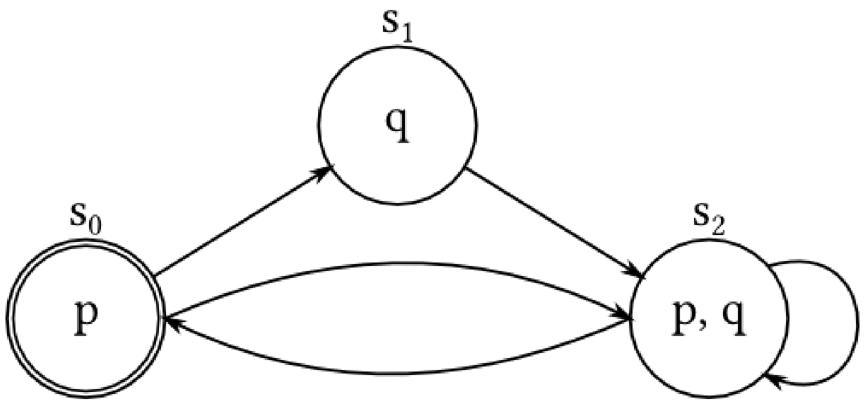
Bounded Model Checking



$$G(\neg(q \wedge \neg p))$$

Sebastian Henningsen,

Contents

Motivation

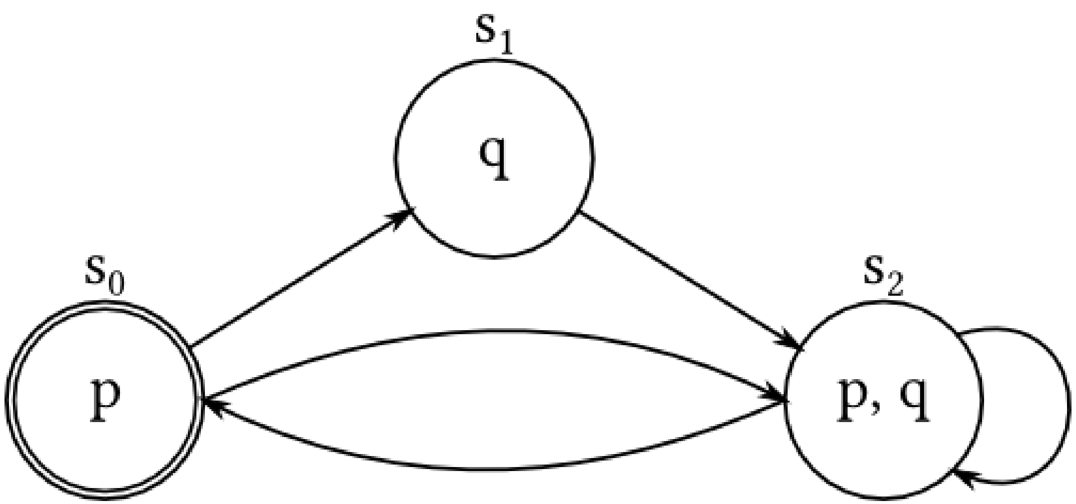
- Model Checking problems in general are hard
- "After initialization, a safety property holds"
- Previous™ algorithms designed for *verification*, not *falsification*
- Falsification means: Bug hunting
- Central Assumption: Errors occur in the first k steps of a transition system
  - Context switches, program steps

Bounded Model Checking

What is BMC?

- Given a transition system...
- ... and a safety property.
- Unroll the transition relation
- Formulate a SAT formula
  feed it into a solver
  and obtain trace leading to the bug

Towards SAT



$$s_0 \wedge \quad \wedge \quad (\llbracket\varphi\rrbracket_k^0 \quad \vee \quad \bigvee_{l=0}^{k} \lambda_l \wedge {}_l\llbracket\varphi\rrbracket_k^0)$$

$$T(s_0, s_1) \wedge$$

$$T(s_1, s_2) \wedge$$

$$\ldots$$

unroll this k-times

check the condition (directly)

check the condition (through loops)

Towards SAT

Transition relation:

$$(s_0 \wedge s_1' \vee s_0 \wedge s_2') \vee (s_1 \wedge s_2') \vee (s_2 \wedge s_0' \vee s_2 \wedge s_2')$$

Gets unrolled to: $\mathcal{T} =$

$$\begin{gathered}
(s_0 \wedge s_1' \vee s_0 \wedge s_2') \vee (s_1 \wedge s_2') \vee (s_2 \wedge s_0' \vee s_2 \wedge s_2') \\
\wedge \\
(s_0' \wedge s_1'' \vee s_0' \wedge s_2'') \vee (s_1' \wedge s_2'') \vee (s_2' \wedge s_0'' \vee s_2' \wedge s_2'') \\
\wedge \\
(s_0'' \wedge s_1''' \vee s_0'' \wedge s_2''') \vee (s_1'' \wedge s_2''') \vee (s_2'' \wedge s_0''' \vee s_2'' \wedge s_2''') \\
\cdots
\end{gathered}$$

Satisfiable: $\mathcal{T} \wedge s_0 \wedge s_1' \wedge s_2''$

Not satisfiable: $\mathcal{T} \wedge s_0 \wedge s_2' \wedge s_1''$

Towards SAT

Checking the condition $q \wedge \neg p$ for every unfolding step:

$$holds(q \wedge \neg p) \vee$$
$$holds'(q \wedge \neg p) \vee$$
$$holds''(q \wedge \neg p)$$

translates to:

$$s_1 \vee s_1' \vee s_1''$$

Finished formula:

$$\boxed{\mathcal{I} \wedge \mathcal{T} \wedge (s_1 \vee s_1' \vee s_1'')}$$

Satisfied by: s_0, s_1', s_2''

Contents

TCBMC

- Rabinovitz, Grumberg
- Show absence of bugs in C-programs
- Source code describes transition system
- Safety property given by `assert` statements
- SAT solver (modulo theories) searches counter examples.

TCBMC – One Thread

Original program:

```
x = x + y;
if(x != 1)
  x = 2;
else
  x++;

assert(x <= 3);
```

In single-assignment form:

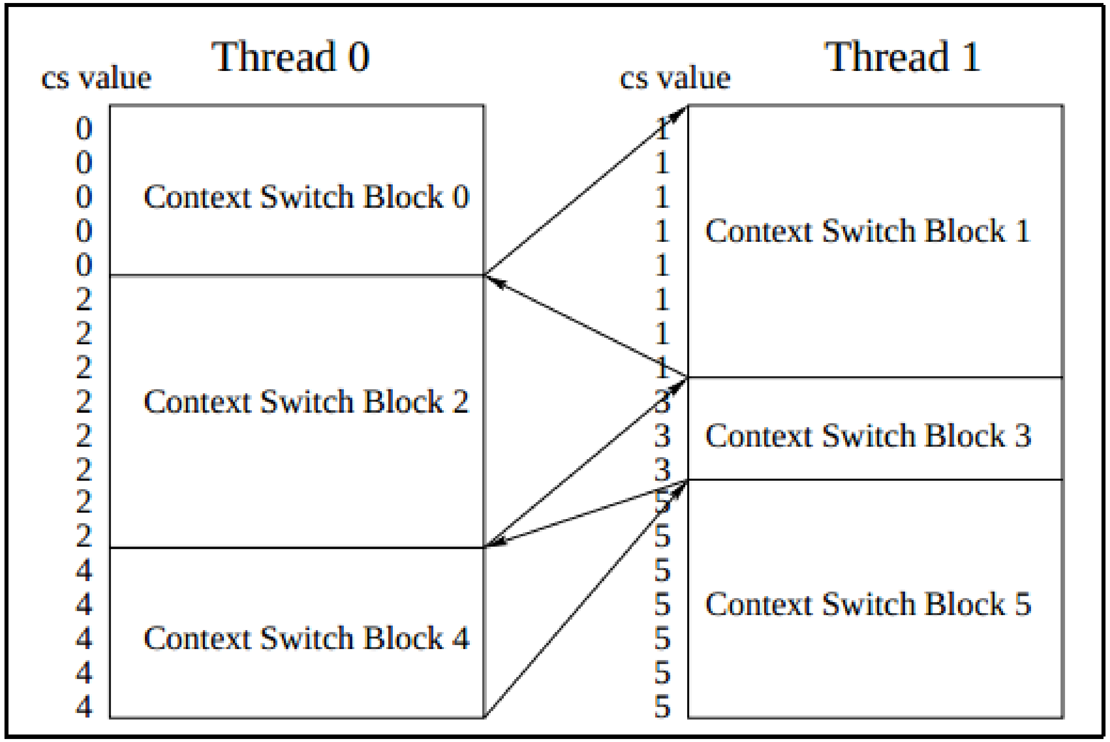
```
x1 = x0 + y0;
if(x1 != 1)
  x2 = 2;
else
  x3 = x1 + 1;

x4 = (x1 != 1) ? x2 : x3

assert(x4 <= 3);
```

Yields equations:

$$
x_1 = x_0 + y_0 \quad \wedge \quad x_2 = 2 \wedge
$$
$$
x_3 = x_1 + 1 \quad \wedge \quad x_4 = (x_1 \neq 1) ? x_2 : x_3
$$
$$
\wedge\, x_4 > 3
$$

TCBMC – More Threads

- Instrument every thread as shown
- Introduce context switch blocks
- Copy global variables



Rabinovitz, Grumberg: Context switch blocks in TCBMC

Contents

- Introduction / Bounded Model-Checking
- Model Checking C-programs
- Interpolation
- Conclusion

Interpolation

- Originally from model theory (Craig 1957)
- Extension of BMC which allows the verification of certain properties instead of plain falsification
- Various applications beyond BMC: Predicate Abstraction/Refinement, theorem proving, …

Definition of Interpolants

Assume $A \to B$ holds in some logic. An interpolant (sometimes Craig interpolant) is a formula I such that:

1. $A \to I$ and $I \to B$ are valid and
2. every non-logical symbol in I occurs in both A and B

where non-logical symbols are variables, free function symbols etc.

Definition is used "in reverse": If $A \wedge B$ are unsat. then there is I s.t. $A \to I$ is valid and $B \to I$ is unsat.

Example

Consider $\underbrace{(p \land q)}_{\text{A}} \to \underbrace{(q \lor r)}_{\text{B}}$ which is valid

Then $I := q$ is an interpolant for $A \text{ and } B$

Verification:

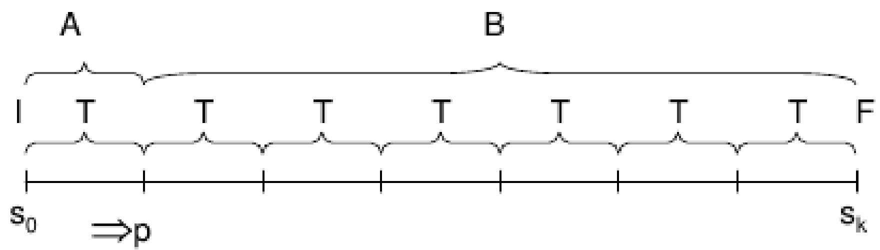
1. $(p \land q) \to q)$ holds
2. $q \to (q \lor r)$ holds
3. q is the only variable in both $A \text{ and } B$

In practice interpolants are not guessed but computed from resolution proofs: Given a proof of unsatisfiability of $A \land B$ an interpolant for $A \to \neg B$ can be derived in linear time

What's the point?

Recall BMC formula from above:

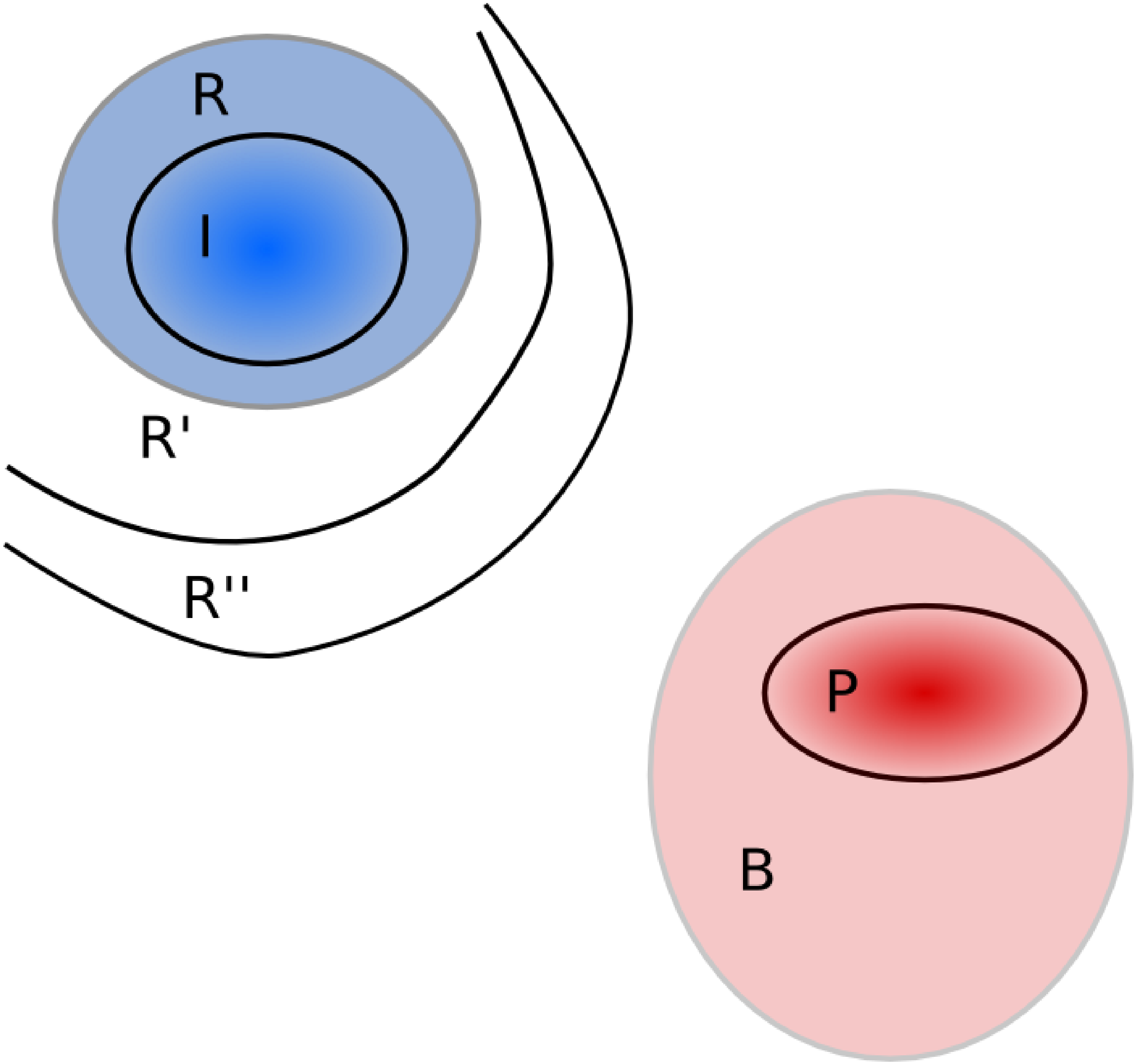
$$BMC_j^k = I(s_0) \wedge \underbrace{\left(\bigwedge_{0 \leq i \leq k} T(s_i, s_{i+1}) \right)}_{\text{Unfold transition relation } k \text{ times}} \wedge \underbrace{\left(\bigvee_{j \leq i \leq k} F(s_i) \right)}_{\text{Reachable in } j \text{ to } k \text{ steps}}$$



Partition the formula into two parts:

Check if $A \wedge B$ is sat. with resolution. If yes $\rightsquigarrow$ counter example, Else $\rightsquigarrow$ compute interpolant

Algorithmic Idea


R
I
R'
R''
P
B

Contents

Conclusion

Key take aways:

- Fundamental ideas of bounded model checking:
  - Finite unrolling of transition relation
  - Check for counter example
  - If there is any, obtain corresponding trace
- Possible application domains: Single and multi-threaded C programs
- Basic idea of interpolants and their role as an extension to BMC

References

- A. Biere: Bounded Model Checking. In: Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications.
- I.Rabinovitz, O. Grumberg: Bounded model checking of concurrent programs.
- K McMillan: Interpolation and sat-based model checking. In: Computer Aided Verification, 15th International Conference