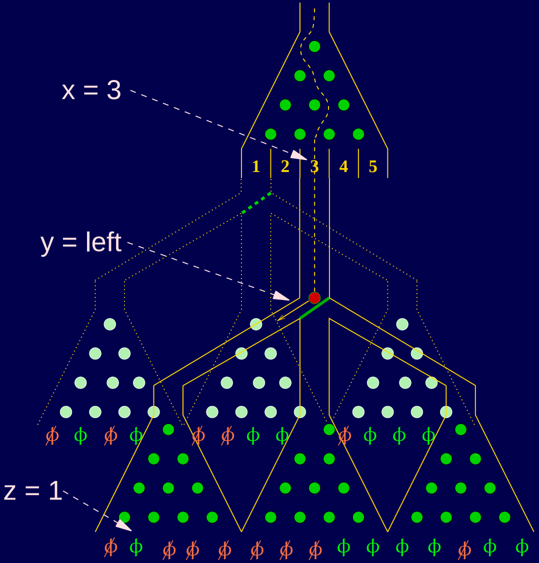
$\text{Я}_{d_1} x \in \{0, 1, 2, 3, 4\}$

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

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

ϕ

Stochastic satisfiability modulo theory (SSMT)



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

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

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

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:

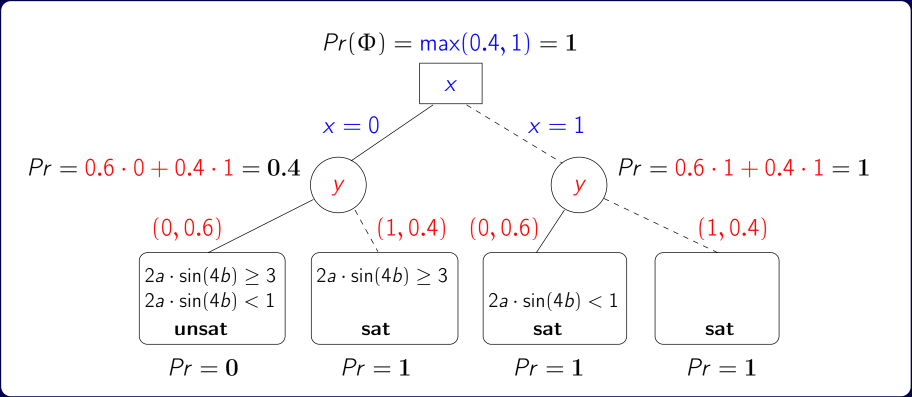
3. $Pr(\exists x \in \mathcal{D}\ \mathcal{Q} : \varphi) = \max_{v \in \mathcal{D}} Pr(\mathcal{Q} : \varphi[v/x])$.

$\text{ꓤ} \triangleq$ Weighted sum of all alternatives:

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

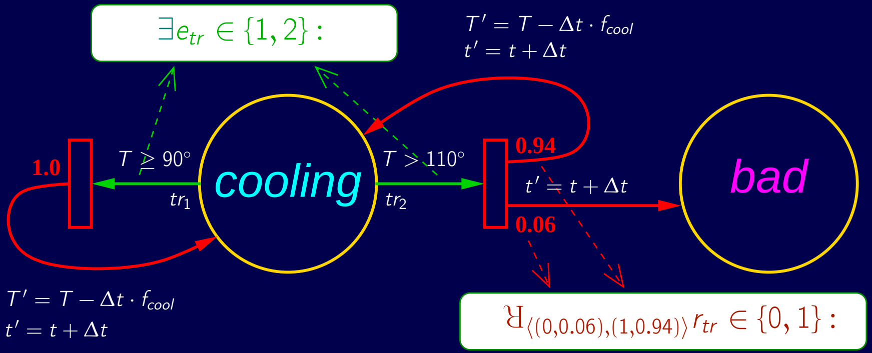
Semantics of an SSMT formula: Example

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

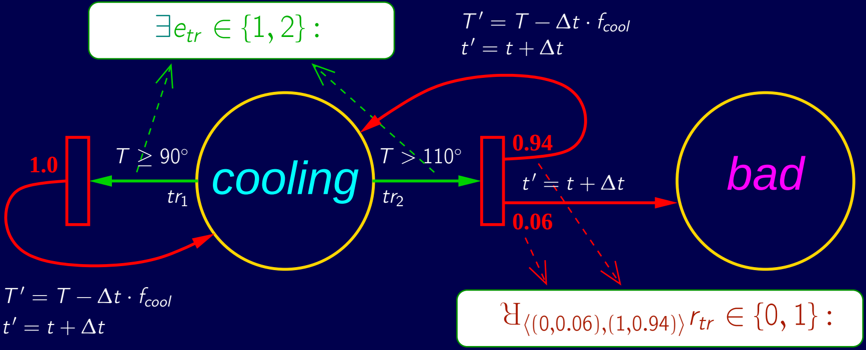


Translating PHA Problems to SSMT Problems

Translating PHA into SSMT



Translating PHA into SSMT



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

Unwinding

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

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

Unwinding

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

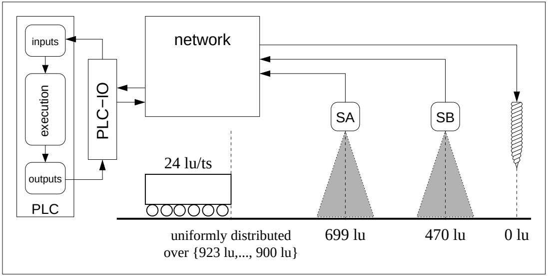
- Alternating quantifier prefix encodes alternation of
  - nondeterministic transition selection
  - probabilistic choice between transition variants
- $Pr(\Phi)$ = accumulated probability over all paths of reaching bad state under malicious adversary within k steps $= \max_{(\sigma,\mathbf{x})\text{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

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

A Case Study

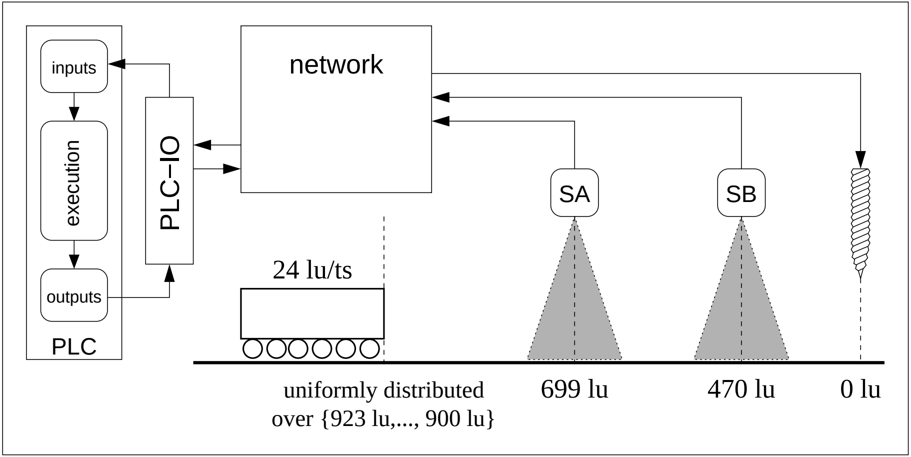
Networked automation systems

Case study: Networked automation systems

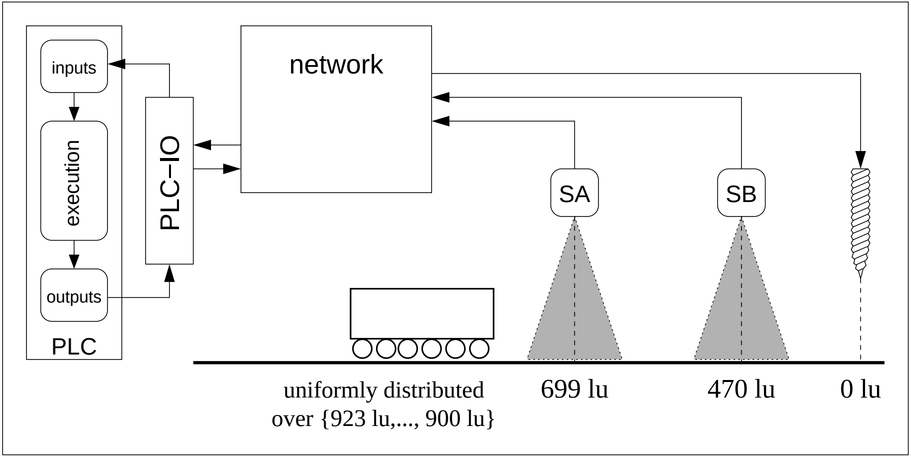


- Networked automation system (NAS) [Greifeneder, Frey 2006]
- typical NAS consists of
  - programmable logic controllers (PLCs),
  - several sensors and actuators,
  - wired or wireless communication networks,
  - various input-output devices

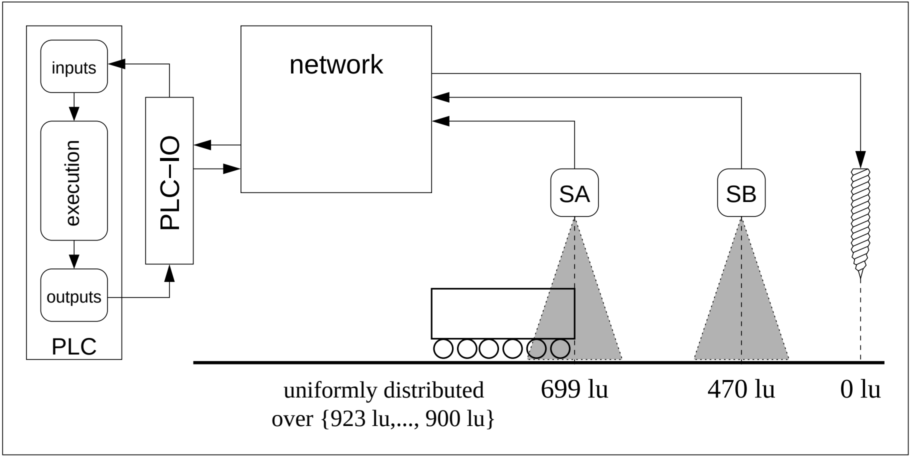
Case study: Networked automation systems



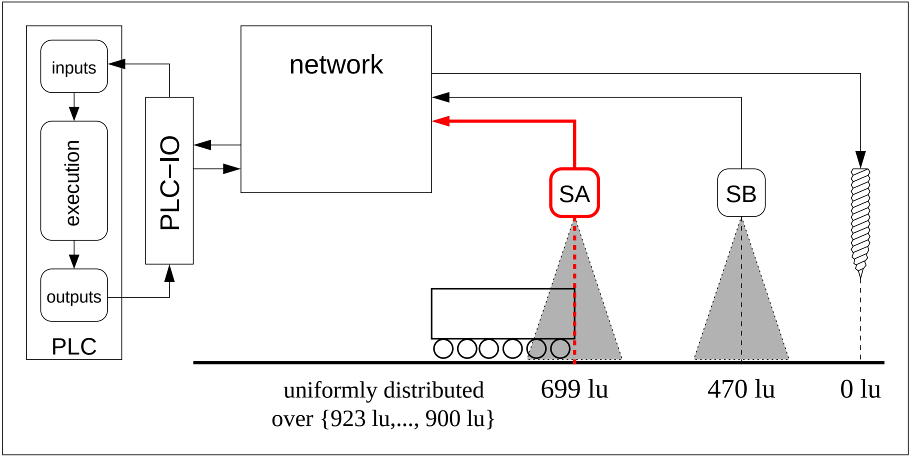
Case study: Networked automation systems



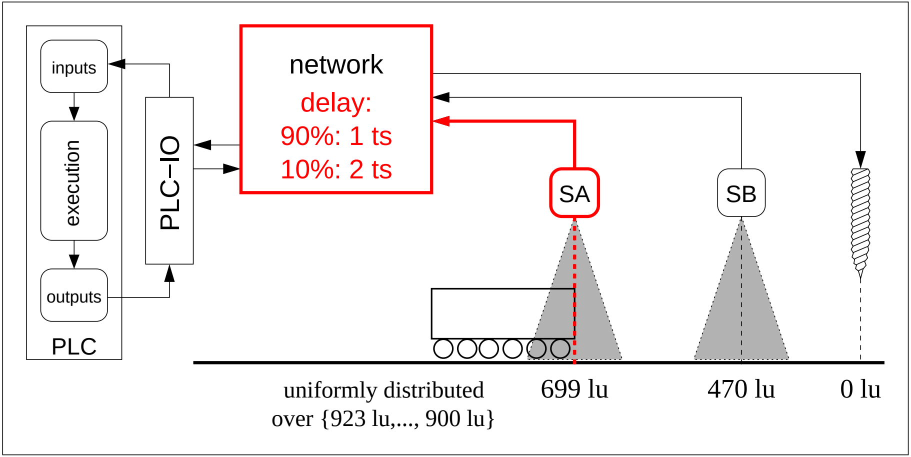
Case study: Networked automation systems



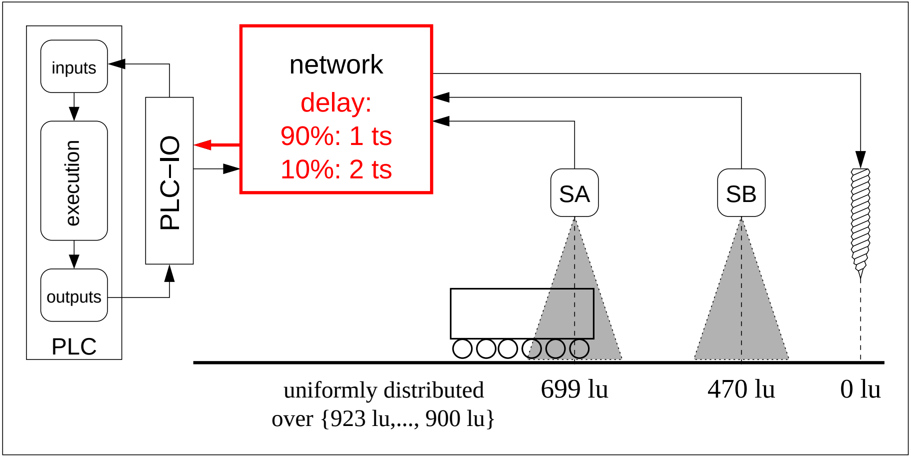
Case study: Networked automation systems



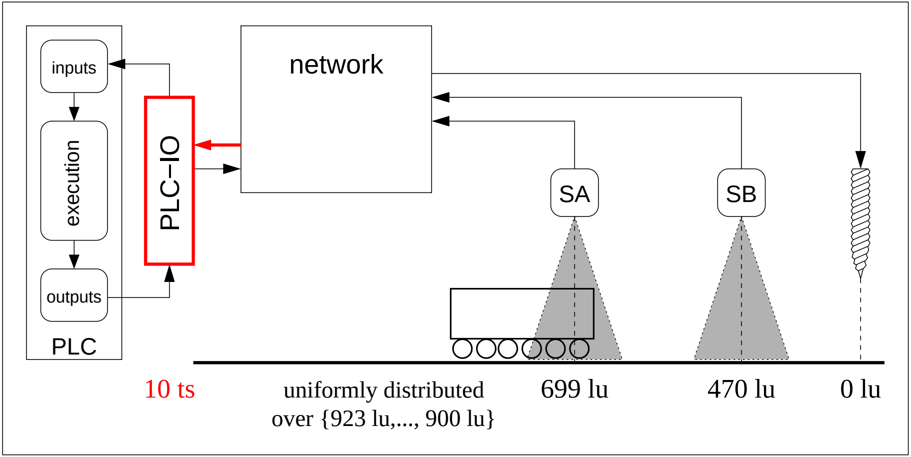
Case study: Networked automation systems



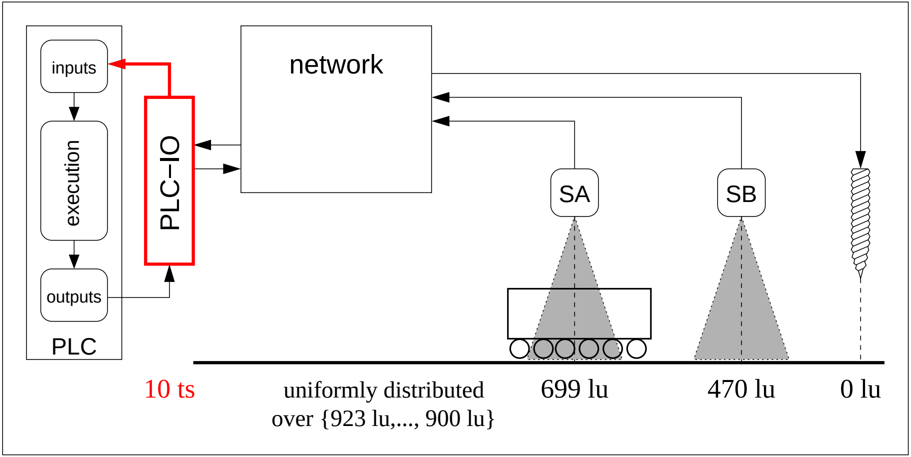
Case study: Networked automation systems



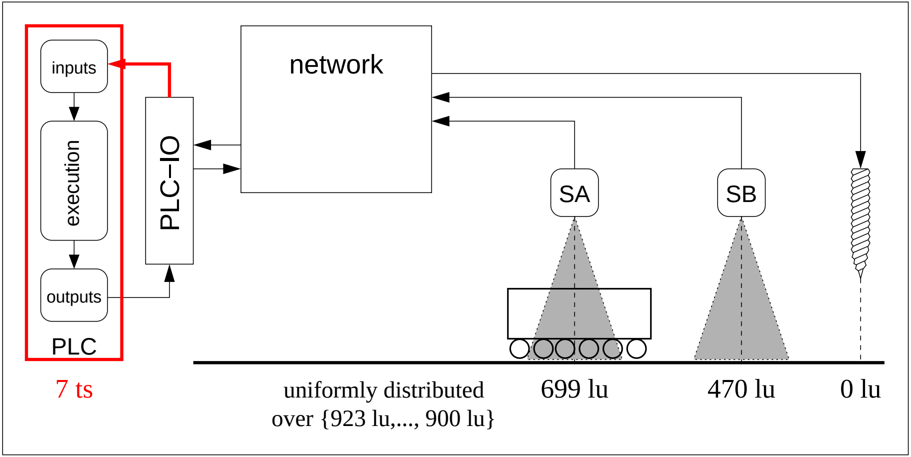
Case study: Networked automation systems



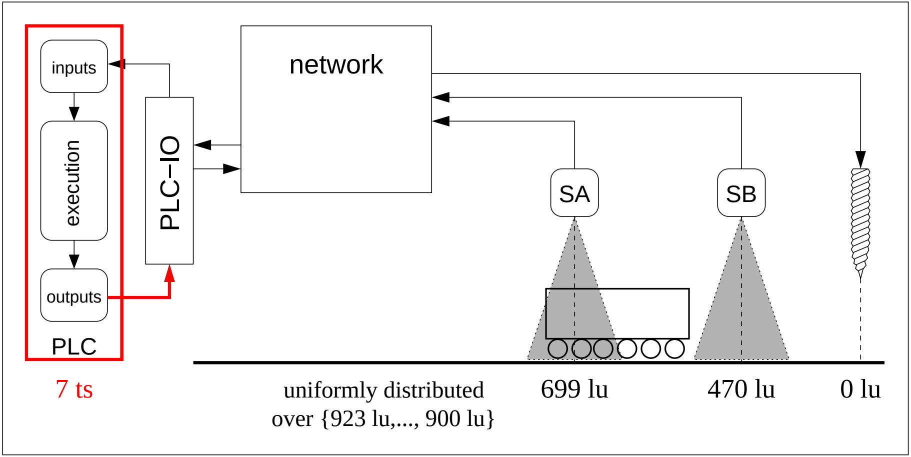
Case study: Networked automation systems



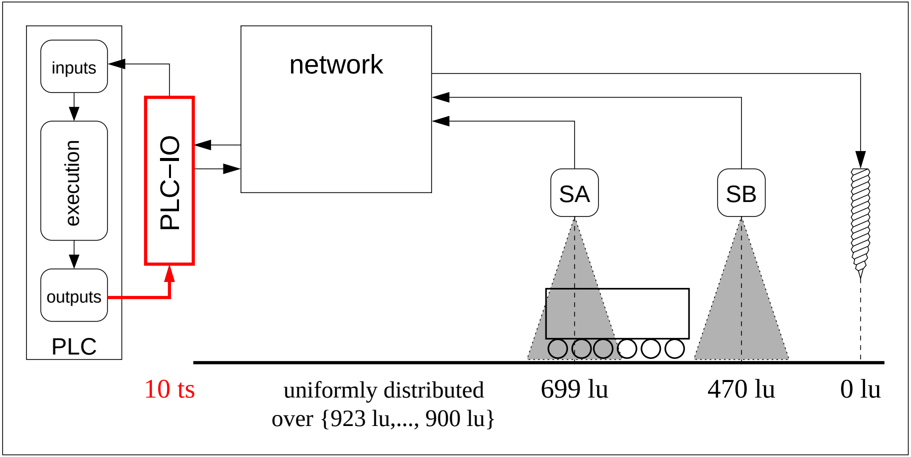
Case study: Networked automation systems



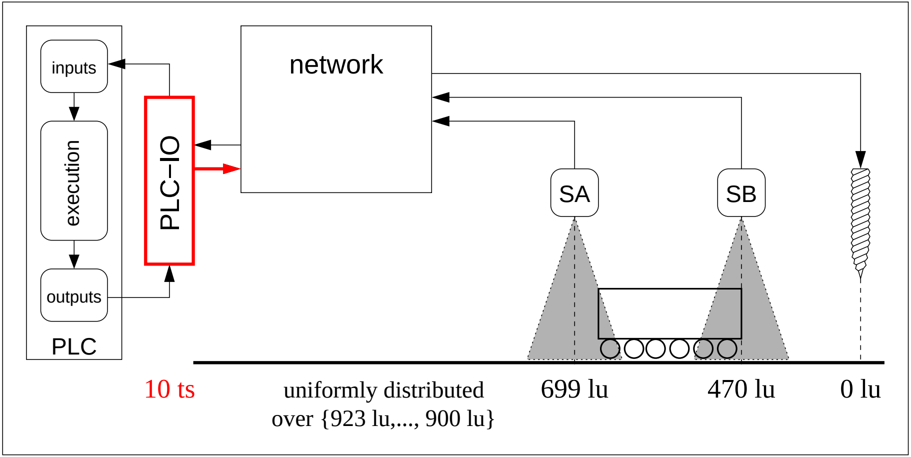
Case study: Networked automation systems



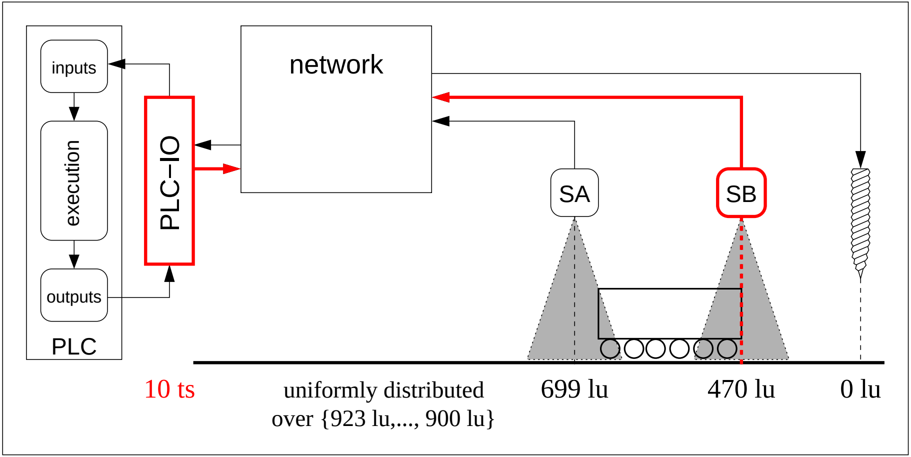
Case study: Networked automation systems



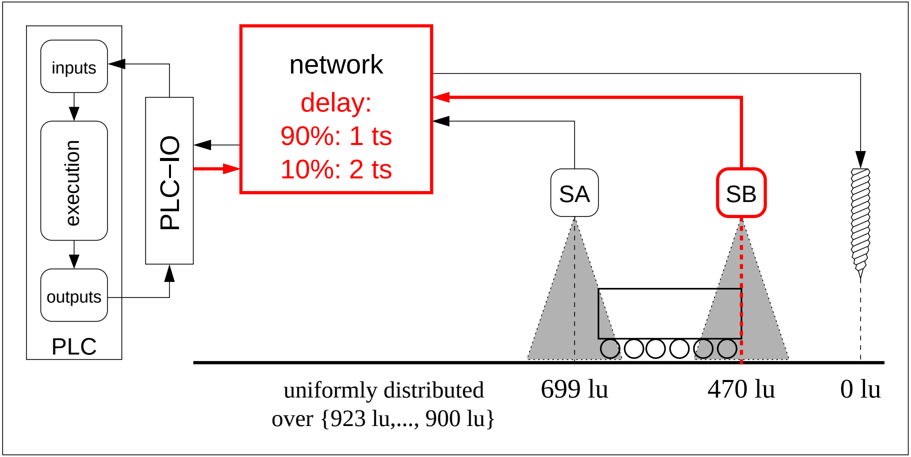
Case study: Networked automation systems



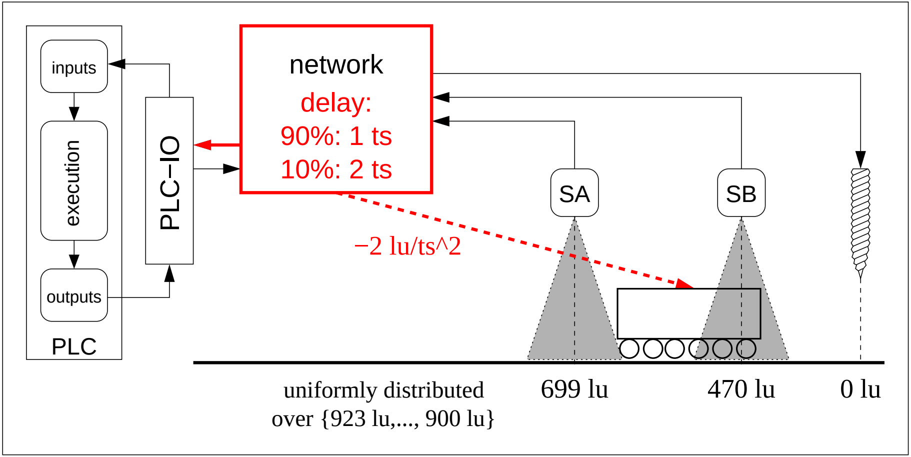
Case study: Networked automation systems



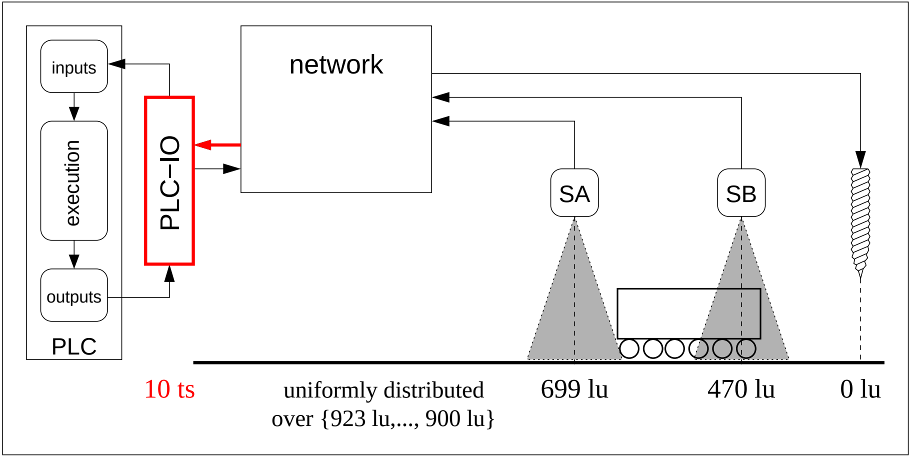
Case study: Networked automation systems



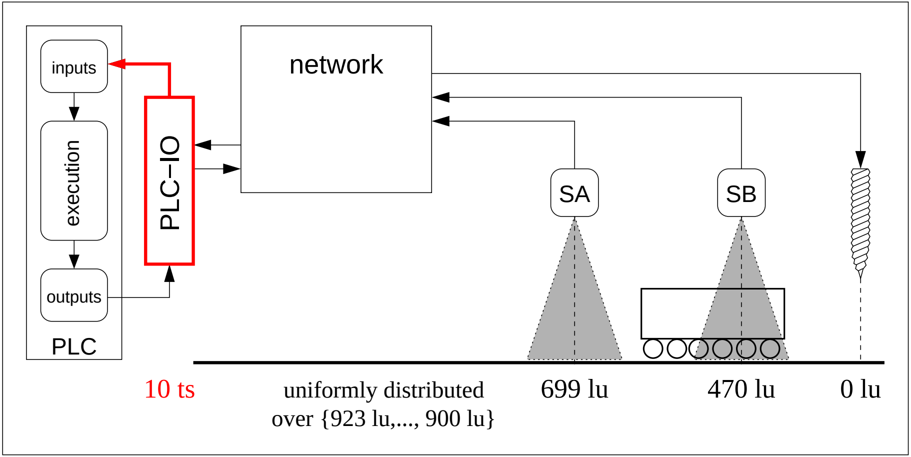
Case study: Networked automation systems



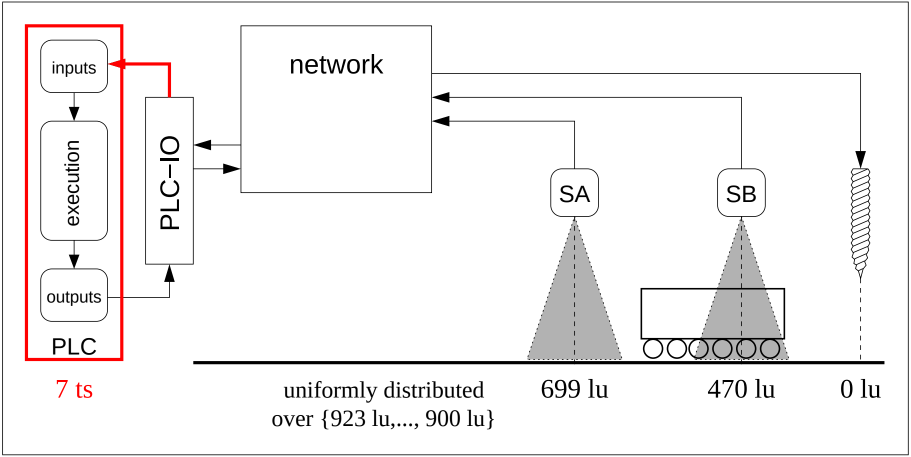
Case study: Networked automation systems



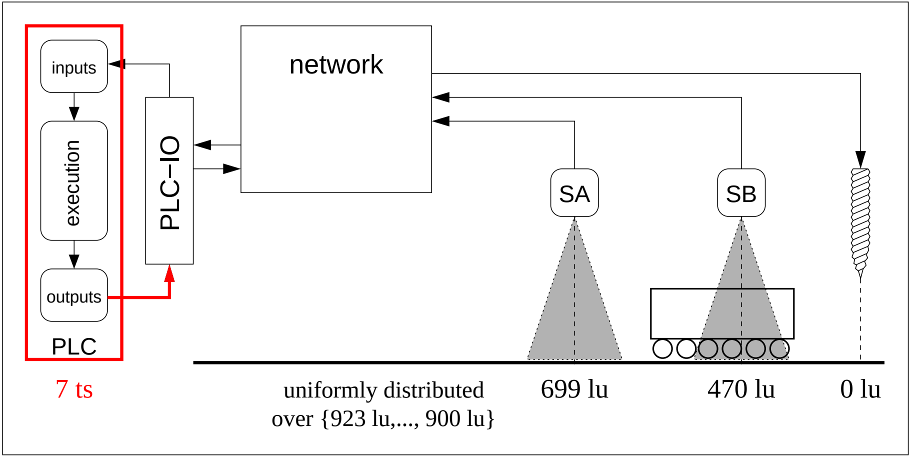
Case study: Networked automation systems



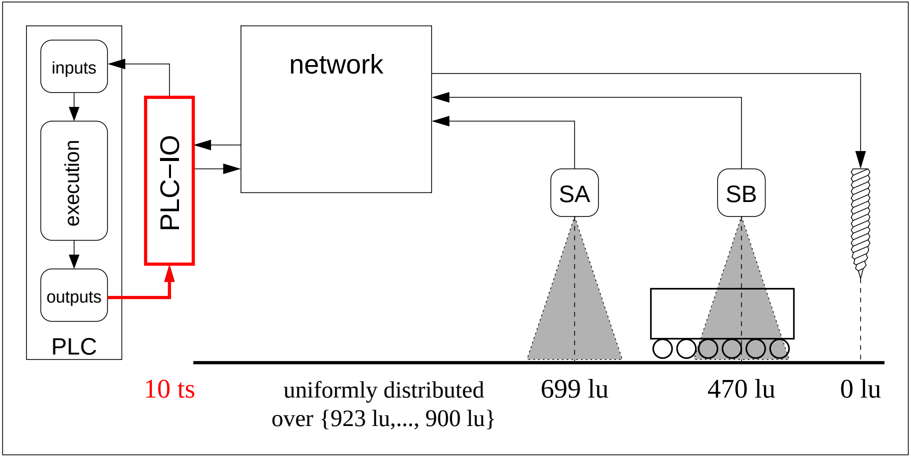
Case study: Networked automation systems



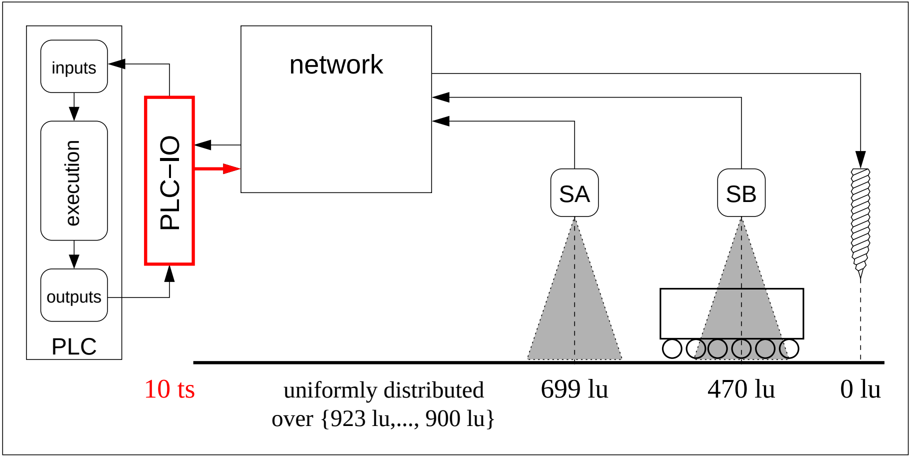
Case study: Networked automation systems



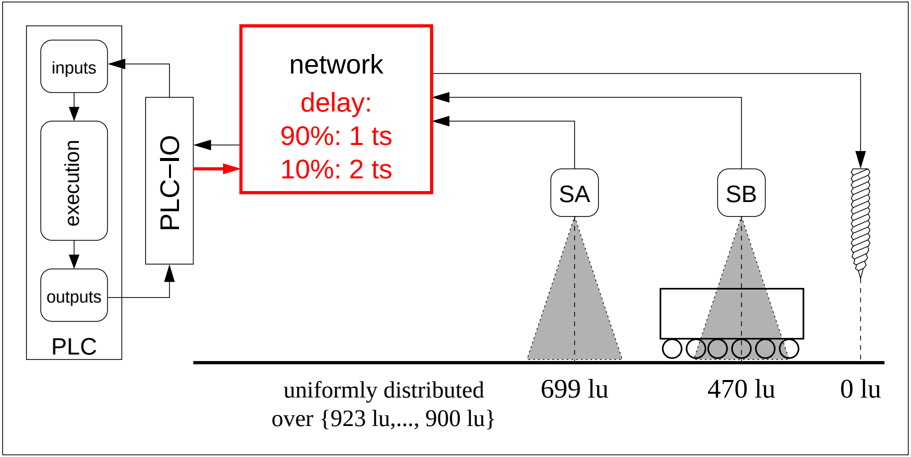
Case study: Networked automation systems



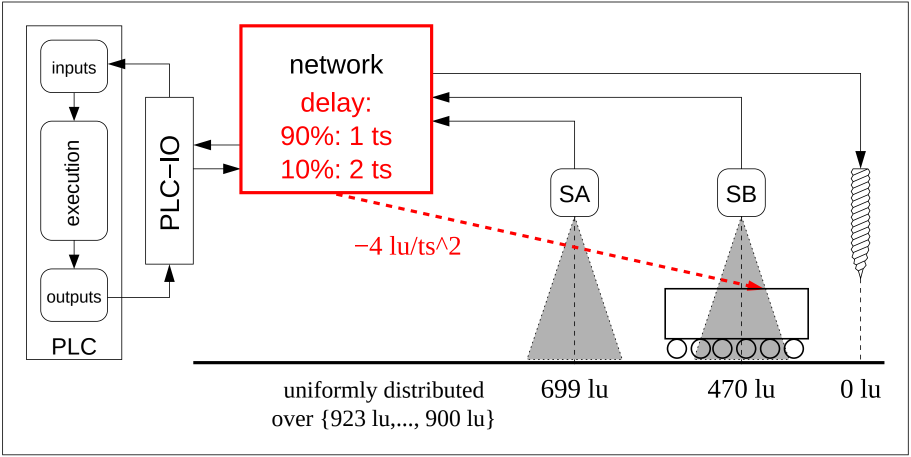
Case study: Networked automation systems



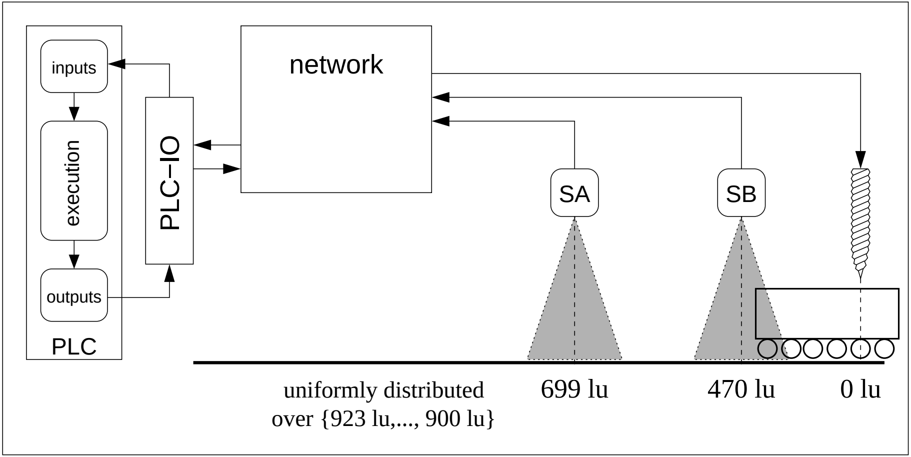
Case study: Networked automation systems



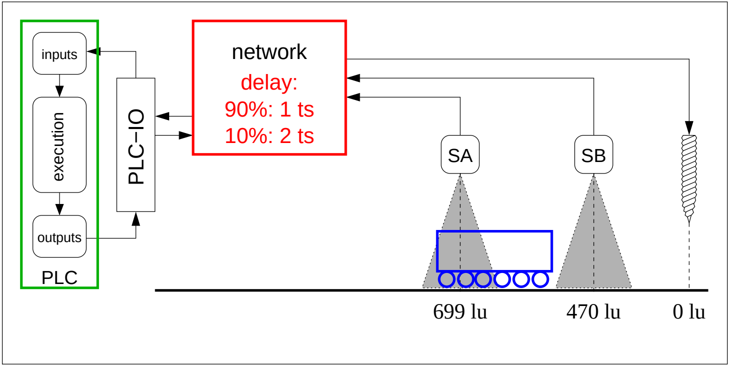
Case study: Networked automation systems



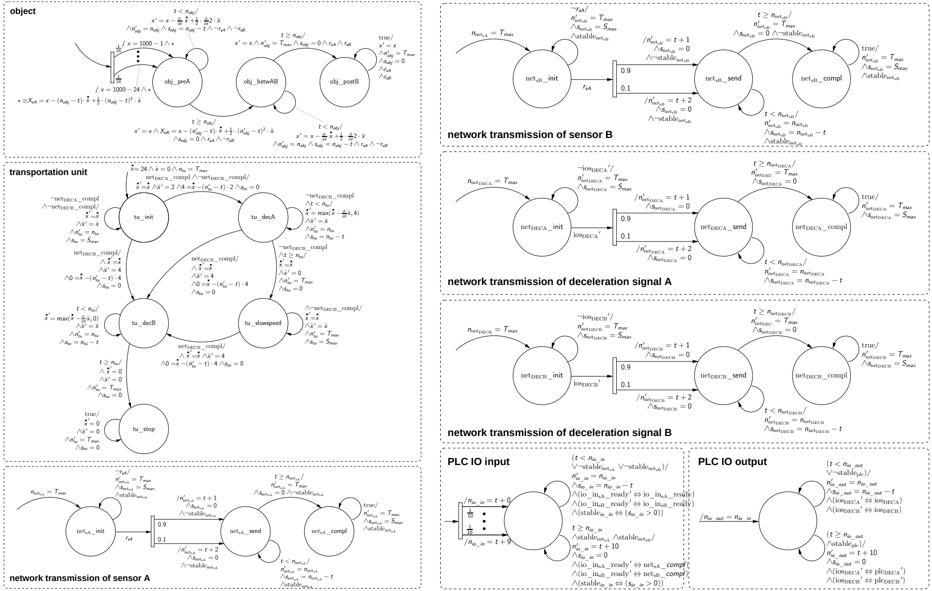
Case study: Networked automation systems

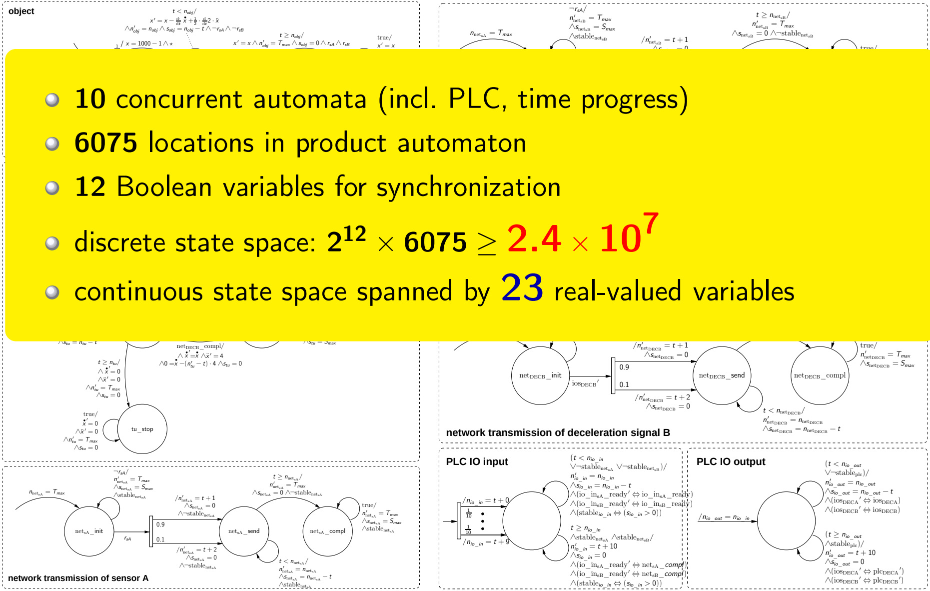


Case study: Discrete-time system model

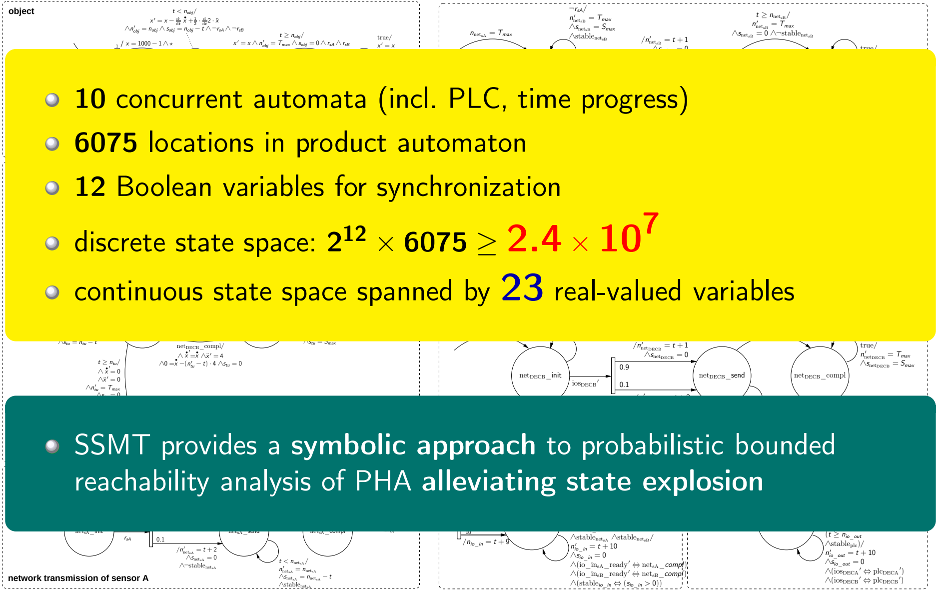


- continuous **dynamics** of conveyor: $\frac{ds}{dt} = v$, $\frac{dv}{dt} = a$
  $\leadsto s' = s + v \cdot \Delta t + \frac{1}{2} \cdot a \cdot \Delta t^2,\ v' = v + a \cdot \Delta t$
- discrete **computations** updating deceleration a, passing messages,...
- **discrete** probabilistic **choices**: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


object
obj_preA
obj_betwAB
obj_postB
transportation unit
tu_init
tu_decA
tu_decB
tu_slowspeed
tu_stop
network transmission of sensor A
network transmission of sensor B
network transmission of deceleration signal A
network transmission of deceleration signal B
PLC IO input
PLC IO output



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



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

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

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.

SSMT algorithm

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

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

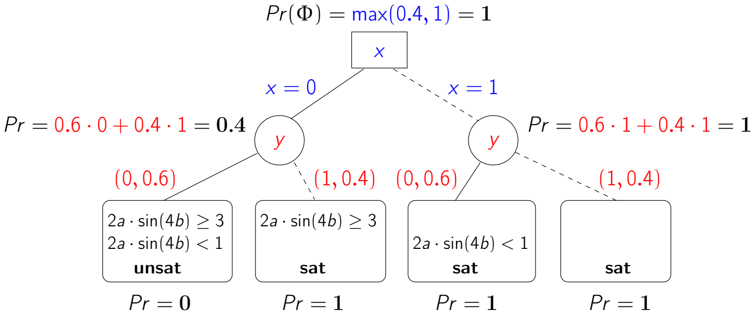
Solution: Take appropriate SMT solver, implant branching rules for quantifiers, add rigorous proof-tree pruning:

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

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



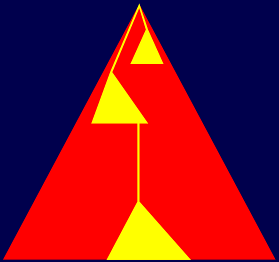
SSMT algorithm: Pruning rules

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:

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

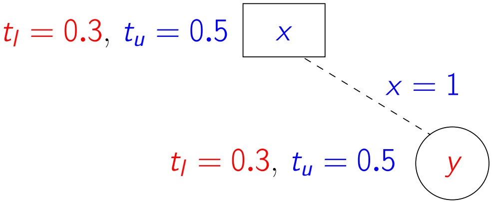
Efficient quantifier handling: Thresholding

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

$t_l = 0.3,\ t_u = 0.5$ $\boxed{x}$

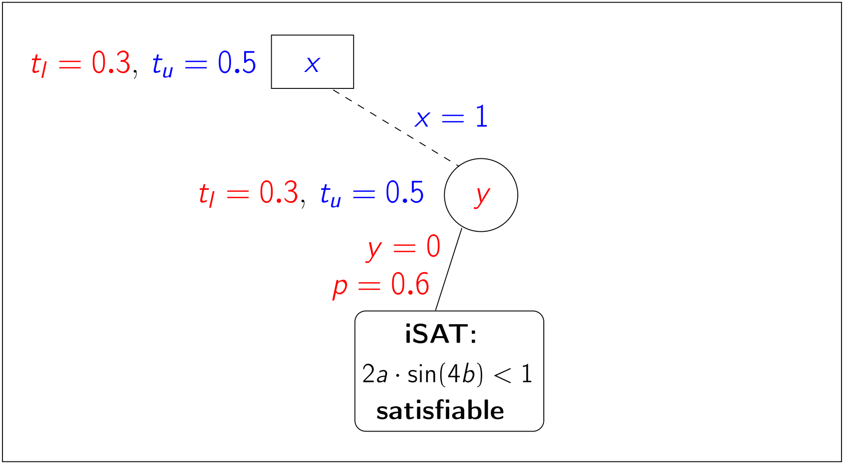
Efficient quantifier handling: Thresholding

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



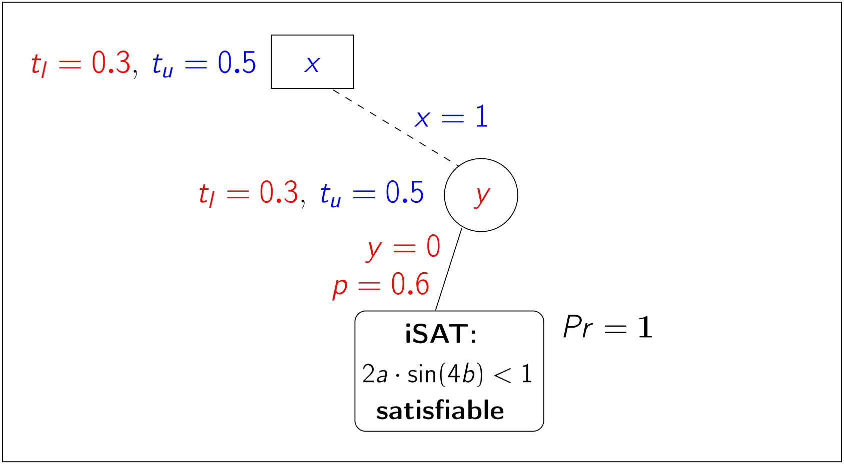
Efficient quantifier handling: Thresholding

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



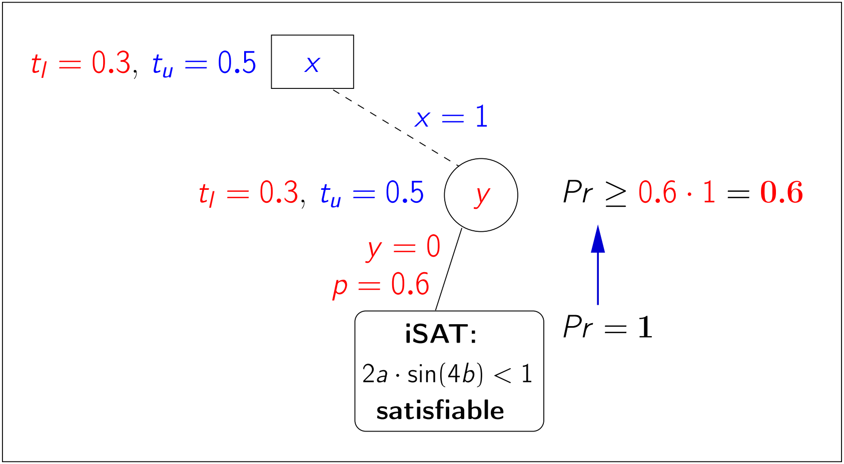
Efficient quantifier handling: Thresholding

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



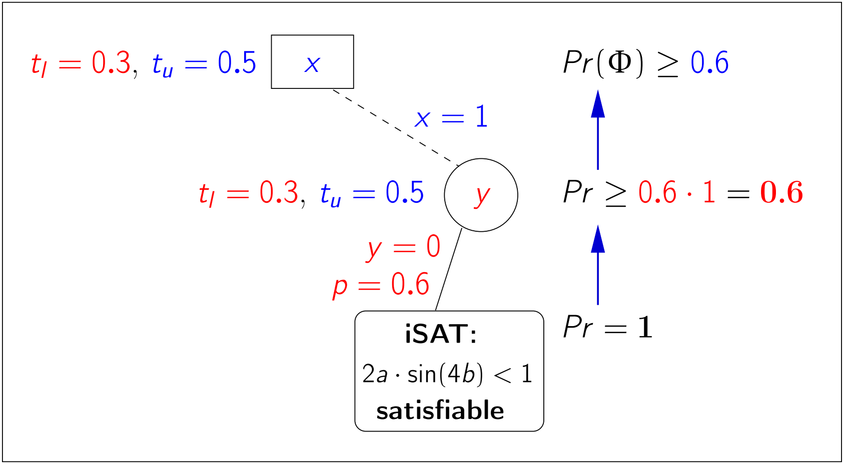
Efficient quantifier handling: Thresholding

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



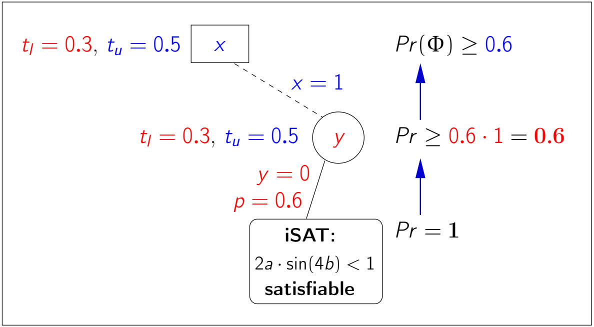
Efficient quantifier handling: Thresholding

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



Efficient quantifier handling: Thresholding

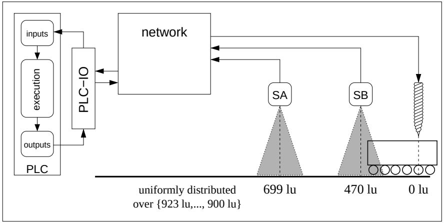
$$\Phi = \exists x \in \{0,1\}\ \text{Я}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} :$$
$$(x > 0 \vee 2a \cdot \sin(4b) \geq 3) \wedge (y > 0 \vee 2a \cdot \sin(4b) < 1)$$



Pruning occurs

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

Case study: Analysis

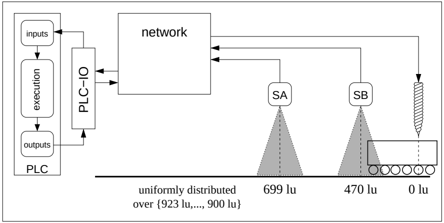


Goal: Determine whether probab. of stopping close to drilling pos. sufficient

1.
- 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 $k = 44$, runtime 134 min (with thresholding)

Case study: Analysis



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

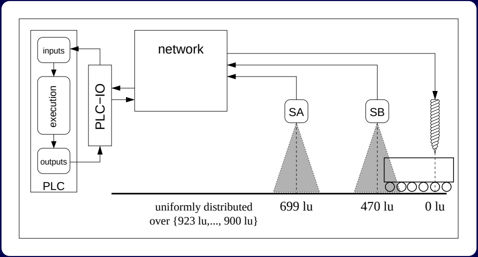
1
- find BMC unwinding depth k s.t. object has stopped
- i.e., find k s.t. $Pr(PBMC(k)) = \mathbf{1}$ with $TARGET(\mathbf{x}) := tu\_stop$

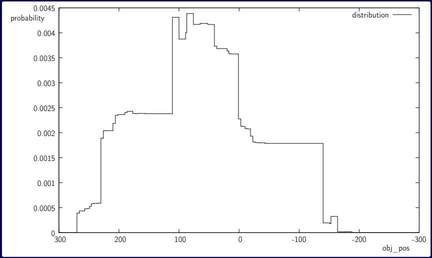
$\rightsquigarrow$ holds for $k = \mathbf{44}$, runtime 134 min (with thresholding)

2

$TARGET(\mathbf{x})$	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**

Case study: Analysis





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

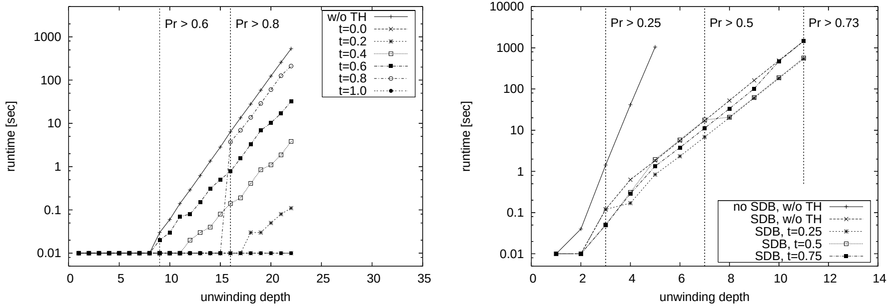
1.
   - find BMC unwinding depth k s.t. object has stopped
   - i.e., find k s.t. $Pr(PBMC(k)) = \mathbf{1}$ with $TARGET(\mathbf{x}) := tu\_stop$

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

2.

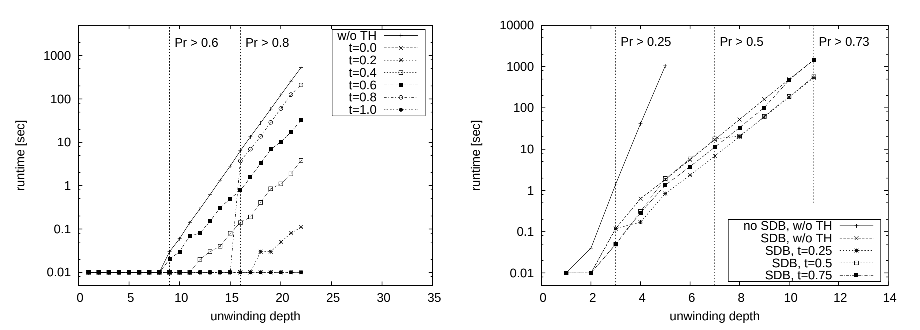
$TARGET(\mathbf{x})$	probability	runtime
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$=$ **0.397345[16,29]**	**71 min**
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$\geq$ **0.9**	**13 min**
$100 \geq obj\_pos \wedge obj\_pos \geq 0$	$\geq$ **0.95**	**11 min**

SSMT algorithm: Early experimental results



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

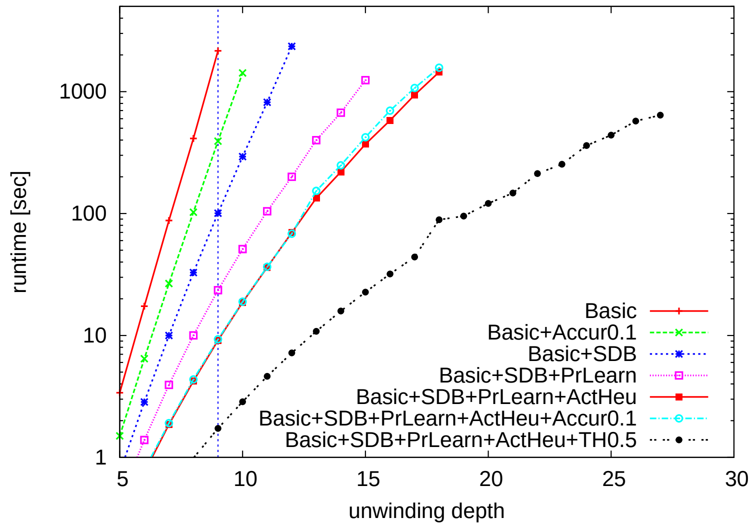
SSMT algorithm: Early experimental results



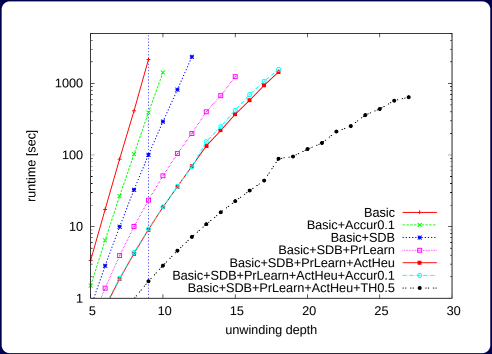
Impact of thresholding (left) and solution-directed backjumping (right)

SSMT often traverses only minor fraction of quantifier domains!

SSMT algorithm: Recent experimental results

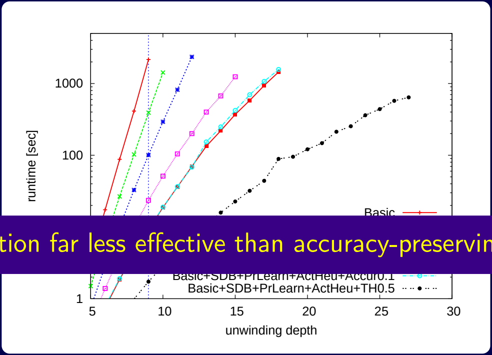


SSMT algorithm: Recent experimental results



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			

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			

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

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;

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

  within the
  - DFG-funded Transregional Research Center 14 "AVACS" (Automatic Analysis and Verification of Complex 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
  - MPII, Saarbrücken, Germany
  - Universität des Saarlandes, Saarbrücken, Germany
- and to Andreas Eggers, Christian Herde, and Tino Teige for contributing many slides.