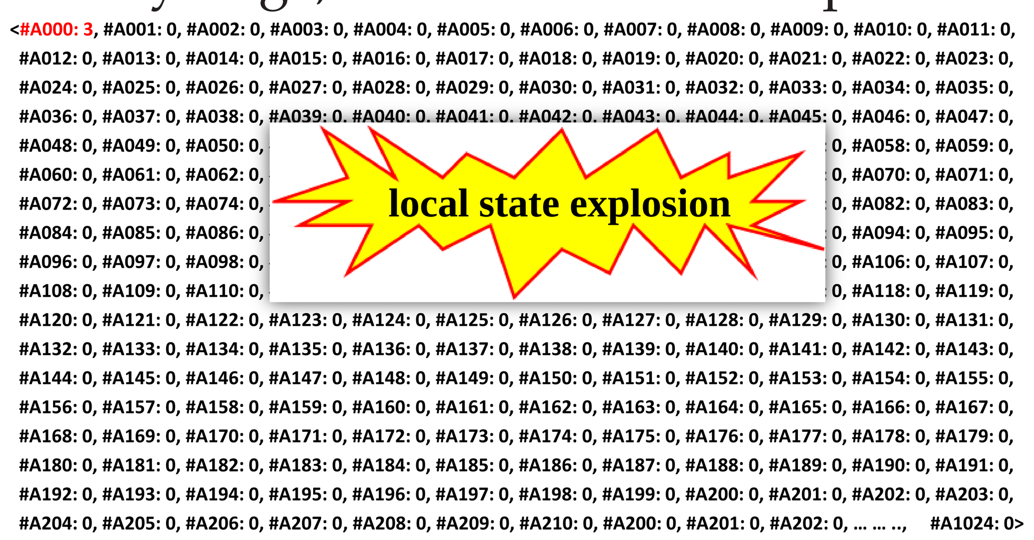
Boolean Program Exploration Using an ALL-SAT Solver Backend

Peizun Liu and Thomas Wahl, Northeastern University, Boston, USA

SAT/SMT Summer School, Semmering, Austria, July 10-12, 2014

Problem

Program state reachability analysis for replicated Boolean programs run by an unbounded number of threads is decidable in principle via a reduction of the Boolean program families to *well-structured transition systems* (WSTS). The obtained transition systems would, however, in general be intractably large, due to local state explosion:



Preliminaries

Notation: $\mathcal{B}$ = Boolean program, pc = program counter, S = set of shared states, $L = PC \times C$ = set of local states, consisting of program counters PC and local variable valuations C.

BWRA operates on WSTS [2]. A WSTS is a transition system equipped with a well quasi-ordering $\preceq$ on its states that satisfy a monotonicity property. $\mathcal{B}$ induces a WSTS, with $\preceq$ defined as follows:

$$\langle s, \{(\ell_1, n_1), \ldots, (\ell_k, n_k)\}\rangle \preceq \langle s', \{(\ell_1, n_1'), \ldots, (\ell_k, n_k'), \ldots)\}\rangle$$

if $s = s'$ and $\forall 1 \le i \le k$: $n_i \le n_i'$. We say r covers τ if $\tau \preceq r$.

Definition. *Let* $\uparrow \tau := \{r \mid \tau \preceq r\}$. *Then*

$$\mathit{CovPre}(\tau') := \{\tau \mid \exists\, \tau \longrightarrow r,\ r \in \uparrow \tau'\} \quad \text{and}$$

$$\mathit{C\text{-}Pre}(\tau') := \min\{\tau : \tau \in \mathit{CovPre}(\tau')\}.$$

References

[1] G. Basler, M. Mazzucchi, T. Wahl, and D. Kroening, "Context-aware counter abstraction," *Form. Methods Syst. Des.*, vol. 36, no. 3, pp. 223–245, Sep. 2010.

[2] P. A. Abdulla, "Well (and better) quasi-ordered transition systems," *Bulletin of Symbolic Logic*, vol. 16, no. 4, pp. 457–515, 2010.

Contributions

In this work, we extend the *context-aware* idea for Boolean programs run by a fixed, finite number of threads [1] to families with *unbounded thread counts*, based on Backward Reachability Analysis (BWRA) [2].

Our main contributions include:

1. performing BWRA on-the-fly by operating directly on Boolean programs;
2. avoiding local state explosion with the aid of on-the-fly exploration and efficient ALL-SAT solvers;
3. optimizations to limit the size of obtained covering pre-images.

Example

C Program

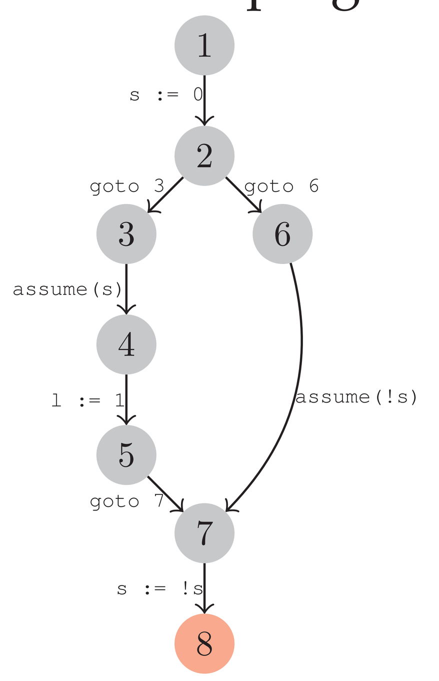
```
int x = 1;
int main(){
   int y = 0;
   x = 0;
   if(x)
       y = 1;
   x = !x;
   assert(!y);
   return 0;
}
```

$\Rightarrow$ Predicate Abstraction

Boolean Program

```
decl s := 1;
void main() begin
  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);
end
```

Control Flow Graph of Boolean program



A path explored by on-the-fly BWRA

$s = 0 \wedge \ell_1 = (1,8)$

$s = 1 \wedge \ell_1 = (1,7)$

$s = 1 \wedge \ell_1 = (1,5)$

$s = 1 \wedge \ell_1 = (\star,4)$

$s = 1 \wedge \ell_1 = (\star,3)$

$s = 1 \wedge \ell_1 = (\star,2)$

$s = 1 \wedge \ell_1 = (\star,2) \wedge \ell_2 = (0,8)$

$s = 0 \wedge \ell_1 = (\star,2) \wedge \ell_2 = (0,7)$

$s = 0 \wedge \ell_1 = (\star,2) \wedge \ell_2 = (0,6)$

$s = 0 \wedge \ell_1 = (\star,2) \wedge \ell_2 = (0,2)$

$s = \star \wedge \ell_1 = (\star,1) \wedge \ell_2 = (0,1)$

$\star$: nondeterminism; local state $\ell = (1,8)$: $l = 1 \wedge pc = 8$

On-the-fly Backward Exploration

Idea: compute C-Pre(τ') based on *control flow graph* (CFG) and *weakest precondition* (WP) propagation.

1. CFG $G = (V, E)$, with V = set of program locations, and E = set of execution flows.
2. WP defined as $\mathrm{WP}_{e.stmt}(s, \ell, s', \ell')$, where $e.stmt$ is a statement associated with edge $e \in E$. It is encoded as a CNF formula, where s, ℓ are free variables, and then input into an ALL-SAT solver.

Algorithm On-the-fly Bw Exploration

Input: $\mathcal{B}$: a Boolean program with the set of initial thread states I; $\mathcal{T}_{fin}$: the set of target thread states; G=(V, E): a CFG constructed from $\mathcal{B}$

Output: Is $\uparrow \mathcal{T}_{fin}$ reachable?

$\Phi := \mathcal{T}_{fin}$ ▷ the set of unexplored states
$\Psi := \emptyset$ ▷ the set of explored states
$\Omega :=$ Candidate-Local-States$(\mathcal{B})$
while $\Phi \neq \emptyset$
  remove $\tau' = \langle s', Z'\rangle$, with $Z' = \{(\ell_1', n_1'), \ldots, (\ell_k', n_k')\}$, from min Φ
  if $\tau' \in \mathcal{T}_{init}$ **then**
    return true
  else if $\Psi \cap \downarrow \tau' \neq \emptyset$ **then**
    discard τ'
  else
    C-Pre(τ') := Cov-Predecessors(τ')
    $\Phi := \Phi \cup$ C-Pre(τ')
    $\Psi := \Psi \setminus (\uparrow \tau') \cup \{\tau'\}$
return false

Procedure Cov-Predecessors(τ')

C-Pre$(\tau) := \emptyset$
for each $i \in \{1, \ldots, k\}$ ▷ direct predecessors
  for each $e \in E$ s.t. $target(e) = \ell_i'.pc$
    for each (s, ℓ) s.t. $\mathrm{WP}_{e.stmt}(s, \ell, s', \ell_i')$
      $\tau := \langle s,$ Update-Counters$(\ell, \ell_i', Z')\rangle$
      insert τ into C-Pre(τ)
for each (s, ℓ) s.t. $\exists \ell' \notin \{\ell_1', \ldots, \ell_k'\}$: $e := (\ell.pc, \ell'.pc) \in E \wedge \mathrm{WP}_{e.stmt}(s, \ell, s', \ell')$
  $\tau := \langle s,$ Update-Counters$(\ell, null, Z')\rangle$
  insert τ into C-Pre(τ)
return C-Pre(τ)

Procedure Update-Counters(ℓ, ℓ', Z')

if $\exists\, n$: $(\ell', n) \in Z'$ **then**
  $Z := Z' \setminus \{(\ell', n)\} \cup (n > 1 ? \{(\ell', n-1)\} : \emptyset)$
if $\exists\, n$: $(\ell, n) \in Z'$ **then**
  $Z := Z \setminus \{(\ell, n)\} \cup \{(\ell, n+1)\}$
else
  $Z := Z \cup \{(\ell, 1)\}$
return Z

$\mathcal{T}_{init} := \{\langle s, \{(\ell_1, n_1), \ldots, (\ell_m, n_m)\}\rangle \wedge \forall 1 \le i \le m \text{ s.t. } (s, \ell_i) \in I\}$

Local States Reduction: Candidate-Local-States$(\mathcal{B})$

We compute the set of candidate local states Ω with the following two optimizations:

1. *Restricting PC*: Consider only statements that change the shared state.
2. *Local Configuration Reachability Analysis*: a local configuration $c \in C$ is a valuation of the local variables. Configuration c is *reachable* if there exists a reachable local state ℓ containing it. If c is known a priori to be unreachable, then all ℓ's containing c can be safely removed from L.

Detecting reachability of c is a model checking problem. However, we do not need know the exact set of reachable c's: an over-approximation suffices.

Future Work

1. Extend to Boolean broadcast programs.
2. **Symbolic** on-the-fly backward exploration

Funding

Work supported by NSF grant no. 1253331.