Concolic Unbounded-Thread Reachability via Loop Summaries

Peizun Liu and Thomas Wahl

Northeastern University, Boston, USA

ICFEM 2016, Tokyo, Japan

November 18, 2016

Outline

Motivation

Setting the Stage

Concolic Reachability Analysis

Experimental Evaluation

Conclusion

Outline

Motivation

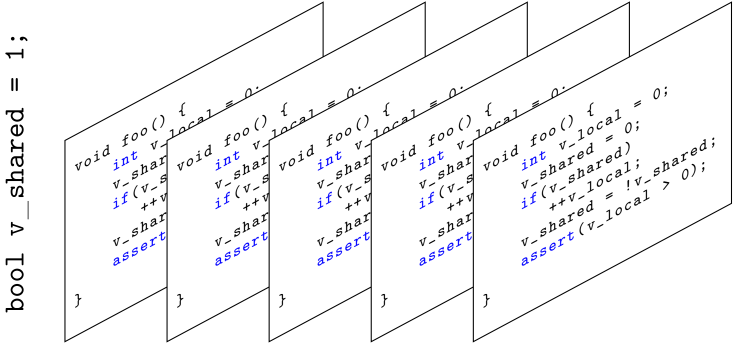
Target: *unbounded-thread shared-memory programs* where each thread executes a non-recursive procedure

```
bool v_shared = 1;

void foo() {
     int v_local = 0;
     v_shared = 0;
     if(v_shared)
        ++v_local;
     v_shared = !v_shared;
➠    assert(v_local > 0);
}

int main() {
     while(...)
        create_thread(&foo);
}
```

Goal: assertion checking $\Rightarrow$ program state reachability analysis

Motivation

Target: *unbounded-thread shared-memory programs* where each thread executes a non-recursive procedure



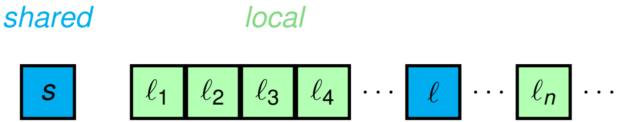
Goal: assertion checking ⇒ program state reachability analysis

Problem Statement

Program State Reachability

Given: a program state (s, ℓ), with shared component s and local component ℓ

Task: check if there exists a reachable global state of the form:



Classical Solutions

Reachability of $(s, \ell) \Rightarrow$ coverability problem

- Karp-Miller Procedure [Karp & Miller, 1969]
- Backward Search [Abdulla et al., 1996]

Questions

- **explicit-state** exploration
  ☞ can symbolic approach speed it up?
- **traversing loops** whenever seeing them
  ☞ inducing huge number of preimages or postimages

Our Approach

Concolic unbounded-thread reachability analysis

... based on Abdulla's Backward Search (BWS).

via:

- pathwise analysis
  - ☞ slice an entire concurrent system into distinct paths
- reachability $\Rightarrow$ satisfiablity of Presburger formulas
  - ☞ summarize paths without nested loops symbolically

Result: *accelerate* explicit-state BWS via adding symbolic flavor

Outline

Motivation

Setting the Stage

Concolic Reachability Analysis

Experimental Evaluation

Conclusion

Thread-Transition Diagrams (TTD)

Convert multi-threaded programs to thread-transition diagrams:

Step 1: Programs to Boolean Programs

```
int x = 1;

int main() {
    int y = 0;

    x = 0;
    if(x)
      y = 1;
    x = !x;
    assert(!y);
    return 0;
}
```

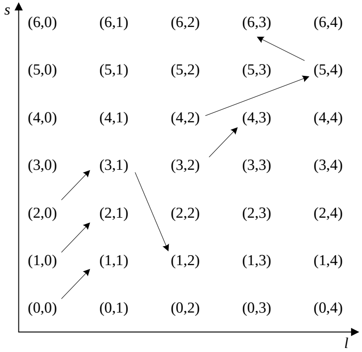
```
decl s := 0;
main() {
      decl l := 0;
    1: s := 0;
    2: goto 3,6;
    3: assume(s);
    4: l := 1;
    5: goto 7;
    6: assume(!s);
    7: s := !s;
    8: assert(!l);
}
```

Thread-Transition Diagrams (TTD)

Convert multi-threaded programs to thread-transition diagrams:

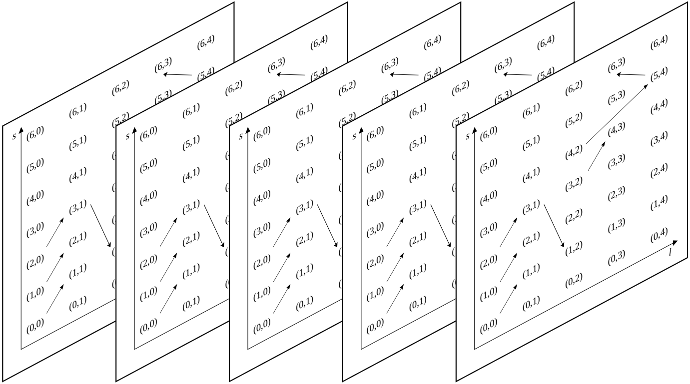
Step 2: Boolean Programs to TTDs

```
decl s := 0;
main() {
     decl l := 0;
   1: s := 0;
   2: goto 3,6;
   3: assume(s);
   4: l := 1;
   5: goto 7;
   6: assume(!s);
   7: s := !s;
   8: assert(!l);
}
```



From TTD to WQOS

TTD gives rise to well quasi-ordered system (WQOS)



From TTD to WQOS

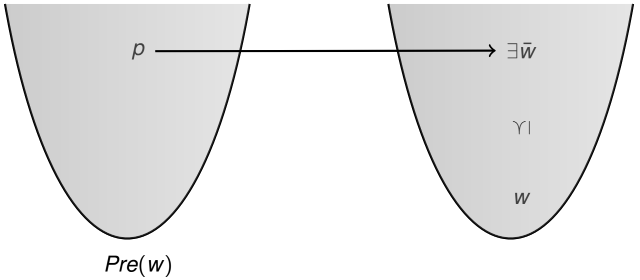
TTD gives rise to well quasi-ordered system (WQOS)

In our case: WQO is the *covers* relation:

$$(s, \bar{\ell}_1, \ldots, \bar{\ell}_{\bar{n}}) \succeq (s, \ell_1, \ldots, \ell_n)$$

whenever $\quad \mathit{multiset}\{\bar{\ell}_1, \ldots, \bar{\ell}_{\bar{n}}\} \supseteq \mathit{multiset}\{\ell_1, \ldots, \ell_n\}$.

Backward Search [Abdulla et al., 1996]



$$\text{CovPre}(w) \;=\; \min\{p \mid \exists\, \bar{w} :\; p \rightarrow \bar{w} \wedge \bar{w} \succeq w\}$$

Outline

Our Approach: Overview

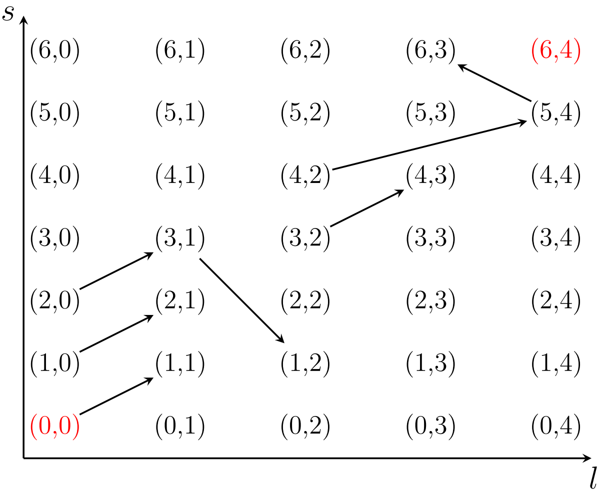
Concolic unbounded-thread reachability analysis

- Reduce reachability to pathwise reachability;
- **Concrete flavor:** Perform explicit-state BWS across complex paths;
- **Symbolic flavor:** Encode reachability across simple paths symbolically.

A single-threaded Abstraction

Original TTD $\mathcal{P}$

$t_I = (0, 0)$
$t_F = (6, 4)$

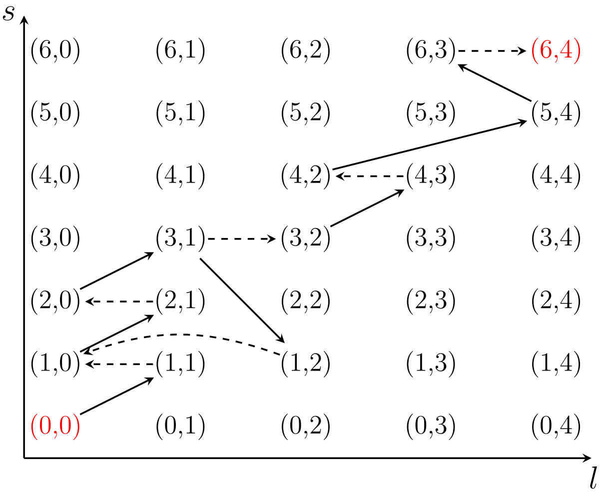


A single-threaded Abstraction

Expanded TTD $\mathcal{P}^+$

$t_I = (0, 0)$
$t_F = (6, 4)$

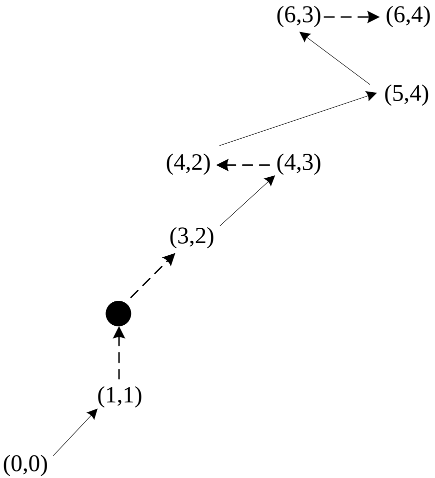
$\succeq$



A single-threaded Abstraction

TTD Quotient Graph $\overline{\mathcal{P}}$

$t_I = (0, 0)$
$t_F = (6, 4)$



A single-threaded Abstraction

Soundness of Abstraction

If thread state t_F is reachable in $\mathcal{P}_\infty$, then t_F is also reachable in $\overline{\mathcal{P}}$.

- $\mathcal{P}_\infty$ represents $\mathcal{P}$ running by unbounded number of threads.
- $\overline{\mathcal{P}}$ simulates $\mathcal{P}_\infty$ in one single thread.

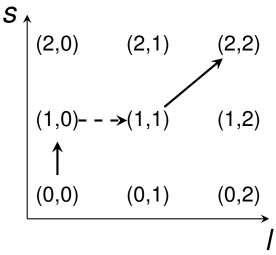
Pathwise Analysis: Overview

for each path $\overline{\sigma}$ in $\overline{\mathcal{P}}$

- if $\overline{\sigma}$ contains nested loops, call BWS($\overline{\sigma}$)
- otherwise, reduce to Presburger formula ϕ:

t_F is reachable along $\overline{\sigma}$ *iff* ϕ is satisfiable

Symbolic Summaries for Loop-free Paths



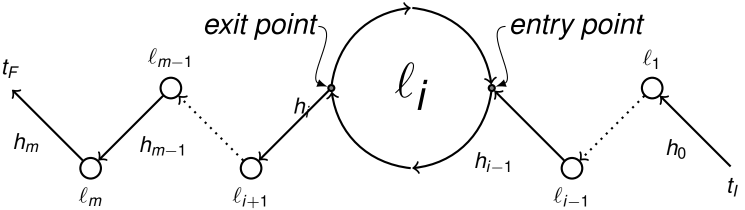
Summary functions for local states $l = 0, 1, 2$:

$$
\begin{aligned}
\Sigma_0(n_0) &= n_0 \ominus 1 + 1 - 1 + 1 = n_0 \ominus 1 + 1 \\
\Sigma_1(n_1) &= n_1 + 1 \\
\Sigma_2(n_2) &= n_2 - 1
\end{aligned}
$$

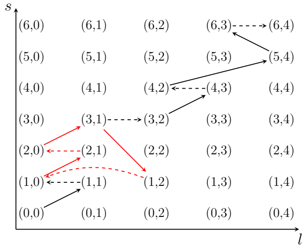
Examples:

$$\Sigma_0(0) = 1,\ \Sigma_0(1) = 1,\ \Sigma_1(0) = 1,\ \Sigma_2(1) = 0 .$$

Symbolic Summaries for Simple Loops



Symbolic Summaries for Simple Loops



Summary functions for simple loop:

$$\Sigma_l(n_l) = n_l \oplus_{b_l} \delta_l \oplus_{b_l} (k-1) \cdot \delta_l \,. \tag{1}$$

Examples:

$$\begin{array}{lllllllll} b_0 & = & 2, & b_1 & = & 1, & b_2 & = & 0 \\ \delta_0 & = & 2, & \delta_1 & = & -1, & \delta_2 & = & -1 \end{array}$$

$$\begin{array}{lllllllll} \Sigma_0(n_0) & = & n_0 & \oplus_2 & 2 & \oplus_2 & (k-1) & \cdot & 2 \\ \Sigma_1(n_1) & = & n_1 & \oplus_1 & -1 & \oplus_1 & (k-1) & \cdot & -1 \\ \Sigma_2(n_2) & = & n_2 & \oplus_0 & -1 & \oplus_0 & (k-1) & \cdot & -1 \end{array}$$

The Example

Presburger formula for the previous example: $t_F = (6, 4)$

$$
\begin{array}{llllllllllllll}
n_0: & 0 & \oplus_0 & 0 & \oplus_2 & 2 & \oplus_2 & (k-1) & \cdot & 2 & \oplus_3 & 3 & \geq & 1 \\
n_1: & 0 & \oplus_1 & 0 & \oplus_1 & -1 & \oplus_1 & (k-1) & \cdot & -1 & \oplus_0 & -3 & = & 0 \\
n_2: & 0 & \oplus_2 & 2 & \oplus_0 & -1 & \oplus_0 & (k-1) & \cdot & -1 & \oplus_0 & 0 & = & 0 \\
n_3: & 0 & \oplus_0 & -2 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0 \\
n_4: & 1 & \oplus_1 & 0 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0
\end{array}
$$

The Example

Presburger formula for the previous example: $t_F = (6, 4)$

$$\begin{array}{lrcrcrcccrcrcr}
n_0: & 0 & \oplus_0 & 0 & \oplus_2 & 2 & \oplus_2 & (k-1) & \cdot & 2 & \oplus_3 & 3 & \geq & 1 \\
n_1: & 0 & \oplus_1 & 0 & \oplus_1 & -1 & \oplus_1 & (k-1) & \cdot & -1 & \oplus_0 & -3 & = & 0 \\
n_2: & 0 & \oplus_2 & 2 & \oplus_0 & -1 & \oplus_0 & (k-1) & \cdot & -1 & \oplus_0 & 0 & = & 0 \\
n_3: & 0 & \oplus_0 & -2 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0 \\
n_4: & \underbrace{1 & \oplus_1 & 0 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0}_{=1} & = & 0
\end{array}$$

The Example

Presburger formula for the previous example: $t_F = (6, 3)$

$$
\begin{array}{lrcrcrccrcrcr}
n_0: & 0 & \oplus_0 & 0 & \oplus_2 & 2 & \oplus_2 & (k-1) & \cdot & 2 & \oplus_3 & 3 & \geq & 1 \\
n_1: & 0 & \oplus_1 & 0 & \oplus_1 & -1 & \oplus_1 & (k-1) & \cdot & -1 & \oplus_0 & -3 & = & 0 \\
n_2: & 0 & \oplus_2 & 2 & \oplus_0 & -1 & \oplus_0 & (k-1) & \cdot & -1 & \oplus_0 & 0 & = & 0 \\
n_3: & 1 & \oplus_0 & -2 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0 \\
n_4: & 0 & \oplus_1 & 0 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0
\end{array}
$$

The Example

Presburger formula for the previous example: $t_F = (6, 3)$

$$
\begin{array}{llllllllllllll}
n_0: & 0 & \oplus_0 & 0 & \oplus_2 & 2 & \oplus_2 & (k-1) & \cdot & 2 & \oplus_3 & 3 & \geq & 1 \\
n_1: & 0 & \oplus_1 & 0 & \oplus_1 & -1 & \oplus_1 & (k-1) & \cdot & -1 & \oplus_0 & -3 & = & 0 \\
n_2: & 0 & \oplus_2 & 2 & \oplus_0 & -1 & \oplus_0 & (k-1) & \cdot & -1 & \oplus_0 & 0 & = & 0 \\
n_3: & 1 & \oplus_0 & -2 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0 \\
n_4: & 0 & \oplus_1 & 0 & \oplus_0 & 0 & \oplus_0 & (k-1) & \cdot & 0 & \oplus_0 & 0 & = & 0
\end{array}
$$

Satisfiable Assignment: $k = 2$, unrolling the loop twice

Outline

Experimental Evaluation

Tool

We implement our approach in a reachability checker named CUTR[1,2].
It uses Z3 as the Presburger solver.

Benchmark

124 BPs, 56 of which are safe

BP	min.	max.
$\lvert S \rvert$	5	257
$\lvert L \rvert$	14	4097
$\lvert R \rvert$	18	20608

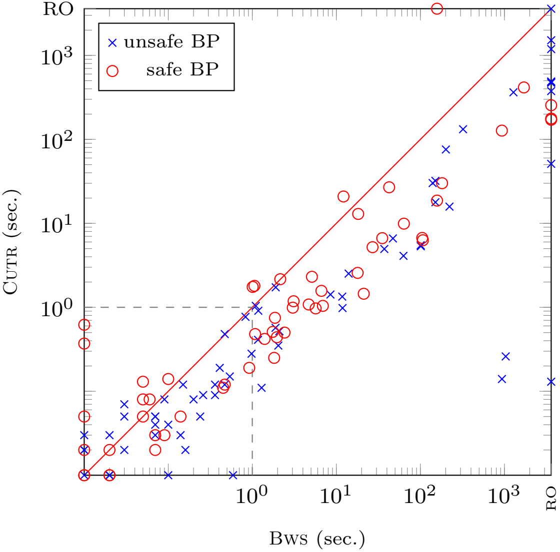
---

[1] CUTR "=" **C**oncolic **U**nbounded-**T**hread **R**eachability analysis.
[2] Download me ☺:

- benchmark & executable:
  http://www.ccs.neu.edu/home/lpzun/cutr
- source code:
  https://github.com/lpzun/cutr

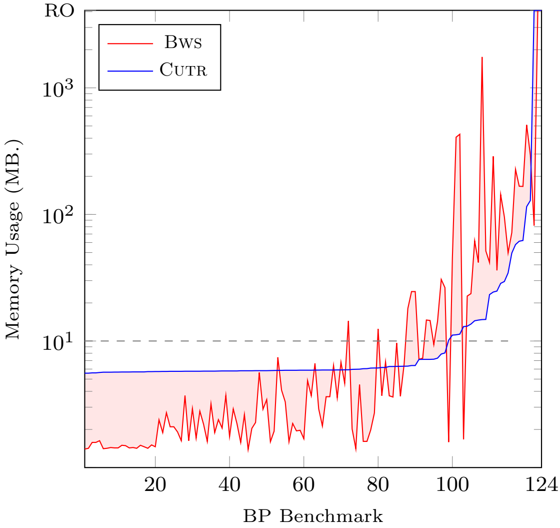
Experimental Evaluation

Evaluation on Time



Experimental Evaluation

Evaluation on Memory



Outline

Conclusion

Conclusion

Our approach

- slices a concurrent system into paths — **pathwise analysis**,
- reduces reachability to satisfiability — **symbolic analysis**,
- successfully accelerates a widely-applicable BWS.

Future work

- symbolically summarize paths with nested loops;
- apply pathwise analysis and loop summaries to other infinite-state search algorithms, like [A. Kaiser, 2012].

Thank You

References

R. M. Karp and R. E. Miller, "Parallel program schemata," *J. Comput. Syst. Sci.*, 1969.

P. A. Abdulla, "Well (and better) quasi-ordered transition systems," *Bulletin of Symbolic Logic*, 2010.

A. Kaiser, D. Kroening, and T. Wahl, "Efficient coverability analysis by proof minimization," in *CONCUR*, 2012.