The CEBug project

Erika Ábrahám[1] Nils Jansen[1] Joost-Pieter Katoen[1]
Bernd Becker[2] Bettina Braitling[2] Ralf Wimmer[2]

[1]RWTH Aachen University, Germany

[2]Albert-Ludwigs-University Freiburg, Germany

ROCKS project workshop

Oct 06, 2011

1 Stochastic counterexamples

2 First CEBug example:
Path-based counterexamples via bounded model checking

3 Second CEBug example:
Counterexamples as minimal critical subsystems

Counter**e**xample generation for stochastic systems using **bo**u**n**ded model checkin**g**

DTMC/MDP/MRM model checking
$\longrightarrow$ property is violated
$\longrightarrow$ counterexample

Counter**e**xample generation for stochastic systems using **bo**u**n**ded model checkin**g**

DTMC/MDP/MRM model checking
$\longrightarrow$ property is violated
$\longrightarrow$ counterexample

DTMC model checking $\rightarrow$ property is violated $\rightarrow$ counterexample

A discrete-time Markov chain (DTMC) is a tuple

$$M = (S, s_{init}, P, L).$$

A discrete-time Markov chain (DTMC) is a tuple

$$M = (S, s_{init}, P, L).$$

S: finite state space

A discrete-time Markov chain (DTMC) is a tuple

$$M = (S, s_{init}, P, L).$$

initial state $s_{init} \in S$

A discrete-time Markov chain (DTMC) is a tuple

$$M = (S, s_{init}, P, L).$$

P: transition probability matrix $P : S \times S \rightarrow [0, 1]$

A discrete-time Markov chain (DTMC) is a tuple

$$M = (S, s_{init}, P, L).$$

L: state labeling function $L : S \to 2^{AP}$

DTMC model checking $\longrightarrow$ property is violated $\longrightarrow$ counterexample

- linear equation system
- SCC-based model checking
- simulation
- ...

DTMC model checking $\rightarrow$ property is violated $\rightarrow$ counterexample

Counterexamples for Kripke structures and LTL properties:

- by-product of model checking
- consist of a single path

Counterexamples for Kripke structures and LTL properties:

- by-product of model checking
- consist of a single path

Counterexamples for DTMCs and probabilistic properties:

- not provided by model checking
- consist of a large (maybe infinite) set of paths

Some state-of-the-art methods

- shortest path *(Damman, Han, and Katoen 2008) (Günther, Schuster & Siegle, 2010)*
- SCC abstraction *(Andrés, D'Argenio and van Rossum, 2008)*

Counterexamples are represented as

- sets of paths, path trees,
- regular expressions

CEBug aims

In CEBug we develop further counterexample generation and representation techniques.

In this talk: two areas based on

- SAT- and SMT-based bounded model checking
- SMT- and MILP-based computation of minimal critical subsystems

1 Stochastic counterexamples

2 First CEBug example:
Path-based counterexamples via bounded model checking

3 Second CEBug example:
Counterexamples as minimal critical subsystems

SBMC: stochastic bounded model cheking

- reduce the PCTL model checking problem to a probabilistic reachability problem
- finite paths $\sigma_0, \ldots, \sigma_k$ of length k that lead to a target state can be encoded as solutions to the propositional logic formula

$$\mathit{Init}(\sigma_0) \wedge \left(\bigwedge_{i=0}^{k-1} \mathit{Trans}(\sigma_i, \sigma_{i+1}) \right) \wedge \mathit{Target}(\sigma_k)$$

- use SAT-solving to find solutions with increasing k until the probability mass is sufficient
- important: loop detection
- important: encoding of (MT)BDDs

SSBMS: SMT-based stochastic bounded model cheking

Problem: we find paths with fewer transitions first, instead of paths with higher probabilities.

- transform probabilities on logarithmic scale
- consider transition probabilities in bounded model checking:

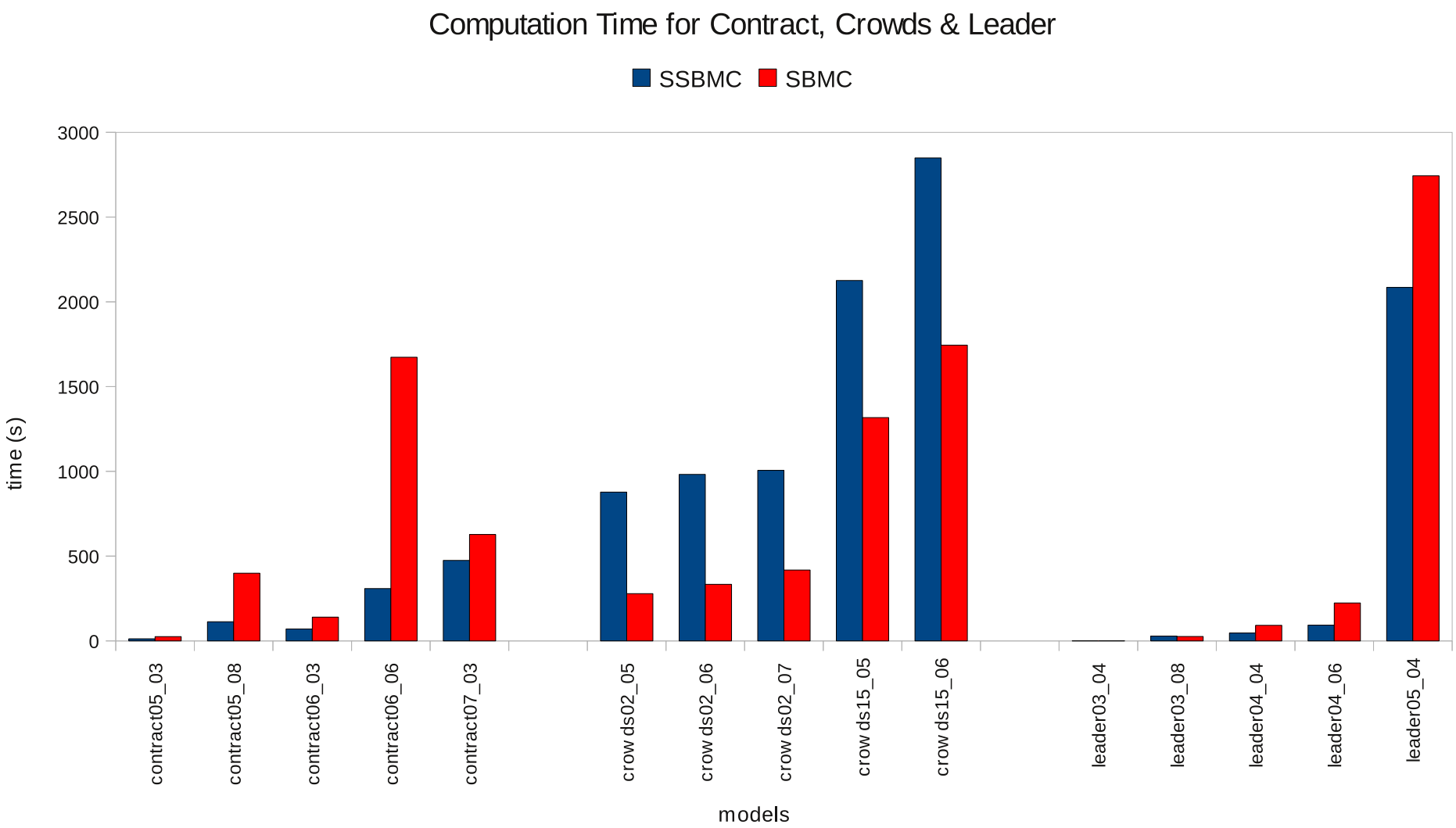
$$\mathit{Init}(\sigma_0) \wedge \left(\bigwedge_{i=0}^{k-1} \mathit{Trans}(\sigma_i, \sigma_{i+1}, \hat{p}_i) \right) \wedge \mathit{Target}(\sigma_k) \wedge \sum_{i=0}^{k-1} \hat{p}_i \geq \log p_t$$

- use SMT-solving to find solutions with increasing k / decreasing p_t until the probability mass is sufficient

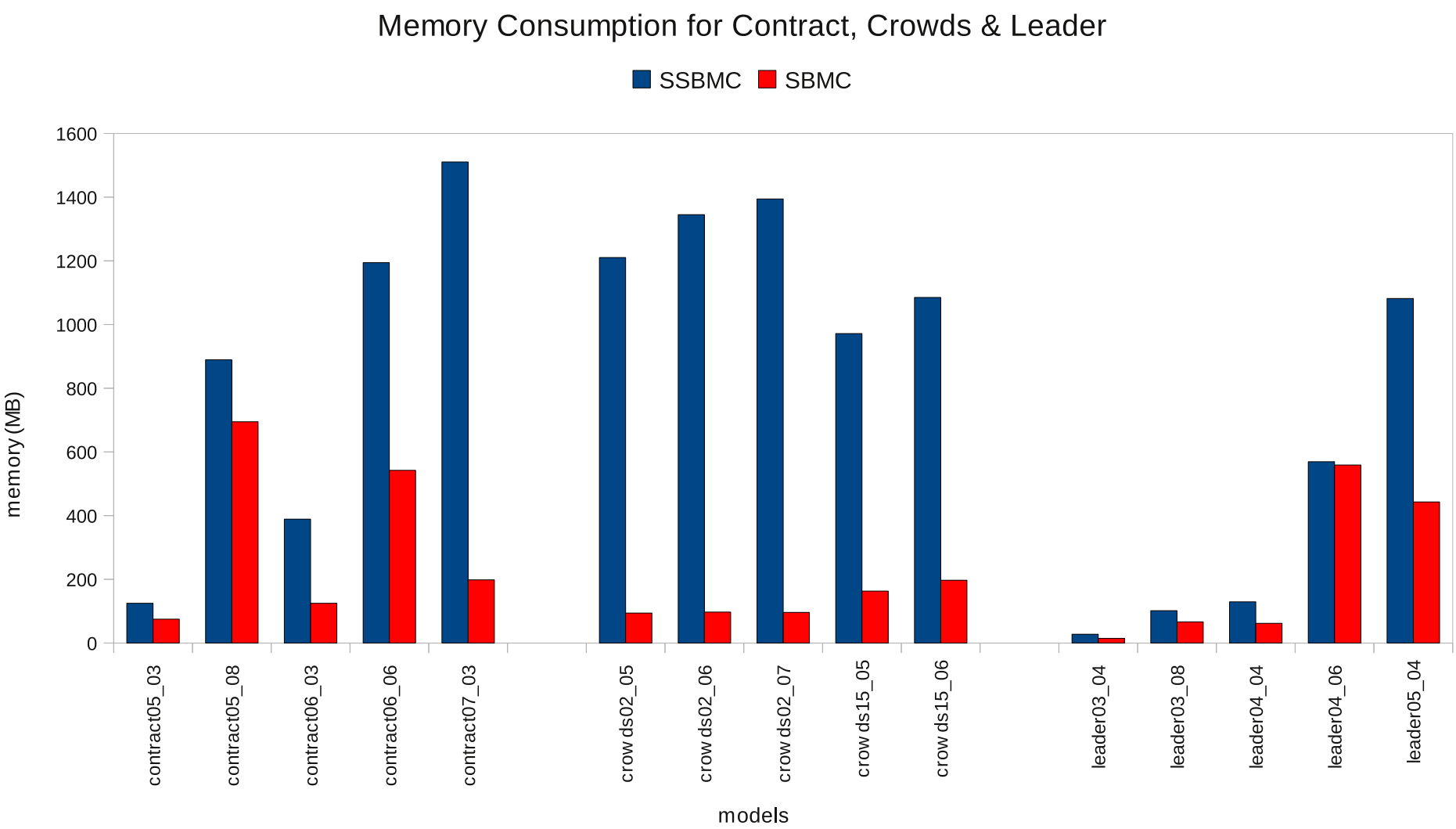
Experimental Results: SBMC (SAT) vs. SSBMS (SMT)

- **Benchmarks:** Contract signing protocol, Crowds protocol, Leader election protocol, Self-stabilizing minimal spanning tree algorithm
- **Solver:** Minisat, Yices

SMT vs. SAT (1)


Computation Time for Contract, Crowds & Leader
SSBMC
SBMC
time (s)
3000
2500
2000
1500
1000
500
0
contract05_03
contract05_08
contract06_03
contract06_06
contract07_03
crowds02_05
crowds02_06
crowds02_07
crowds15_05
crowds15_06
leader03_04
leader03_08
leader04_04
leader04_06
leader05_04
models

SMT vs. SAT (2)


Memory Consumption for Contract, Crowds & Leader
SSBMC
SBMC
memory (MB)
1600
1400
1200
1000
800
600
400
200
0
contract05_03
contract05_08
contract06_03
contract06_06
contract07_03
crow ds02_05
crow ds02_06
crow ds02_07
crow ds15_05
crow ds15_06
leader03_04
leader03_08
leader04_04
leader04_06
leader05_04
models

Further Results

Minimal Spanning Tree:

			SSBMC			**SBMC**		
Name	p	$k_{\max}$	#paths	time	mem.	#paths	time	mem.
mst15	0.049	15	4531	98.58	148.82	$>$ 600000	– TO –	
mst16	0.047	16	4648	107.27	158.25	$>$ 600000	– MO –	
mst18	0.036	18	4073	109.26	164.24	$>$ 600000	– MO –	
mst20	0.034	20	452	19.57	58.21	$>$ 500000	– TO –	

SAT-search for more probable paths first

Problem: longer but more probable paths are found only after we have searched for shorter paths.

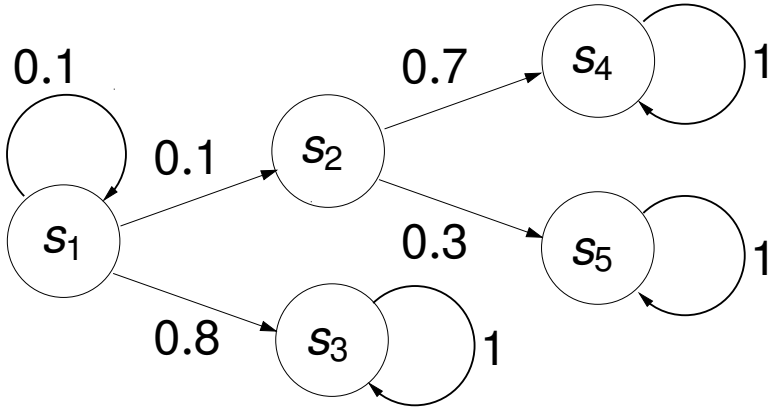
For SAT (the SMT-adaptation is analogous):

- consider τ-transitions in bounded model checking:

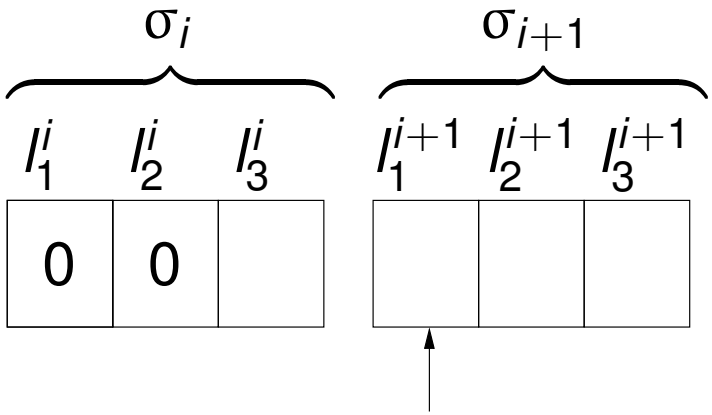
$$\mathit{Init}(\sigma_0) \wedge \left(\bigwedge_{i=0}^{k-1} \mathit{Trans}^{\tau}(\sigma_i, \sigma_{i+1}) \right) \wedge \mathit{Target}(\sigma_k)$$

- modify the SAT-solving to find more probable paths first by an adapted polarity heuristics

Searching for more probable paths



	l_1	l_2	l_3
s_1	0	0	0
s_2	0	0	1
s_3	0	1	0
s_4	0	1	1
s_5	1	0	0



1 Stochastic counterexamples

2 First CEBug example:
Path-based counterexamples via bounded model checking

3 Second CEBug example:
Counterexamples as minimal critical subsystems

Minimal critical subsystems

Minimal critical subsystems

- A subsystem is a part of a DTMC (defined by a subset of the states and the transitions).
- A subsystem is critical iff all paths that lie inside the subsystem and lead to a target state form a counterexample.
- A critical subsystem is state-minimal iff it is a critical subsystem with the smallest number of states.
- Analogously: transition-minimal

Minimal critical subsystems via SMT-solving

Variables:

- for each $s \in S$ a variable $x_s \in [0, 1] \subseteq \mathbb{Z}$ encodes if s belongs to the subsystem ot not, and
- for each $s \in S$ a variable $p_s \in [0, 1] \subseteq \mathbb{R}$ encodes the probability to reach a target state from s within the subsystem.

Minimal critical subsystems via SMT-solving

Variables:

- for each $s \in S$ a variable $x_s \in [0,1] \subseteq \mathbb{Z}$ encodes if s belongs to the subsystem ot not, and
- for each $s \in S$ a variable $p_s \in [0,1] \subseteq \mathbb{R}$ encodes the probability to reach a target state from s within the subsystem.

$\min \sum_{s \in S} x_s$ such that

- $p_{s_{init}} > \lambda$
- $\forall s \in T$:
  $\big((x_s = 0 \wedge p_s = 0) \oplus (x_s = 1 \wedge p_s = 1)\big)$
- $\forall s \in S \setminus T$:
  $\Big((x_s = 0 \wedge p_s = 0) \oplus \big(x_s = 1 \wedge p_s = \sum_{s' \in succ(s)} P(s,s') \cdot p_{s'}\big)\Big)$.

Minimal critical subsystems via MILP

$\min\left(-\frac{1}{2}p_{s_{init}} + \sum_{s \in S} x_s\right)$ such that

- $p_{s_{init}} > \lambda$
- $\forall s \in S \cap T : p_s = x_s$
- $\forall s \in S \setminus T : p_s \leq x_s \quad \wedge \quad p_s \leq \sum_{s' \in succ(s)} P(s, s') \cdot p_{s'}$

Model	without redundant constraints			optimal configuration		
	Z3	SCIP	CPLEX	Z3	SCIP	CPLEX
crowds2-2	0.45	**0.01**	0.02 (0.03)	0.18	**0.01**	**0.01 (0.03)**
crowds2-3	6.70	**0.16**	1.33 (0.28)	4.82	0.12	**0.06 (0.11)**
crowds2-4	127.18	0.47	**0.30 (0.24)**	23.72	**0.30**	**0.30 (0.24)**
crowds2-5		0.90	**0.56 (0.45)**	152.28	0.60	**0.56 (0.24)**
crowds3-3		0.64	**0.49 (0.33)**		**0.35**	0.38 (0.30)
crowds3-4		**4.29**	5.53 (2.07)		1.45	**0.89 (0.58)**
crowds3-5		23.49	**6.66 (2.77)**		5.58	**1.51 (0.87)**
crowds5-4		743.84	**14.23 (5.07)**		13.28	**12.51 (4.89)**
crowds5-6			**302.03 (38.39)**		1947.46	**100.26 (23.52)**
crowds5-8						**1000.79 (145.84)**
leader3-2	**0.06**	0.07	0.62 (0.22)	0.05	**0.01**	0.21 (0.13)
leader3-3		91.89	**0.43 (0.22)**		0.06	**0.02 (0.06)**
leader3-4		2346.59	**0.70 (0.36)**		0.40	**0.07 (0.09)**
leader3-5		**0.44**	1.11 (0.55)		**0.05**	0.14 (0.18)
leader4-2	3.57	**0.23**	0.45 (0.21)	1.38	**0.07**	0.24 (0.17)
leader4-3		1390.79	**22.33 (3.38)**		**0.21**	0.49 (0.37)
leader4-4					**1.49**	1.88 (1.21)
leader4-5					**1.15**	4.06 (2.80)
leader4-6						**8.70 (5.92)**