FMCAD 2013

Parameter Synthesis with IC3

A. Cimatti, *A. Griggio*, S. Mover, S. Tonetta
FBK, Trento, Italy

Motivations and Contributions

- Parametric descriptions of systems arise in many domains
  - E.g. software, cyber-physical systems, task scheduling, ...
- Important problem: find parameter values that guarantee the satisfaction of a given property
- This work: exploit (SMT aware) IC3 for parameter synthesis
  - Simple extension of IC3
    - Exploit incrementality and generation of multiple counterexamples
  - Gives optimal parameter region for a given property
  - Promising experimental results

Problem definition

- Symbolic transition system $S = \langle X, I, T \rangle$
  - State variables X
  - Initial-state formula $I(X)$
  - Transition relation $T(X, X')$
- Parametric system $S = \langle U, X, I, T \rangle$
  - Set of parameters U
  - Init $I(U, X)$ and trans $T(U, X, X')$
  - Valuation γ of U induces $S_\gamma = \langle X, \gamma(I), \gamma(T) \rangle$

- Synthesis problem:
  - Given a property $P(U, X)$
  - Find *all* valuations ρ of U such that $\gamma \in \rho$ iff $S_\gamma \models \gamma(P)$

Our starting point: [RTSS'08]

Start from $\begin{cases} \rho = \top, k = 0 \\ S = \langle X \cup U, I(X) \wedge \rho, T(X, X') \wedge \rho \wedge \bigwedge_{u \in U} u' = u \rangle \end{cases}$

BMC-check(S, P, k) $\xrightarrow{\text{Unsafe}}$ compute $\mathrm{bad}(U) = \exists X, X', \ldots, X^k . \mathrm{BMC}_k^{\pi}$

update $\rho_1 := \rho_0 \wedge \neg \mathrm{bad}$

Our starting point: [RTSS'08]

FONDAZIONE BRUNO KESSLER

Start from $\begin{cases} \rho = \top, k = 0 \\ S = \langle X \cup U, I(X) \wedge \rho, T(X, X') \wedge \rho \wedge \bigwedge_{u \in U} u' = u \rangle \end{cases}$

BMC-check(S, P, k)

$\xrightarrow{\text{Unsafe}}$

compute

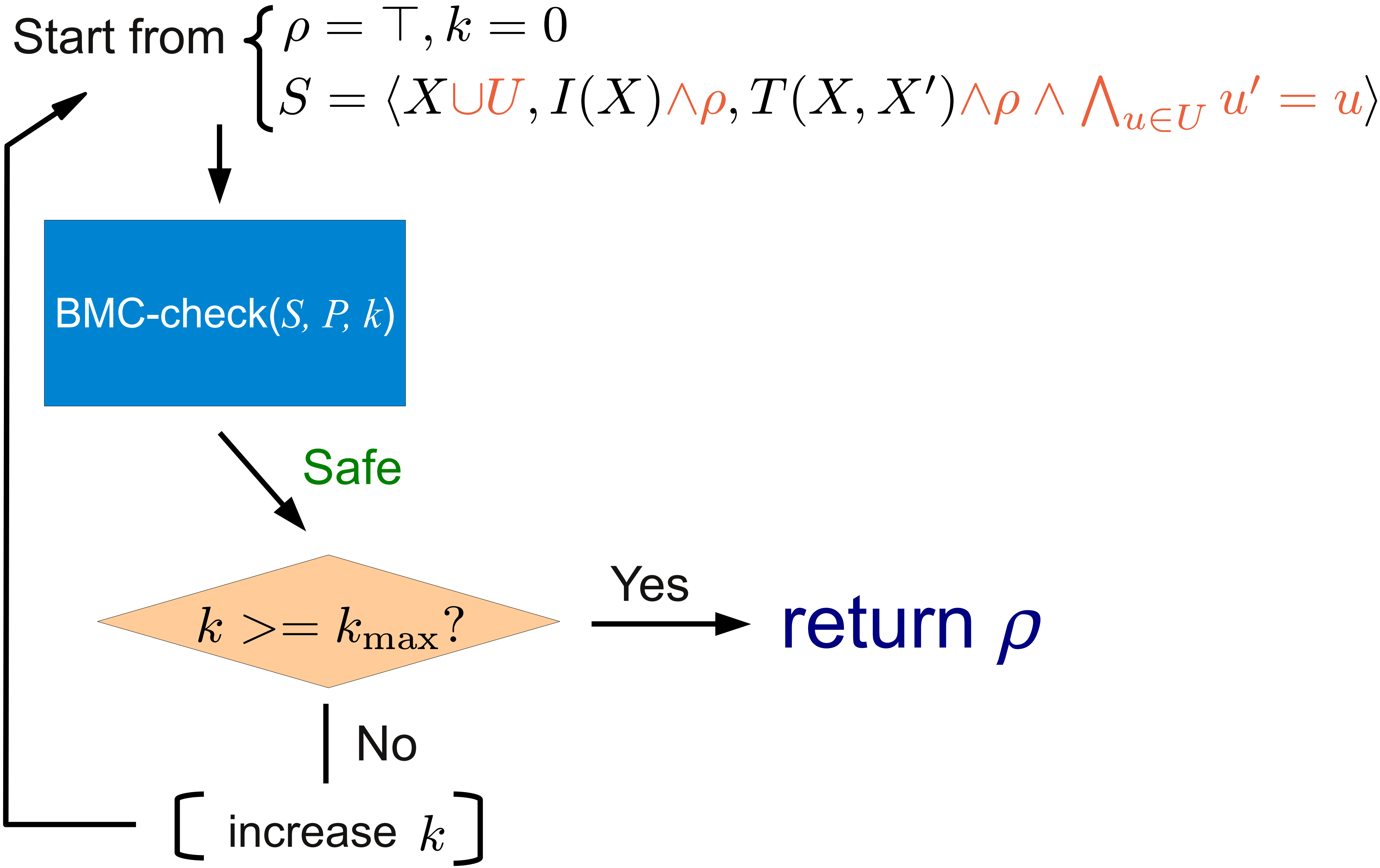
$\mathrm{bad}(U) = \exists X, X', \ldots, X^k \, \boxed{\mathrm{BMC}_k^\pi}$

BMC formula simplified by fixing Boolean variables to the values found in the counterexample trace

update

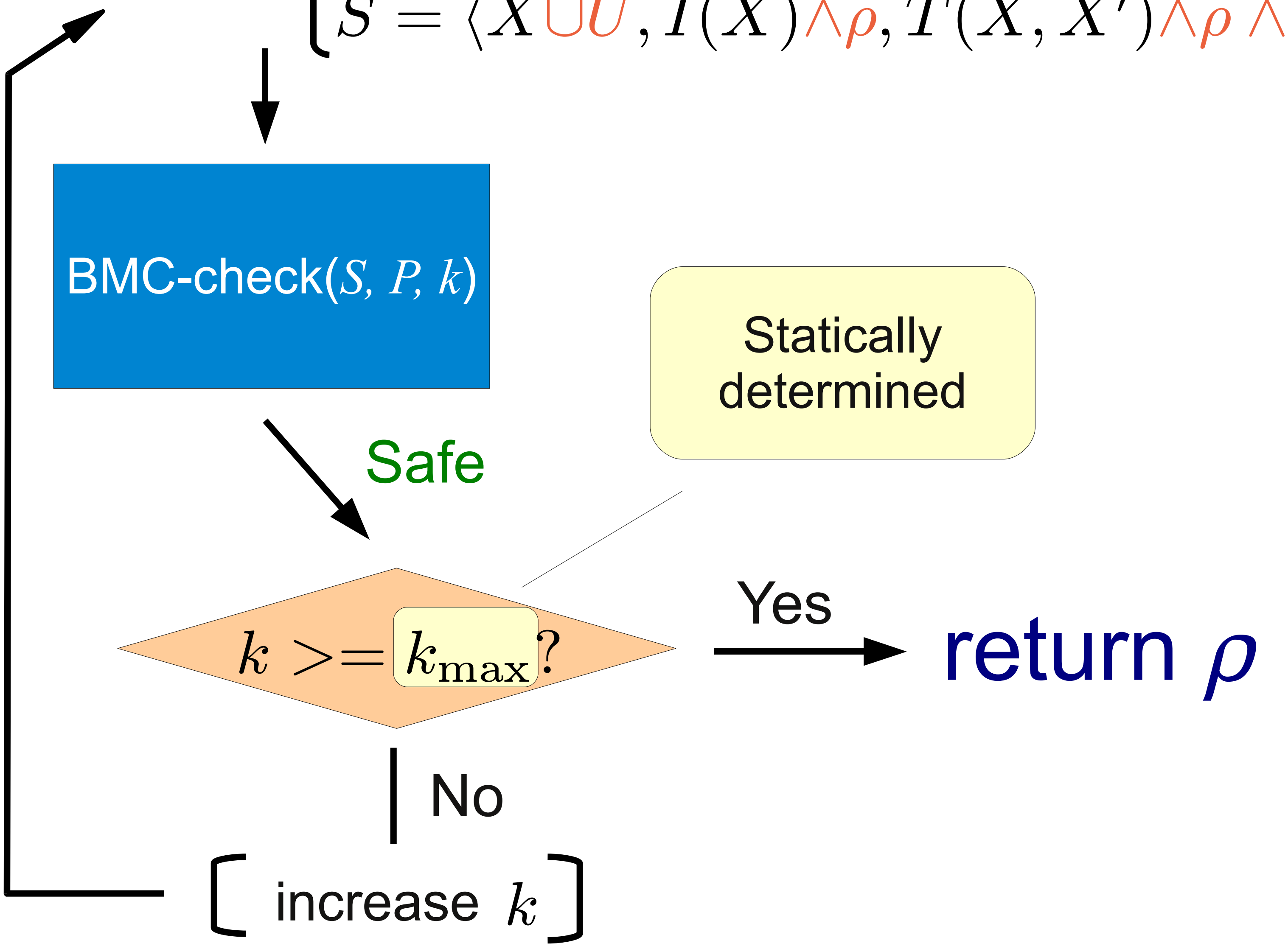
$\rho_1 := \rho_0 \wedge \neg \mathrm{bad}$

Our starting point: [RTSS'08]



Our starting point: [RTSS'08]

Start from $\begin{cases} \rho = \top, k = 0 \\ S = \langle X \cup U, I(X) \wedge \rho, T(X, X') \wedge \rho \wedge \bigwedge_{u \in U} u' = u \rangle \end{cases}$



Drawbacks of [RTSS'08]

FONDAZIONE BRUNO KESSLER

(1) BMC-based, needs to know k_{max} to terminate

- Implementation in [RTSS'08] only for task scheduling problems
  - k_{max} computed from domain knowledge

(2) Quantifier elimination is a bottleneck

- As k grows, quant elim becomes prohibitively expensive
- Even if BMC_k^{π} is used

Drawbacks of [RTSS'08]

(1) BMC-based, needs to know k_{max} to terminate

- Implementation in [RTSS'08] only for task scheduling problems
  - k_{max} computed from domain knowledge

(2) Quantifier elimination is a bottleneck

- As k grows, quant elim becomes prohibitively expensive
- Even if BMC_k^{π} is used

- Solution for (1): use IC3-SMT instead of BMC
  - But still (2) is a problem!
    - We can do better with a tighter integration with IC3

IC3 with SMT [CAV'12]

- IC3 main features (for this work):
  - incremental construction of clauses
  - from counterexamples to induction
  - by recursively blocking predecessors of bad states
    - if initial states are reached, we have a counterexample trace

IC3 with SMT [CAV'12]

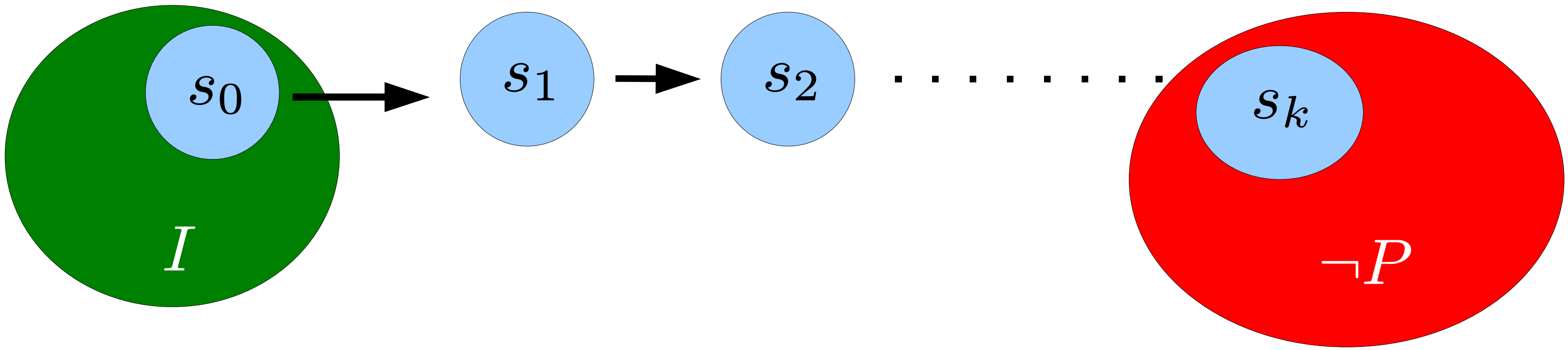
FONDAZIONE BRUNO KESSLER

- IC3 main features (for this work):
  - incremental construction of clauses
  - from counterexamples to induction
  - by recursively blocking predecessors of bad states
    - if initial states are reached, we have a counterexample trace

- We exploit a property of (the SMT extension of) IC3:
  - a counterexample trace represents multiple counterexamples
    - because predecessors are computed with (approximated) quantifier elimination [CAV'12]

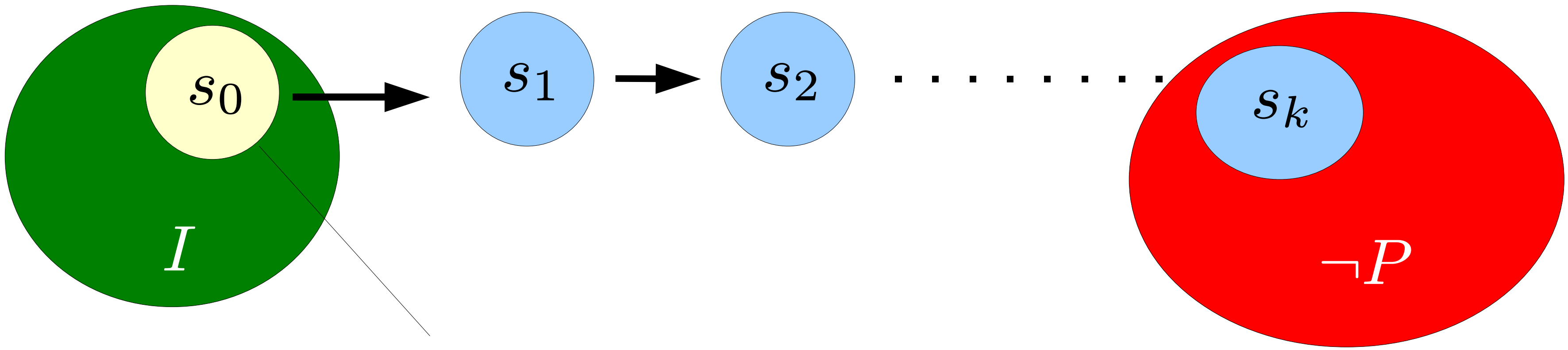
Exploiting IC3-SMT counterexamples

- Consider the cex $s_0(X,U), s_1(X,U), \ldots, s_k(X,U)$



Exploiting IC3-SMT counterexamples

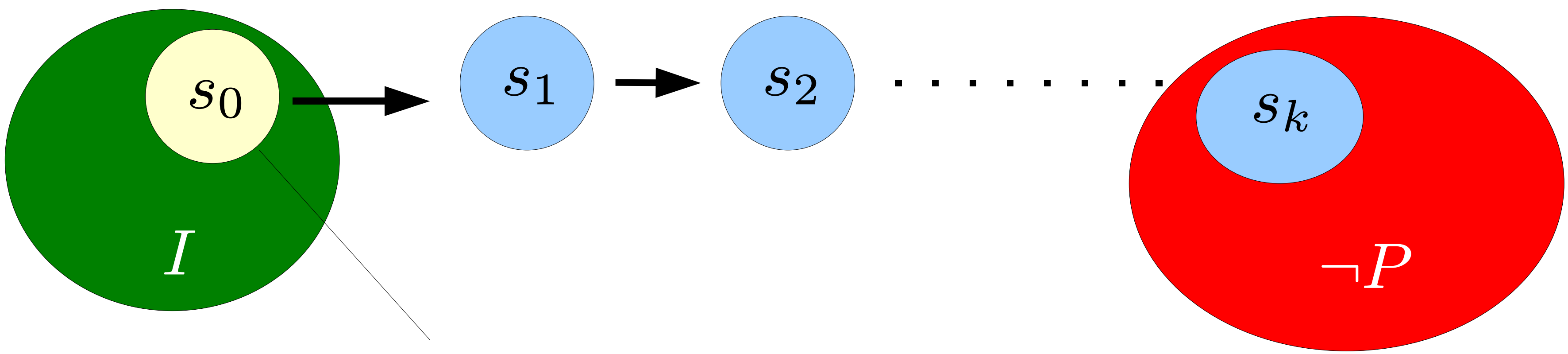
- Consider the cex $s_0(X, U), s_1(X, U), \ldots, s_k(X, U)$



- Two (simple) observations:
  - $s_0(X, U)$ represents multiple states "by construction"
  - ALL the states in $s_0(X, U)$ are bad and need to be blocked

Exploiting IC3-SMT counterexamples

- Consider the cex $s_0(X,U), s_1(X,U), \ldots, s_k(X,U)$



- Two (simple) observations:
  - $s_0(X,U)$ represents multiple states “by construction”
  - ALL the states in $s_0(X,U)$ are bad and need to be blocked

- Therefore, we can use the cheaper $\mathrm{bad}(U) = \exists X . s_0(X,U)$ instead of $\mathrm{bad}(U) = \exists X, X', \ldots, X^k . \mathrm{BMC}_k^\pi$

IC3-based algorithm

Start from $\begin{cases} \rho = \top \\ S = \langle X \cup U, I(X) \wedge \rho, T(X, X') \wedge \rho \wedge \bigwedge_{u \in U} (u' = u) \rangle \end{cases}$

IC3-check(S, P)

Unsafe

$\left[\begin{array}{l} \text{get counterexample trace} \\ s_0(X, U), s_1(X, U), \ldots, s_k(X, U) \end{array}\right]$

compute

$\mathrm{bad}(U) = \exists X . s_0(X, U)$

$\left[\begin{array}{l} \text{update} \\ \rho := \rho \wedge \neg \mathrm{bad} \end{array}\right]$

Safe

return ρ

Optimizations

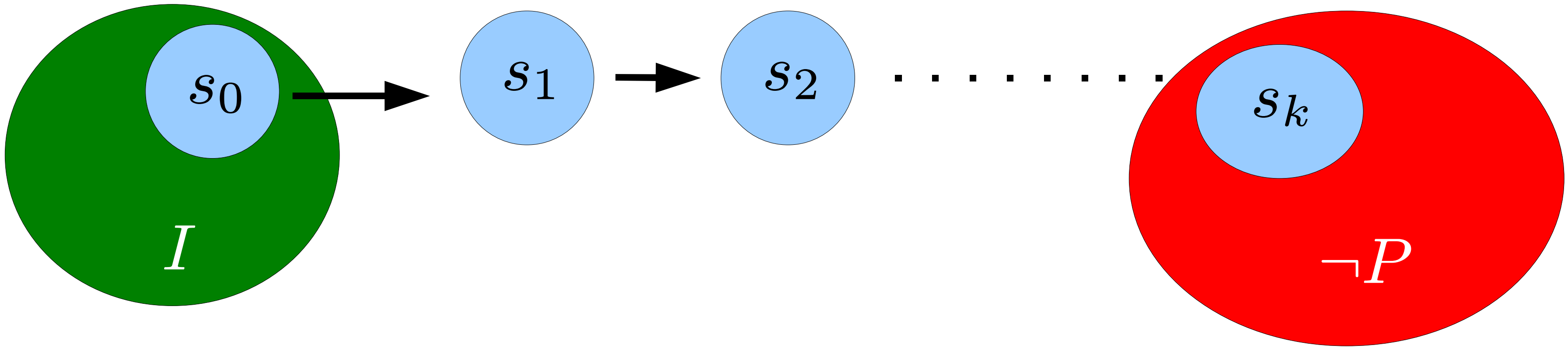
(1) Exploit incrementality

- At each iteration:
  - $I_{\text{new}} := I \wedge \neg \text{bad}$
  - $T_{\text{new}} := T \wedge \neg \text{bad}$

- No need to restart from scratch, can keep all the previous F_i's
  - Similarly, exploit incrementality in the underlying SMT solver

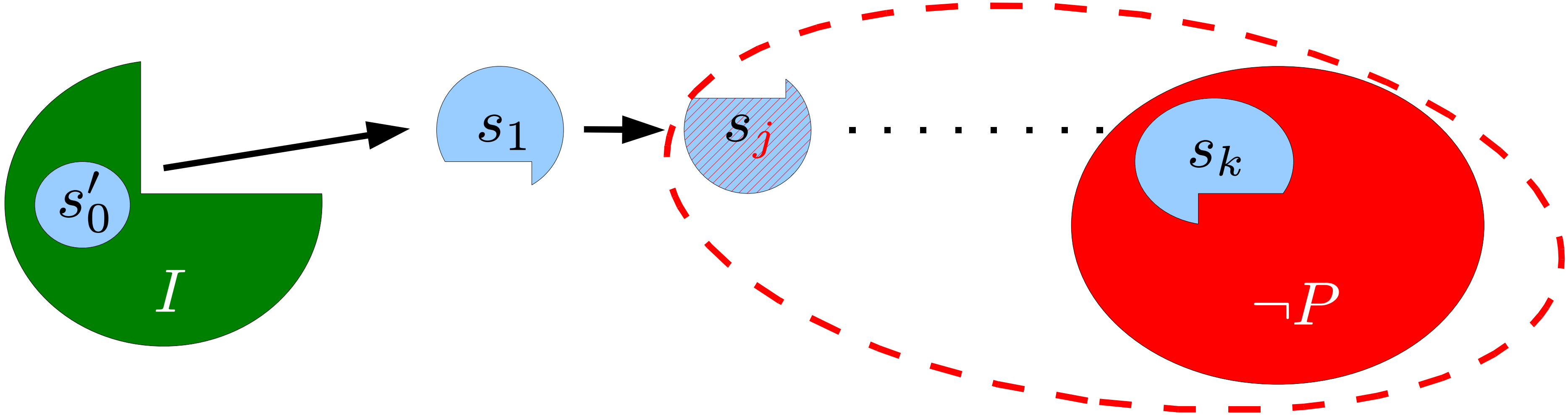
Optimizations

(2) The IC3 cex trace allows to play with the tradeoff generality / cost of quantifier elimination



Optimizations

(2) The IC3 cex trace allows to play with the tradeoff generality / cost of quantifier elimination



- Each state s_j is bad, because it leads to $\neg P$
- Can also try blocking $\exists X, X', \ldots, X^j . s_0(X, U) \wedge T \ldots \wedge s_j(X^j, U)$
- Or in the limit $\exists X, X', \ldots, X^k . I(X, U) \wedge T \ldots \wedge \neg P(X^k, U)$
- Various heuristics are possible (see paper)

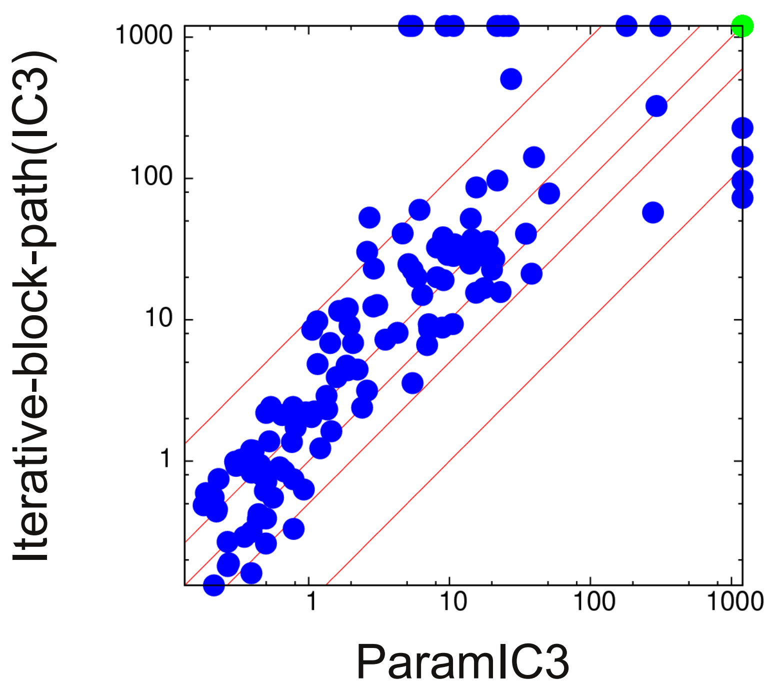
Experimental evaluation

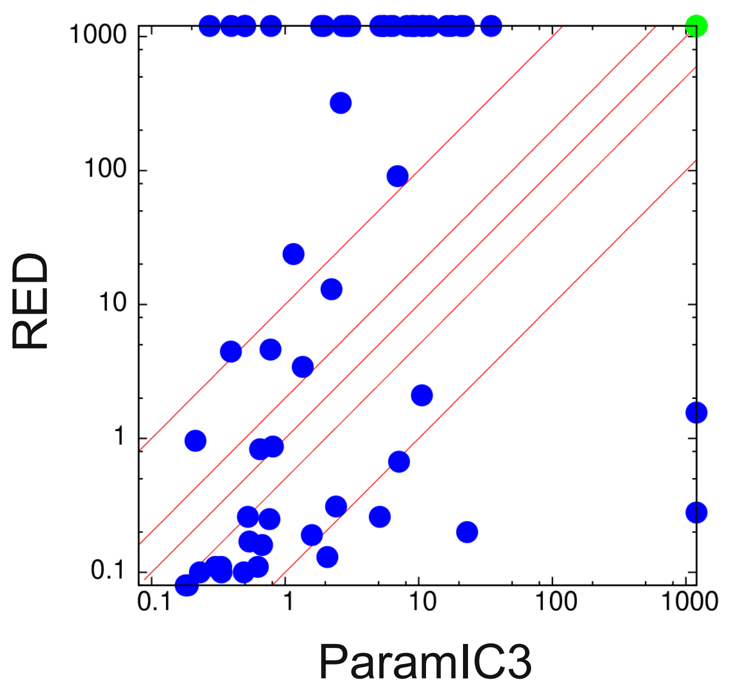
- Implemented in the IC3-SMT tool of [CAV'12]
  - Using MathSAT for SMT check and quantifier elimination

- Comparison with:
  - Non incremental algorithm of [RTSS'08], but using IC3
    - "black box" use of IC3
  - RED [Wang'05], a state-of-the-art tool for linear-hybrid automata
    - Based on the computation of reachable states
    - Specialized for hybrid automata

- Benchmarks from linear hybrid systems

Results


1000
100
10
1
1
10
100
1000
Iterative-block-path(IC3)
ParamIC3

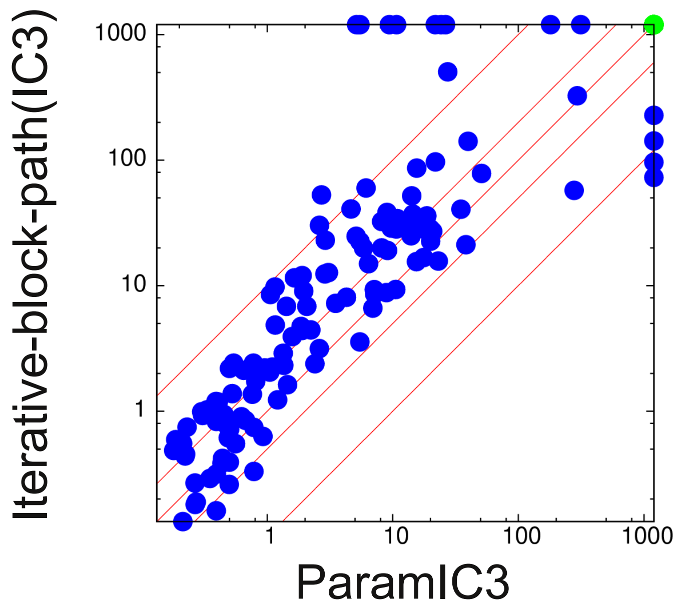

1000
100
10
1
0.1
0.1
1
10
100
1000
RED
ParamIC3

Conclusions

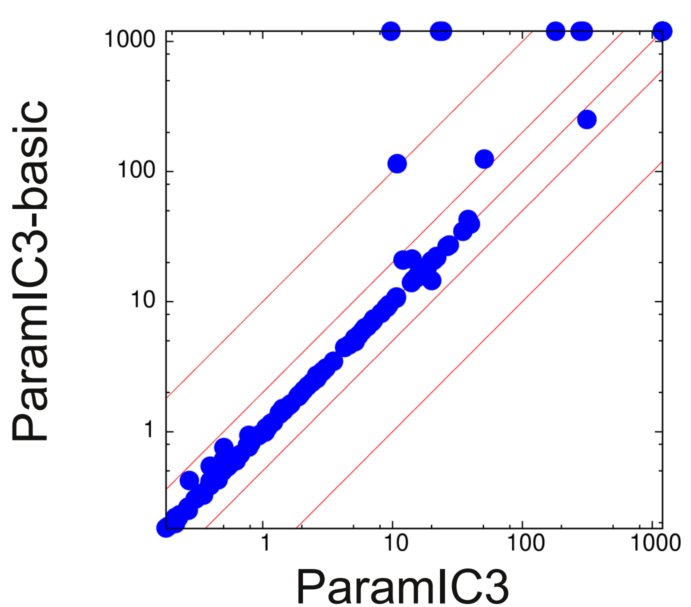
- Simple extension of IC3-SMT for parameter synthesis
- Exploit IC3 features
  - Construction of a trace encoding multiple counterexamples
  - Incrementality
  - Allows to control cost of quantifier elimination
- Easy to implement
- Compares positively with alternative approaches

Thank You

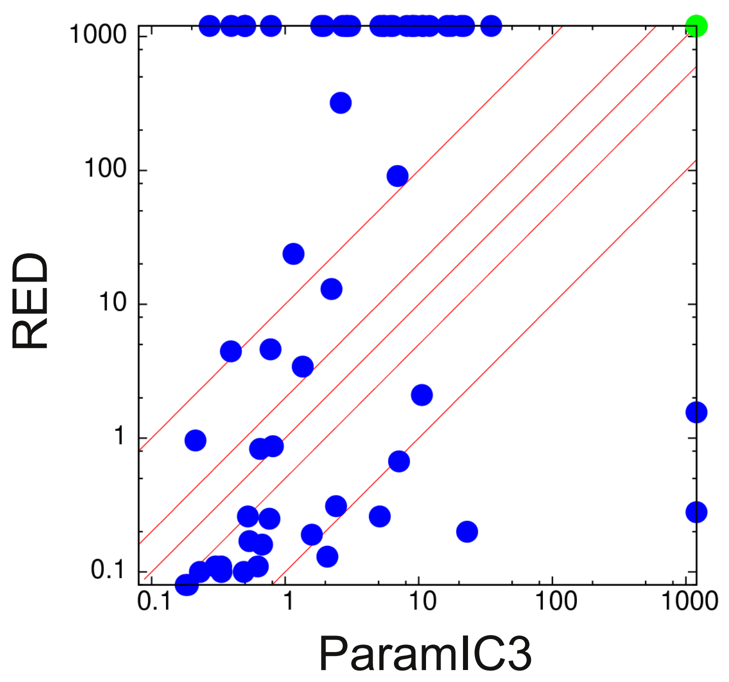
Results



Iterative-block-path(IC3)
1000
100
10
1
1
10
100
1000
ParamIC3



ParamIC3-basic
1000
100
10
1
1
10
100
1000
ParamIC3



RED
1000
100
10
1
0.1
0.1
1
10
100
1000
ParamIC3