Bounded model checking for hybrid systems

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen

SS09

Contents

Contents

Overview

- Bounded model checking
- Optimizations
  - Formula representation
  - Constraint sharing and replication

Bounded Model Checking (BMC)

Counterexamples of length k are described by a first-order logic formula over $(\mathbb{R}, +, <, 0, 1)$:

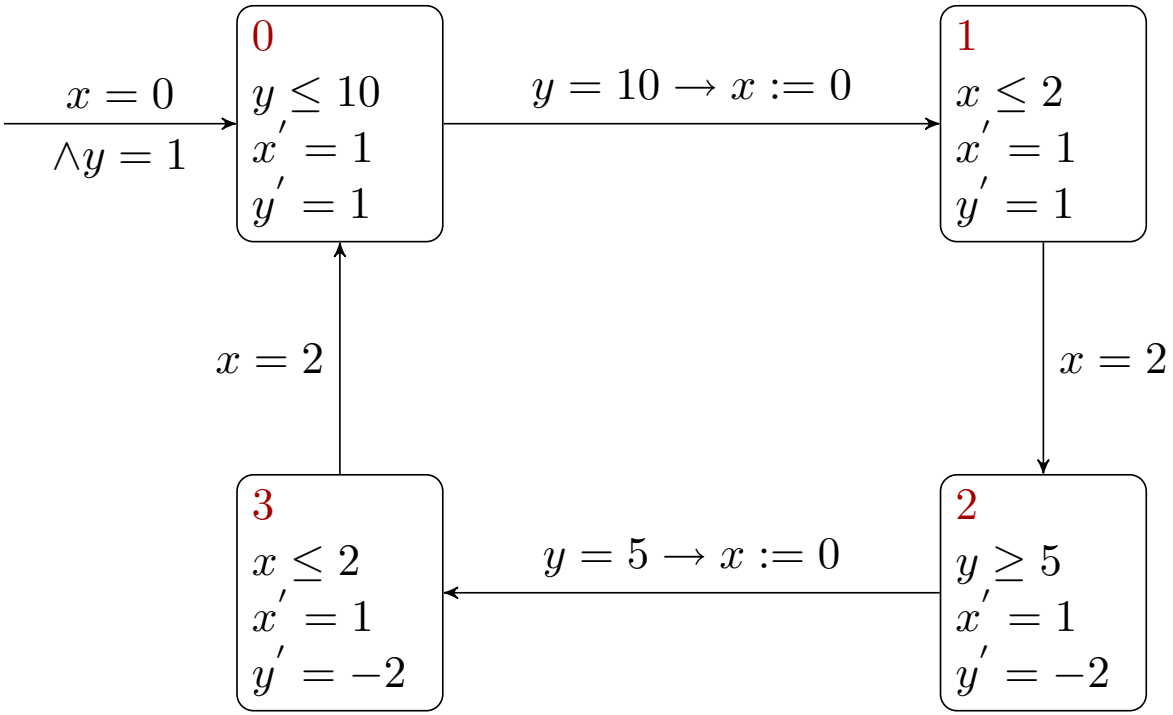
$$BMC_k(s_0, \ldots, s_k) :$$
$$\mathsf{Init}(s_0) \wedge \mathsf{Trans}(s_0, s_1) \wedge \ldots \wedge \mathsf{Trans}(s_{k-1}, s_k) \wedge \neg \mathsf{Safe}(s_k)$$

BMC_k is satisfiable $\iff$ exists run of length k leading to an unsafe state

$\rightarrow$ check BMC_k incrementally for $k = 0, 1, \ldots$ using a suitable solver

Discrete case: [Biere et al. 1999]

Example



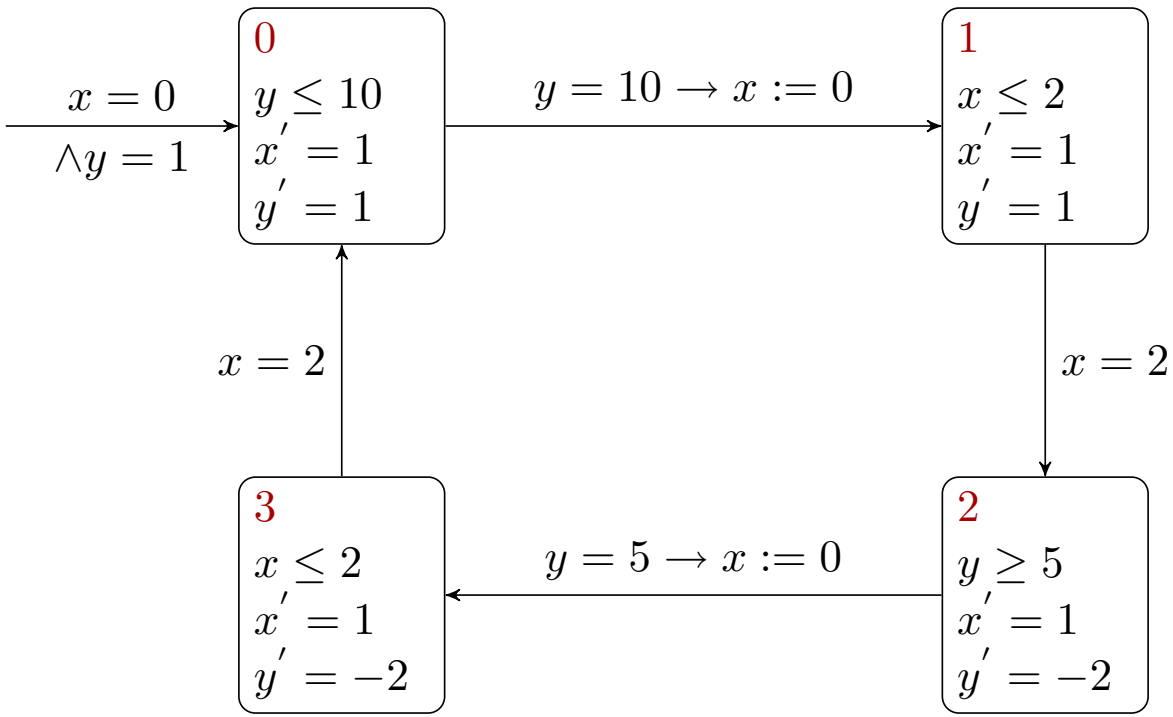
Notation:
$s = (l, x, y, t)$,
$s' = (l', x', y', t'), \ldots$.

$$Init(s) := l = 0 \wedge x = 0 \wedge y = 1$$

$$Trans(s, s') := T_0(s, s') \vee T_1(s, s') \vee T_2(s, s') \vee T_3(s, s') \vee D_{0,1}(s, s') \vee D_{1,2}(s, s') \vee D_{2,3}(s, s') \vee D_{3,1}(s, s')$$

$$Safe(s) := 1 \leq y \wedge y \leq 12$$

Example (cont.)



$$T_0(s, s', t) := l = 0 \wedge l' = 0 \wedge t \geq 0 \wedge x' = x + t \wedge y' = y + t \wedge y' \leq 10$$

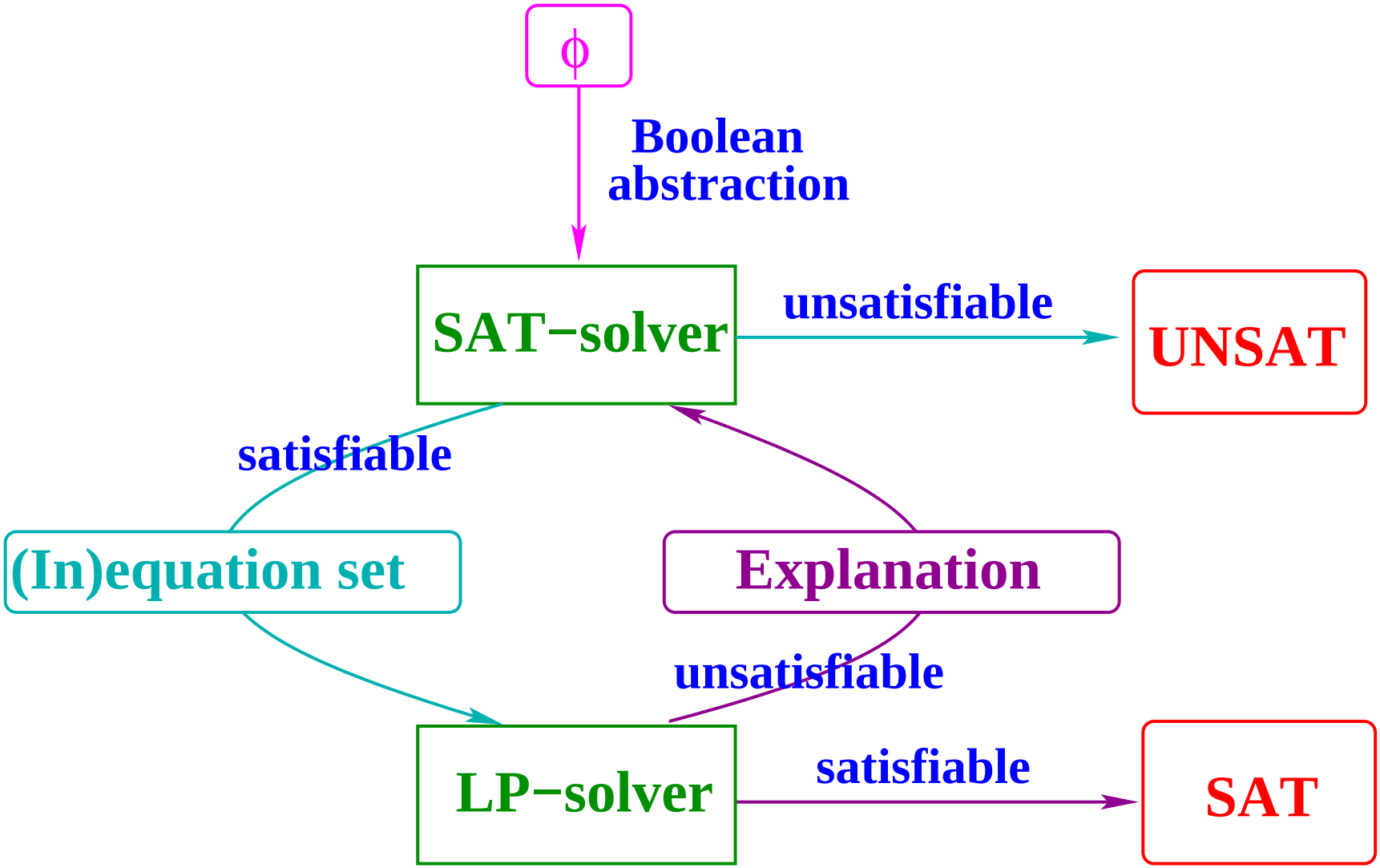
$$D_{0,1}(s, s') := l = 0 \wedge l' = 1 \wedge y = 10 \wedge x' = 0 \wedge y' = y \wedge x' \leq 2$$

Example (cont.)

$$
\begin{aligned}
BMC_0(s_0) &:= Init(s_0) \wedge \neg Safe(s_0) \\
BMC_1(s_0, s_1) &:= Init(s_0) \wedge Trans(s_0, s_1) \wedge \neg Safe(s_1) \\
BMC_2(s_0, s_1, s_2) &:= Init(s_0) \wedge Trans(s_0, s_1) \wedge Trans(s_1, s_2) \wedge \\
&\quad \neg Safe(s_2)
\end{aligned}
$$

...

Lazy satisfiability checking



SMT-solving: Example

Formula :	$x = 1$	$\wedge$	$y \geq 2$	$\wedge ($	$x < 0$	$\vee$	$x + y < 0)$		
Boolean abstraction :	a	$\wedge$	b	$\wedge ($	c	$\vee$	d	$)$	

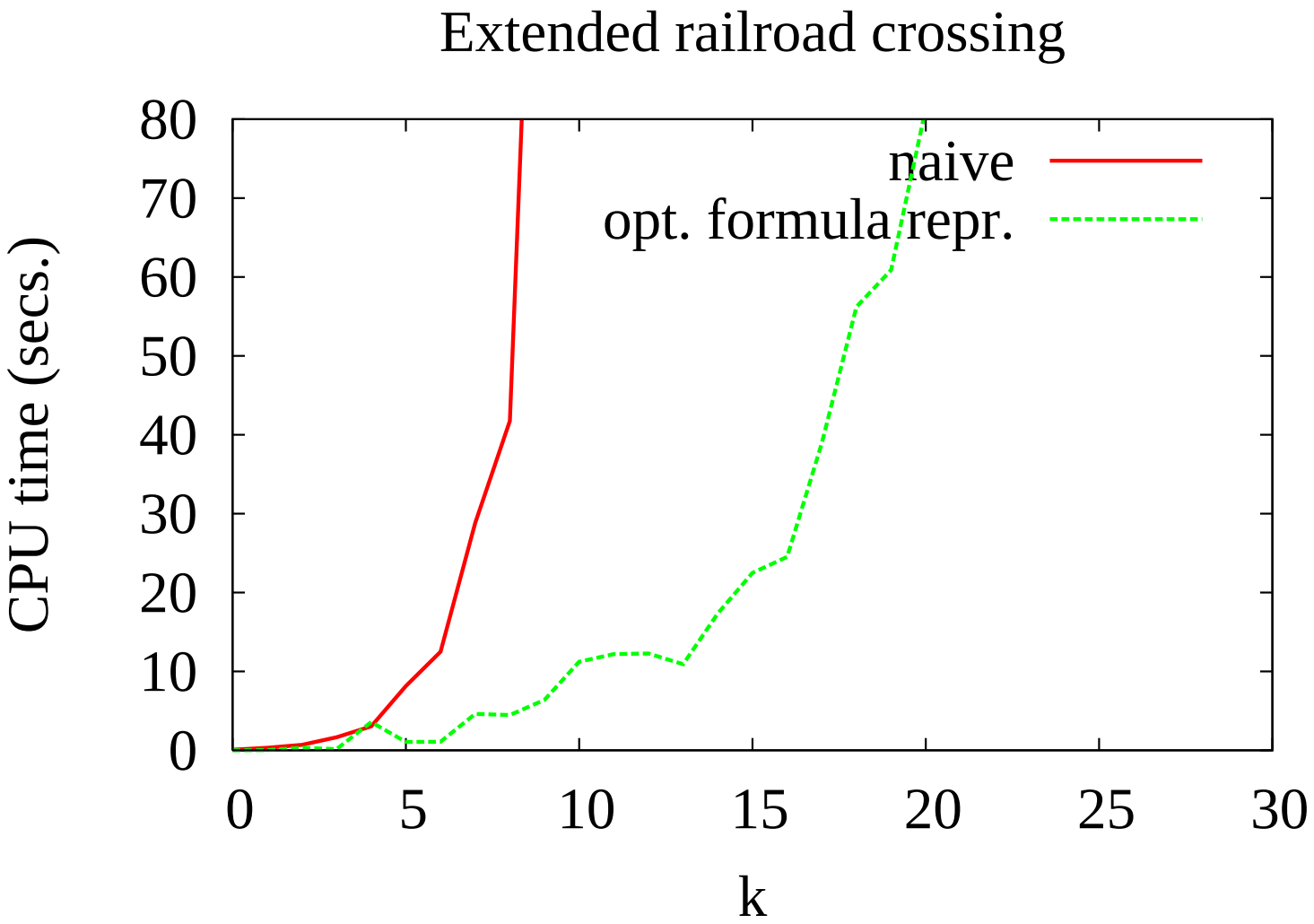
SAT−solver		*Theory solver*
$DL0 : a, b$	$\rightarrow$	$x = 1 \wedge y \geq 2$
$DL0 : a, b$		
$DL1 : c$	$\rightarrow$	$x = 1 \wedge y \geq 2 \wedge x < 0$: *unsat*
learn : $(\neg a \vee \neg c)$	$\leftarrow$	*minimal infeasible subset* :
		$x = 1 \wedge x < 0$
$DL0 : a, b, \neg c, d$	$\rightarrow$	$x = 1 \wedge y \geq 0 \wedge x \geq 0 \wedge x + y < 0$: *unsat*
learn : $(\neg a \vee \neg b \vee \neg d)$	$\leftarrow$	*minimal infeasible subset* :
		$x = 1 \wedge y \geq 0 \wedge x + y < 0$

Formula representation

Formula representation

1. Using Boolean variables for finite domains
   E.g., thermostat locations: at $=$ off $\rightsquigarrow b$, at $=$ on $\rightsquigarrow \neg b$
2. Excluding bad and initial state loops
3. Alternating flows and jumps
   - Instead of
     $(T(s_0, s_1) \vee D(s_0, s_1)) \wedge (T(s_1, s_2) \vee D(s_1, s_2)) \wedge \ldots$
   - we use
     $T(s_0, s_1) \wedge D(s_1, s_2) \wedge T(s_2, s_3) \wedge D(s_3, s_4) \wedge \ldots$.
4. Introducing τ-transitions

Results


Extended railroad crossing
CPU time (secs.)
k
naive
opt. formula repr.
0
10
20
30
40
50
60
70
80
0
5
10
15
20
25
30

Conflict sharing and replication

Conflict sharing and replication

Iteration k:

$$\boxed{I_0} \wedge \boxed{T_{0,1}} \wedge \boxed{T_{1,2}} \wedge \ldots \wedge \boxed{T_{k-2,k-1}} \wedge \boxed{T_{k-1,k}} \wedge \boxed{\neg S_k}$$

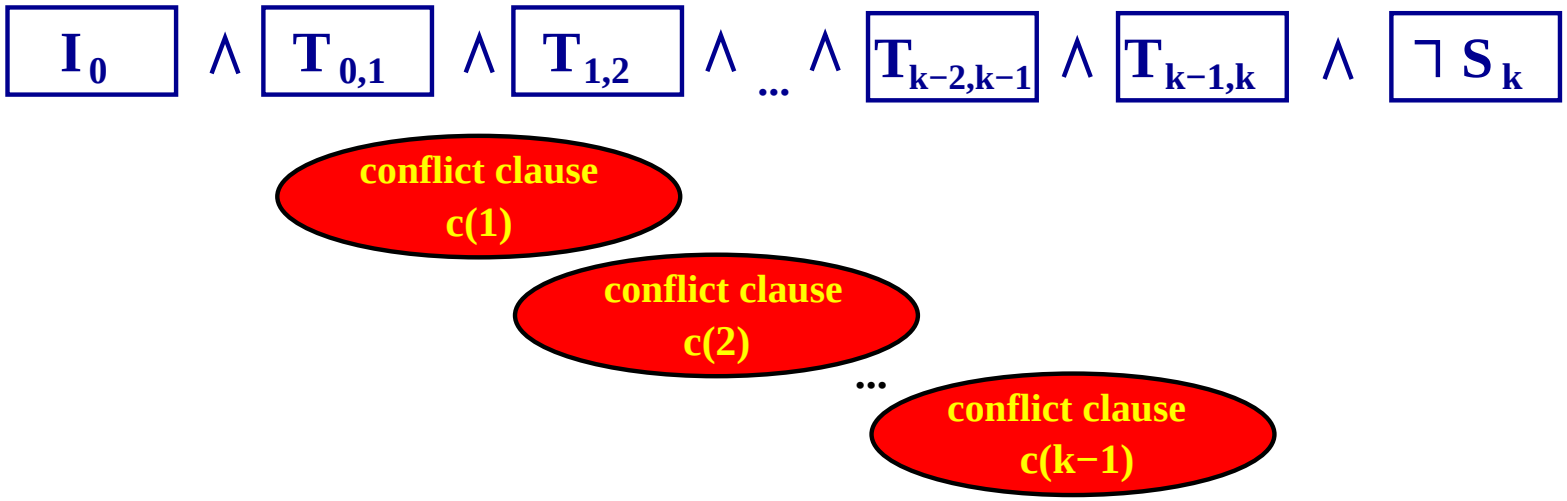
conflict clause c(1)

$$\neg c(1) \longrightarrow \neg\left(\boxed{T_{0,1}} \wedge \boxed{T_{1,2}}\right)$$

Discrete case: [Shtrichman 2001]

Conflict sharing and replication

Iteration k:



Conflict sharing and replication

Iteration k:

$$I_0 \wedge T_{0,1} \wedge T_{1,2} \wedge \ldots \wedge T_{k-2,k-1} \wedge T_{k-1,k} \wedge \neg S_k$$

conflict clause c(1)

conflict clause c(2)

...

conflict clause c(k−1)

Iteration k+1:

$$I_0 \wedge T_{0,1} \wedge T_{1,2} \wedge \ldots \wedge T_{k-2,k-1} \wedge T_{k-1,k} \wedge T_{k,k+1} \wedge \neg S_{k+1}$$

conflict clause c(1)

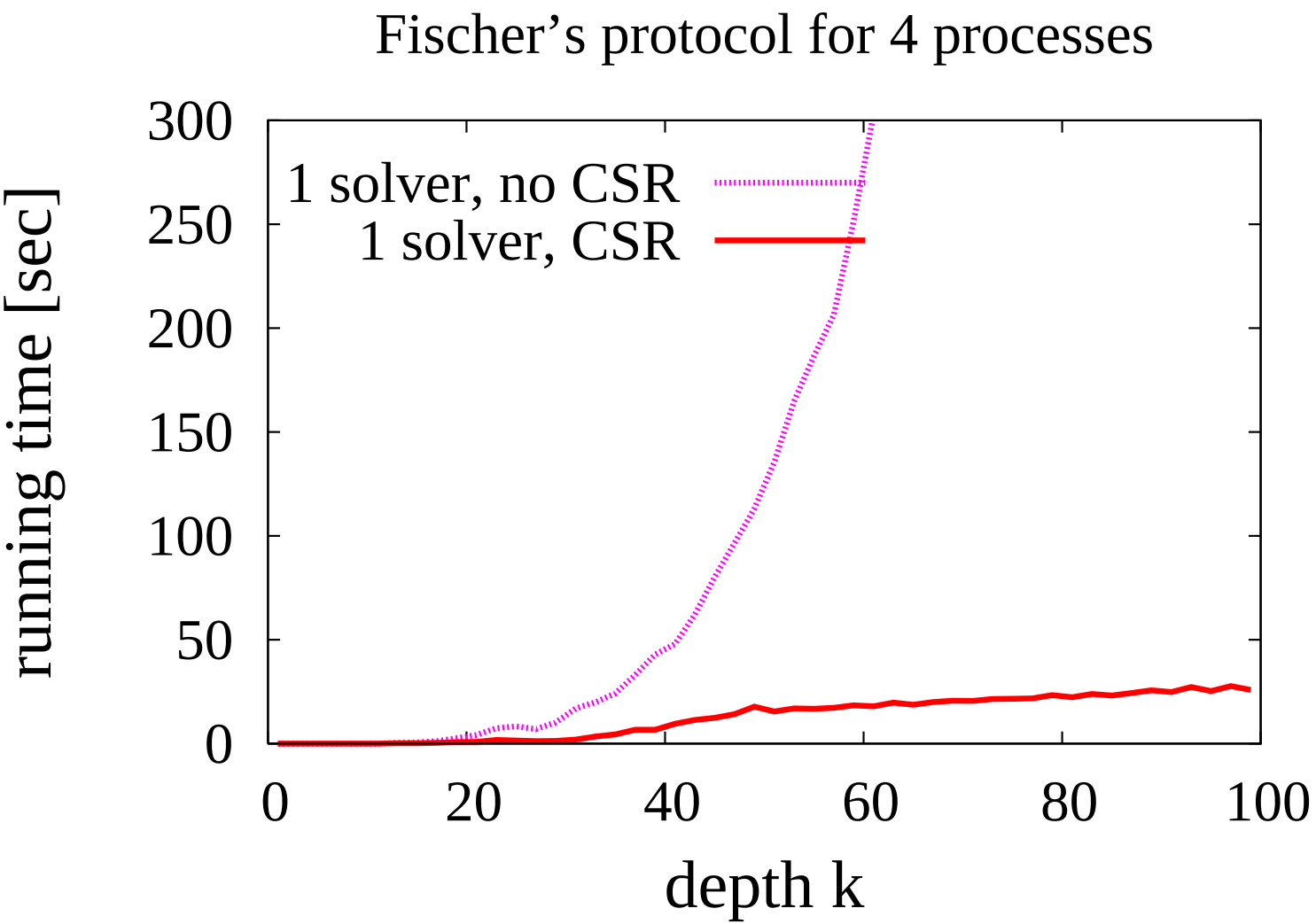
conflict clause c(2)

...

conflict clause c(k−1)

conflict clause c(k)

Results

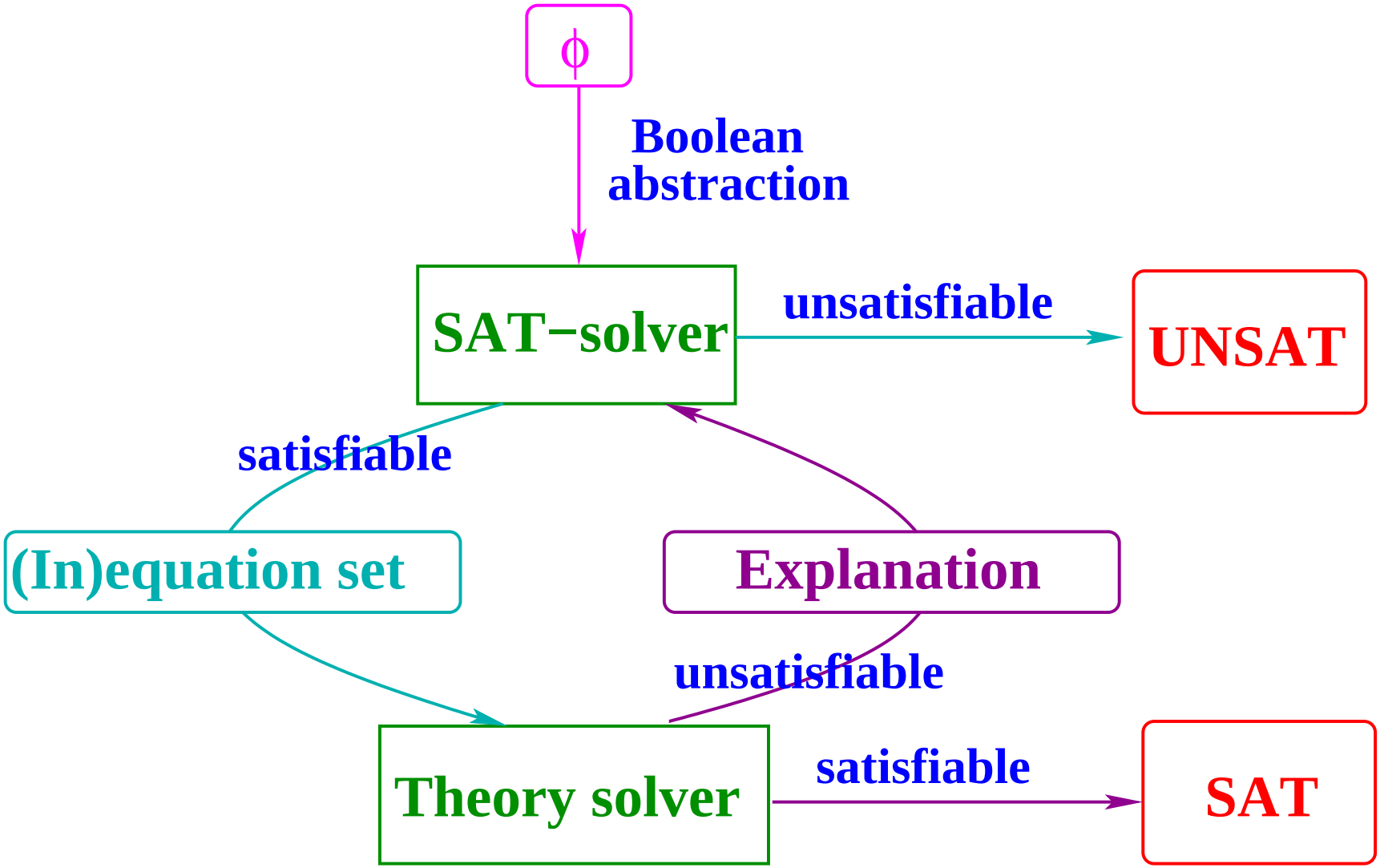

Fischer's protocol for 4 processes
running time [sec]
depth k
1 solver, no CSR
1 solver, CSR
0
50
100
150
200
250
300
0
20
40
60
80
100

Contents

Non-linear hybrid systems

- Instead of the LP-solver, integrate another theory solver, using, e.g.,
  - theorem proving, or
  - interval-based approximation, or
  - a quantifier elimination procedure.

Lazy satisfiability checking



Requirements on the theory solver

The extension to non-linear domains requires the invocation of theory solvers satisfying the following requirements:

- it works incrementally,
- it can compute minimal infeasible subsets, and
- it can backtrack.

Let us have a look at the first-order theory of the reals: $(\mathbb{R}, +, \cdot, <, 0, 1)$ is decidable (Tarski,1948) with upper bound for time complexity 2^{2^n} where n is the number of variables (Brown and Davenport, 2007).

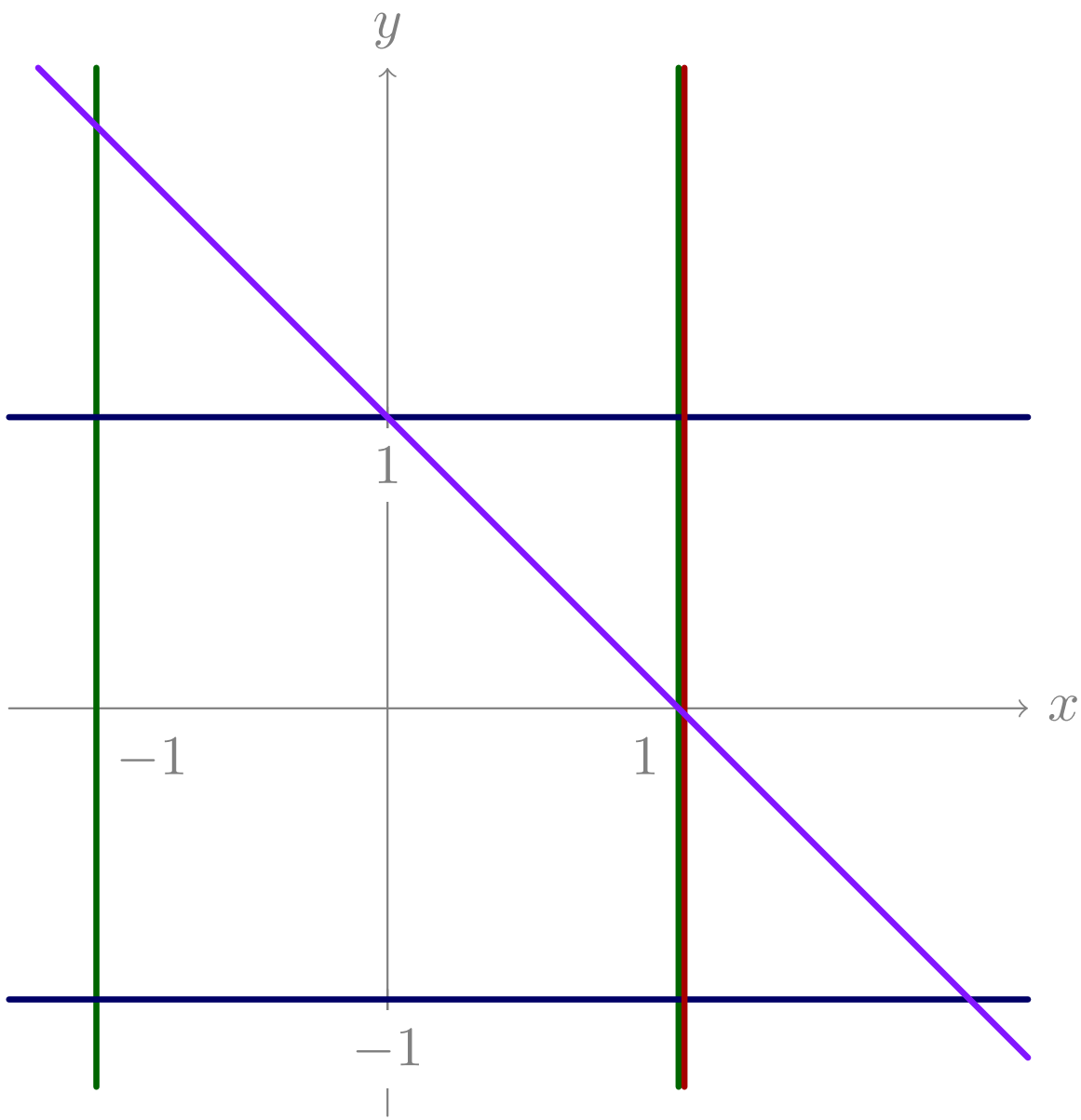
Theory solver: existing implementations

- Cylindrical algebraic decomposition (CAD)
- Gröbner bases
- Virtual substitution
- Interval arithmetic

Theory solver: CAD

$$x^2 - 1 = 0 \wedge$$
$$y^2 - 1 = 0 \wedge$$
$$x - 1 = 0 \quad \wedge$$
$$x + y - 1 = 0$$

- incremental $\star\star$
- minimal infeasible subset $\star$
- backtracking $\star\star$



Theory solver: Gröbner bases

Mathematical background

$p_1 = 0 \wedge \ldots \wedge p_k = 0$ has no solution $\iff$ $q_1 p_1 + \cdots + q_k p_k = 1$ for some polynomials $q_1, \ldots, q_k$

$x^2 - 1 = 0$	$\{x^2 - 1\}$
$y^2 - 1 = 0$	$\{x^2 - 1, y^2 - 1\}$
$x - 1 = 0$	$\{x^2 - 1, y^2 - 1, x - 1\}$
$x^2 - 1 = (x + 1)(x - 1)$	$\{y^2 - 1, x - 1\}$
$x - y + 1 = (x - 1) - (y - 2) = 0$	$\{y^2 - 1, x - 1, y - 2\}$
$y^2 - 1 = (y + 2)(y - 2) + 2 \cdot 1$	$\{x - 1, y - 2, 1\}$

- incremental $\star\star$
- minimal infeasible subset $\star\star$
- backtracking $\star\star$