SimpleCAR: An Efficient Bug-Finding Tool Based On Approximate Reachability

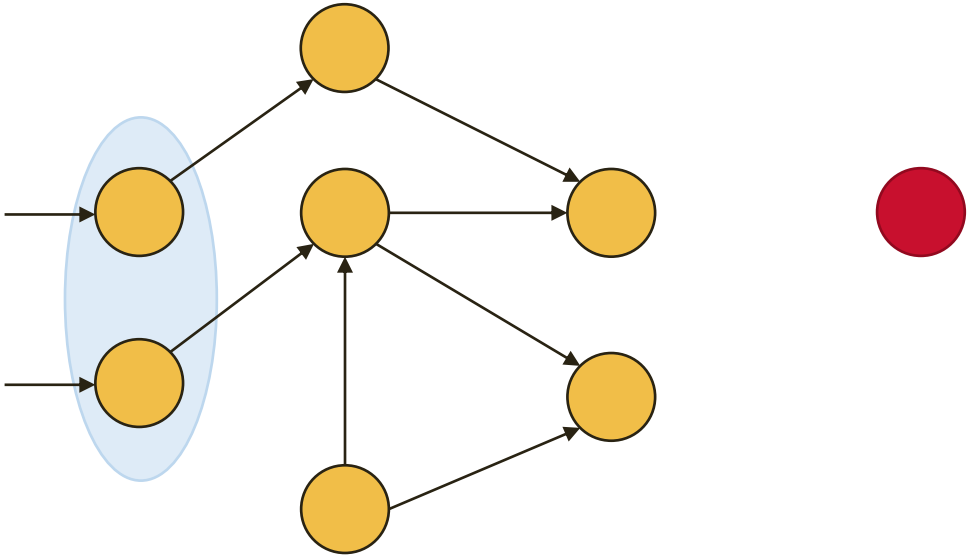
Jianwen Li, **Rohit Dureja**,
Geguang Pu, Kristin Y. Rozier,
Moshe Y. Vardi

July 16, 2018

Standard Reachability Analysis

Model $M = (V, I, T)$

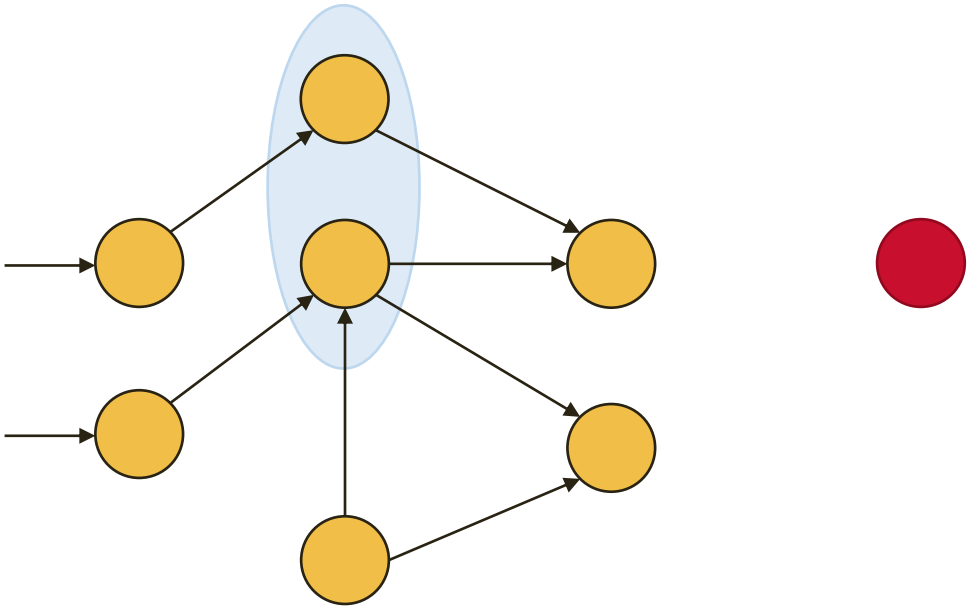
Safety Property P



Standard Reachability Analysis

Model $M = (V, I, T)$

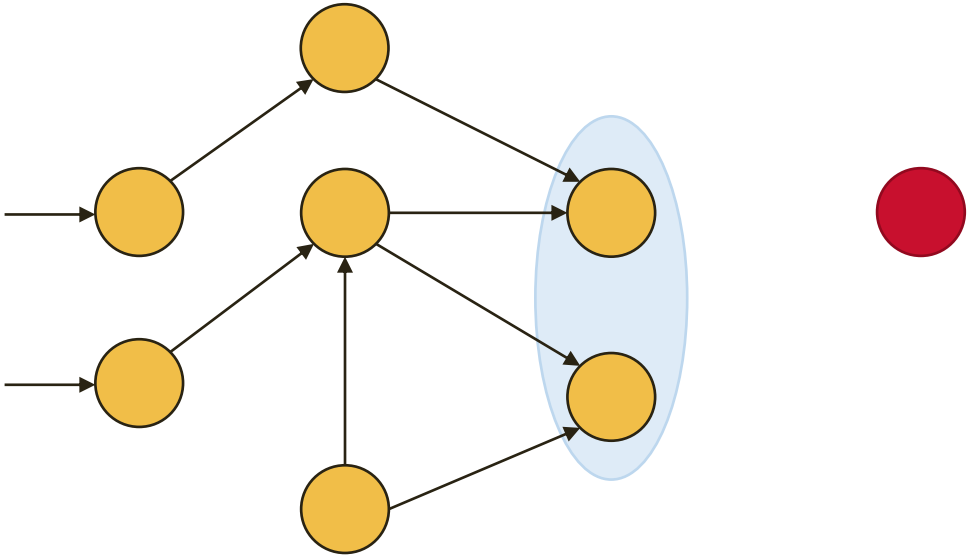
Safety Property P



Standard Reachability Analysis

Model $M = (V, I, T)$

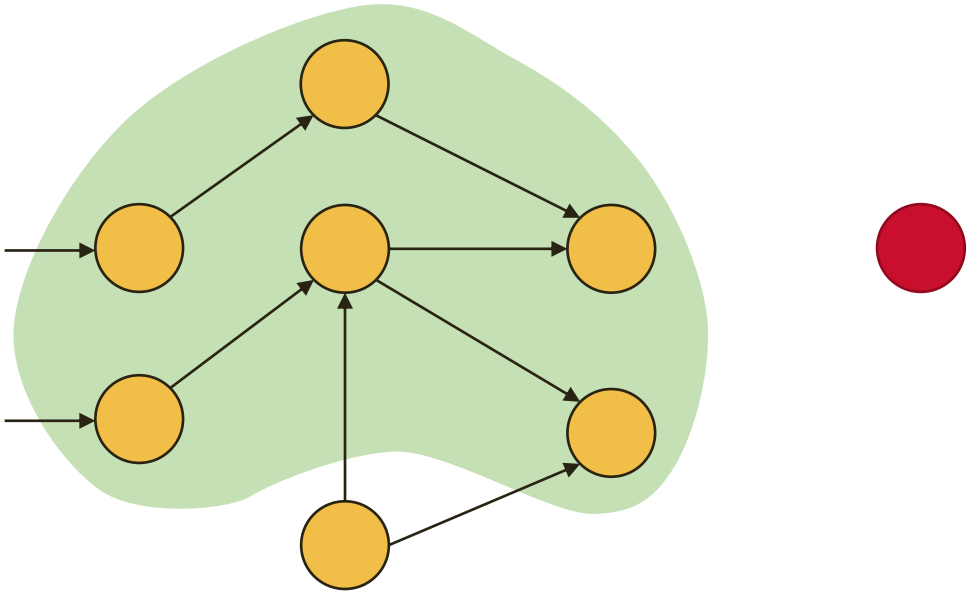
Safety Property P



Standard Reachability Analysis

Model $M = (V, I, T)$

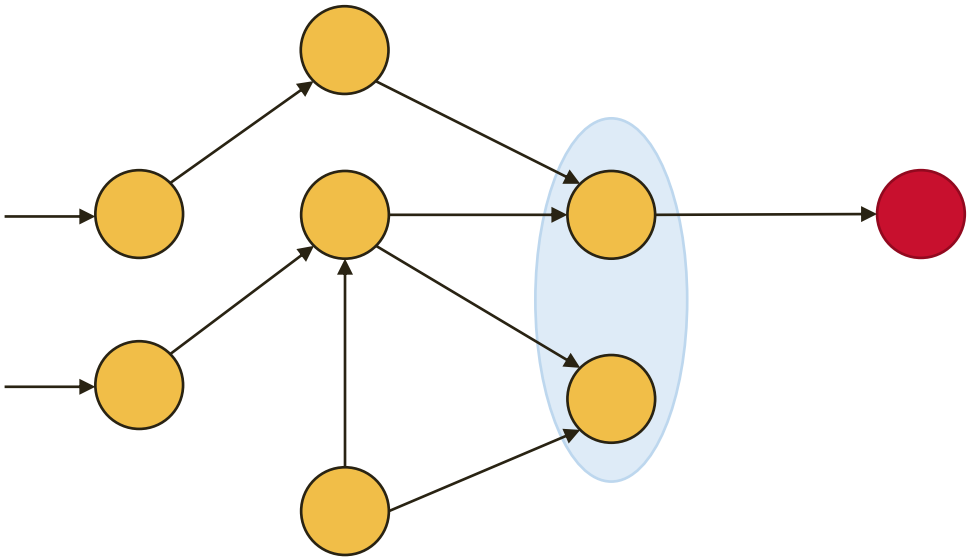
Safety Property P



Standard Reachability Analysis

Model $M = (V, I, T)$

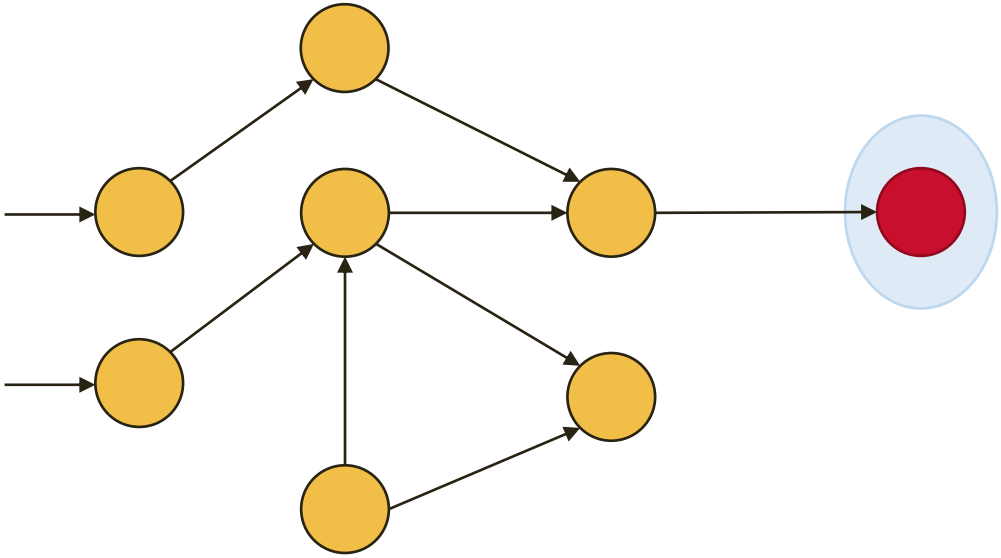
Safety Property P



Standard Reachability Analysis

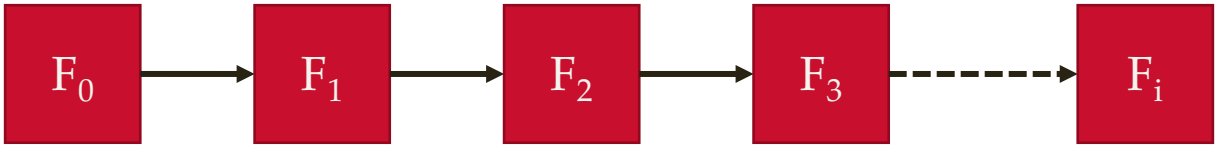
Model $M = (V, I, T)$

Safety Property P



Frame Sequences

Forward Reachability Sequence



Basic:	$F_0 = I$	
Induction:	$F_{i+1} = Reach(F_i)$	
Terminate:	$F_{i+1} \subseteq \bigcup_{0 \leq j \leq i} F_j$	← Safety
Check:	$F_i \cap \neg P \neq \emptyset$	← Unsafety (bug-finding)

Maintaining exact frame sequences is hard!

Use approximate sequences

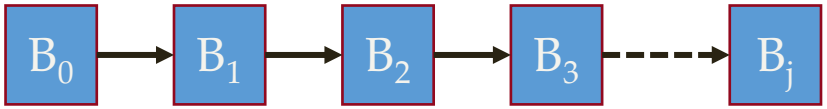
Complementary Approximate Reachability

Maintains two approximate sequences

Forward-CAR

Forward Sequence
(over-approximate)

Backward Sequence
(under-approximate)



Basic: $F_0 = I$

Induction: $F_{i+1} \supseteq Reach(F_i)$

Terminate: $F_{i+1} \subseteq \bigcup_{0 \leq j \leq i} F_j$

Safety Checking

Basic: $B_0 = \neg P$

Induction: $B_{j+1} \subseteq Reach^{-1}(B_j)$

Check: $B_j \cap I \neq \emptyset$

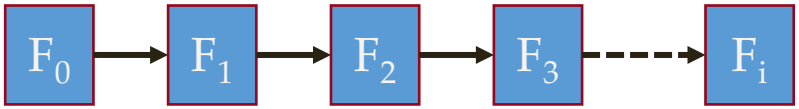
Unsafety Checking

Complementary Approximate Reachability

Maintains two approximate sequences

Backward-CAR

Forward Sequence
(under-approximate)



Backward Sequence
(over-approximate)

Basic: $F_0 = I$

Induction: $F_{i+1} \subseteq Reach(F_i)$

Check: $F_i \cap \neg P \neq \emptyset$

Unsafety Checking

Basic: $B_0 = \neg P$

Induction: $B_{j+1} \supseteq Reach^{-1}(B_j)$

Terminate: $B_{j+1} \subseteq \bigcup_{0 \leq k \leq j} B_k$

Safety Checking

SimpleCAR

- Model checker based on Complementary Approximate Reachability
  - Forward and Backward (and heuristics)
- Input: hardware circuit models expressed as AIG
- Configurable heuristics, uses Glucose as the underlying SAT solver
- Baseline performance measure for future extensions to CAR.
- Performance comparable to other state-of-the-art model checkers

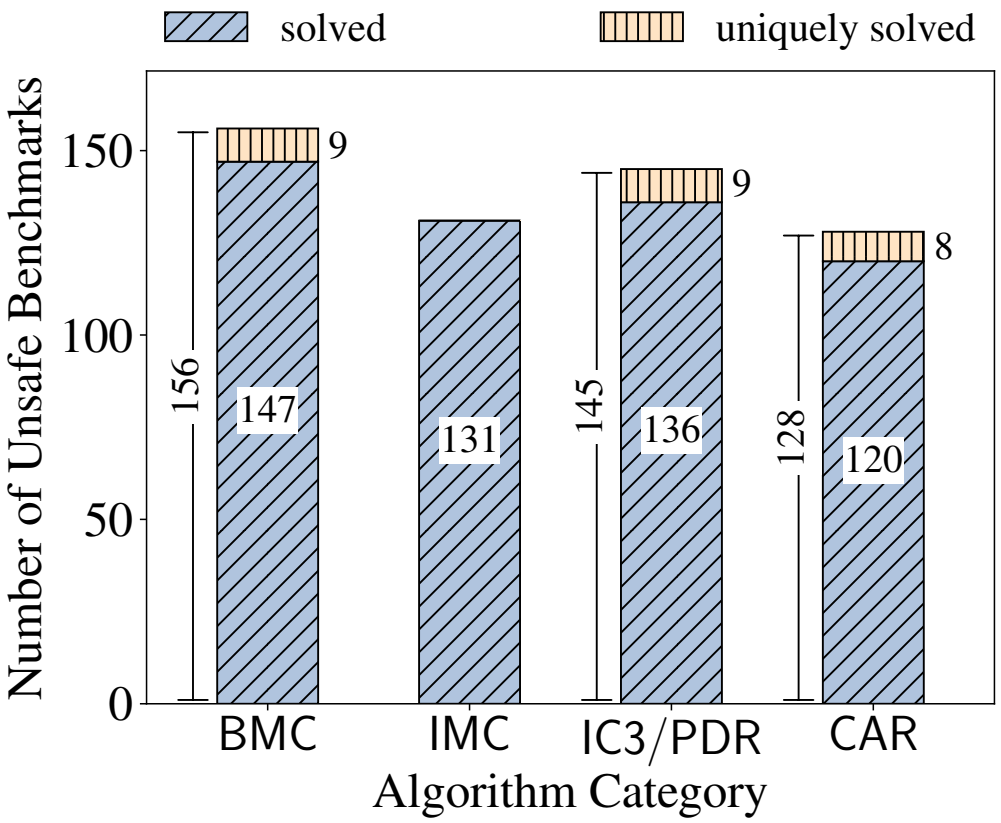
Open-source under GNU GPLv3

`http://temporallogic.org/research/CAV18/`

Performance

- Tools compared:
  - ABC x 3 algorithms
  - IIMC x 2
  - IC3Ref x 1
  - Simplic3 x 4
  - CARChecker x 2
  - SimpleCAR x 4
- 6 tools, 16 algorithms, 748 SINGLE property benchmarks from HWMCC
- Identified a bug, and counterexample generation errors

Performance



- Particularly suited for unsafety checking *aka* bug-finding
  - complements BMC and IC3 algorithm portfolios

Summary and Future Work

- SimpleCAR is a **lightweight** and **extensible** implementation of CAR
- Performance **comparable** to state-of-the-art tools
  - Complements existing model checking algorithm portfolios
- Serves as the "**bottom-line**" performance measure for future extensions
- Backward-CAR is suited for **unsafety checking**!
- Future Work
  - Tradeoff between heuristics and performance gain
  - New heuristics, shorter SAT queries

Thank you!

`http://temporallogic.org/research/CAV18/`