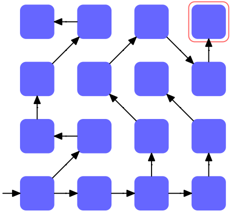
SAT-based Summarization for Boolean Programs

Gérard Basler   Daniel Kroening   Georg Weissenbacher

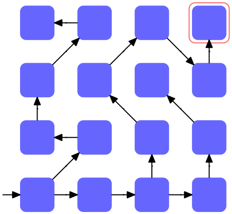
ETH Zurich

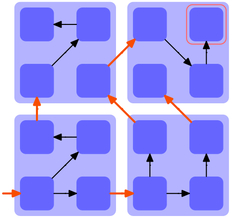
14th International SPIN Workshop on Model Checking Software, Berlin

Abstract-Verify-Refine Paradigm (CEGAR)

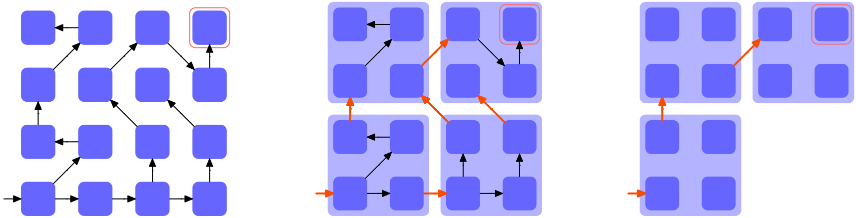


Abstract-Verify-Refine Paradigm (CEGAR)

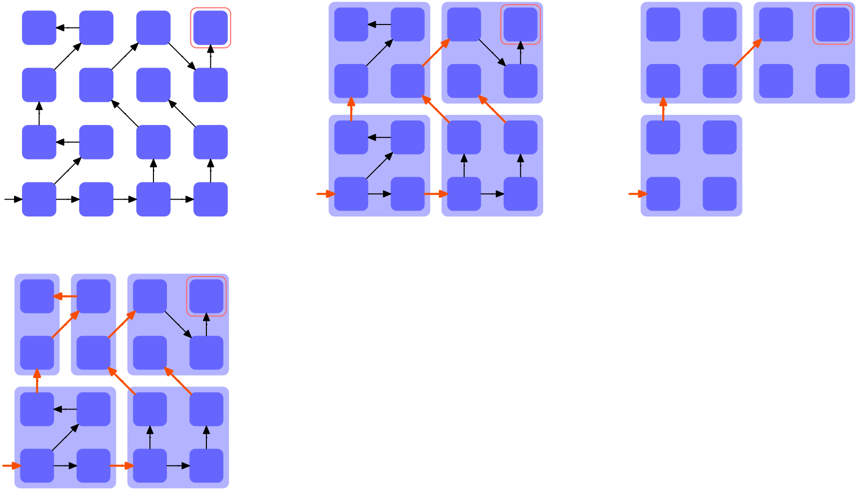




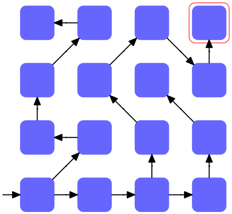
Abstract-Verify-Refine Paradigm (CEGAR)

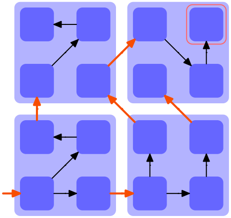


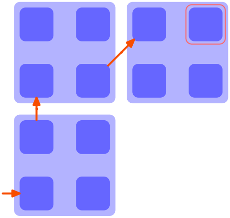
Abstract-Verify-Refine Paradigm (CEGAR)

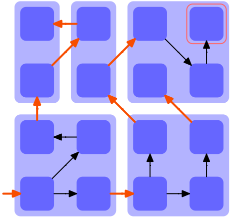


Abstract-Verify-Refine Paradigm (CEGAR)



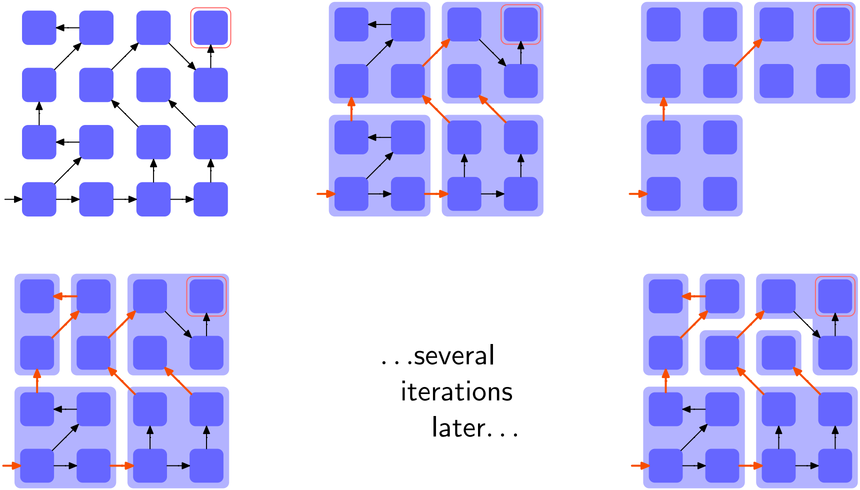






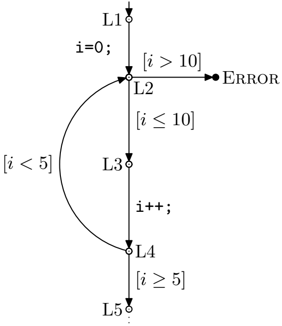
. . .several
iterations
later. . .

Abstract-Verify-Refine Paradigm (CEGAR)



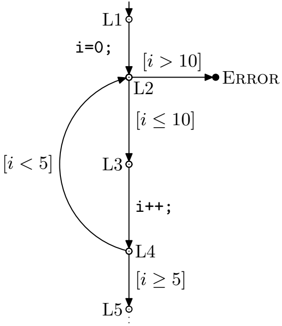
. . .several
iterations
later. . .

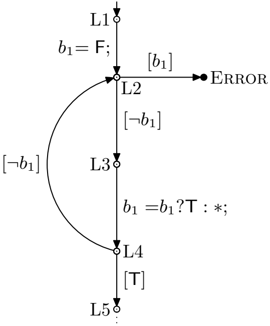
- Tracks facts in the program using *predicates* (e.g., $i < 5$, $i > 10$)
- Preserves control flow structure
- Generates *Boolean Programs*



Predicate Abstraction

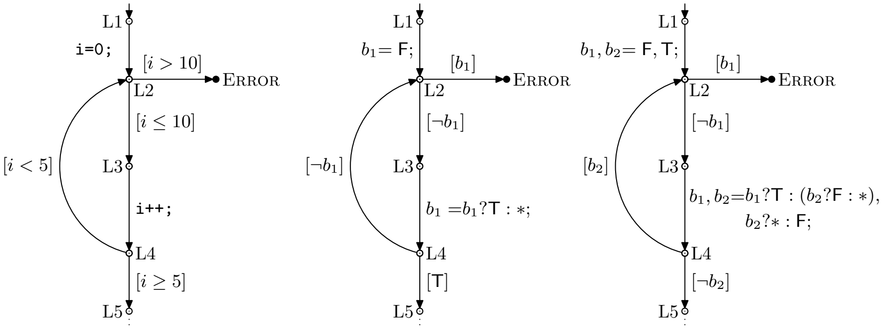
- Tracks facts in the program using *predicates* (e.g., $i < 5$, $i > 10$)
- Preserves control flow structure
- Generates *Boolean Programs*





Predicate Abstraction

- Tracks facts in the program using *predicates* (e.g., $i < 5$, $i > 10$)
- Preserves control flow structure
- Generates *Boolean Programs*



- Reachability of locations in Boolean Programs decidable
  - BDD based symbolic model checkers Bebop, Moped
- So why bother to work on a “solved” problem?

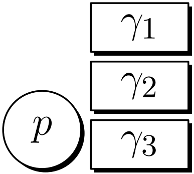
Model Checking Boolean Programs

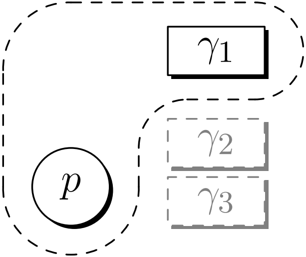
- Reachability of locations in Boolean Programs decidable
  - BDD based symbolic model checkers BEBOP, MOPED
- So why bother to work on a "solved" problem?
  - SATABS: >70% of runtime spent verifying Boolean Programs
  - BDD-based techniques don't scale for large number of variables

Model Checking Boolean Programs

- Reachability of locations in Boolean Programs decidable
  - BDD based symbolic model checkers BEBOP, MOPED
- So why bother to work on a "solved" problem?
  - SATABS: >70% of runtime spent verifying Boolean Programs
  - BDD-based techniques don't scale for large number of variables
- But is there something faster than BDDs?
  - SAT-solvers can solve instances with a huge number of variables
  - QBF-solvers are improving steadily

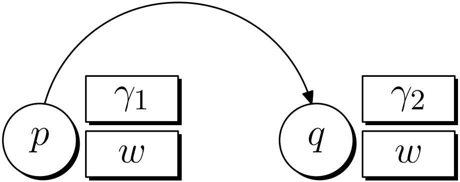
- Finite number of variables, all of them Boolean
- They have a global state and a stack



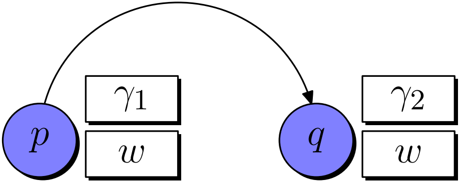


Transitions: *Neutrations*

$$\langle p, \gamma_1 \rangle \hookrightarrow \langle q, \gamma_2 \rangle$$

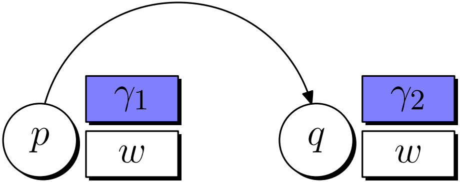


$$\langle p, \gamma_1 \rangle \hookrightarrow \langle q, \gamma_2 \rangle$$



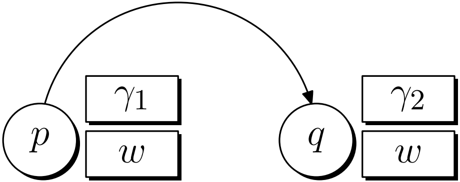
- Modify the control state p

$$\langle p, \gamma_1 \rangle \hookrightarrow \langle q, \gamma_2 \rangle$$



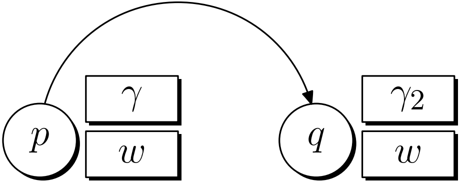
- Modify the control state p
- Modify the topmost stack element γ_1

$$\langle p, \gamma_1 \rangle \hookrightarrow \langle q, \gamma_2 \rangle$$



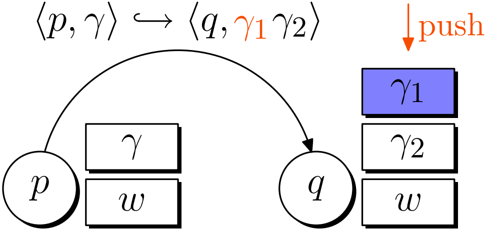
- Modify the control state p
- Modify the topmost stack element γ_1
- Do *not* modify the elements below γ_1

$$\langle p, \gamma \rangle \hookrightarrow \langle q, \gamma_1 \gamma_2 \rangle$$

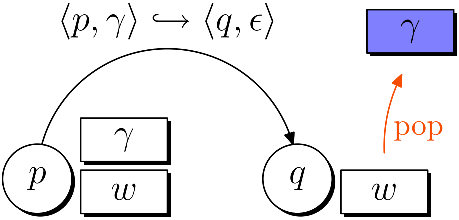


- Modify the control state p
- Modify the topmost stack element γ

$$\langle p, \gamma \rangle \hookrightarrow \langle q, \gamma_1 \gamma_2 \rangle$$



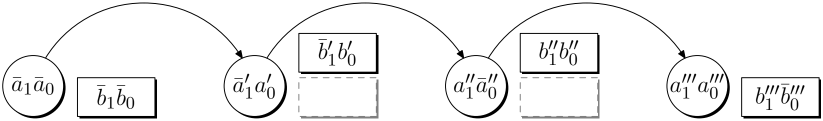
- Modify the control state p
- Modify the topmost stack element γ
- Push a new element on the stack
- Corresponds to "call"



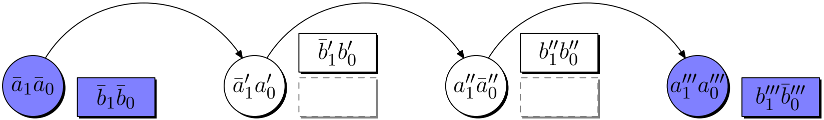
- Modify the control state p
- Pop the topmost stack element γ
- Corresponds to "return"

A Transition Sequence

- Use symbolic representation of transitions

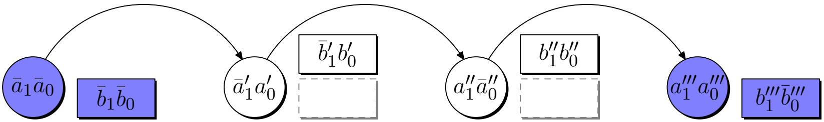


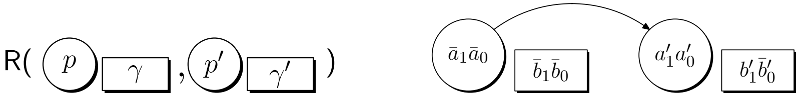
- Use symbolic representation of transitions
- Relates first and last state of path



A Transition Sequence

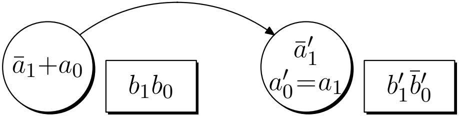
- Use symbolic representation of transitions
- Relates first and last state of path





Symbolic summary for path

- Can represent more than one explicit path

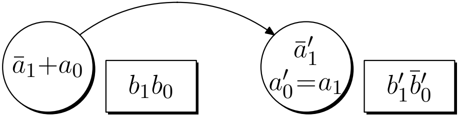


$$R(\langle a_0, a_1, b_0, b_1\rangle, \langle a'_0, a'_1, b'_0, b'_1\rangle) =$$

$$(\overline{a}_1 + a_0) \cdot (b_1 \cdot b_0) \;\cdot\; (\overline{a}'_1 \cdot (a'_0 = a_1)) \cdot (b'_1 \cdot \overline{b}'_0)$$

Symbolic summary for path

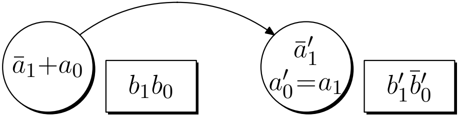
- Can represent more than one explicit path



$$R(\langle a_0, a_1, b_0, b_1 \rangle, \langle a'_0, a'_1, b'_0, b'_1 \rangle) = (\bar{a}_1 + a_0) \cdot (b_1 \cdot b_0) \cdot (\bar{a}'_1 \cdot (a'_0 = a_1)) \cdot (b'_1 \cdot \bar{b}'_0)$$

$$R(\langle p_0, \gamma_3 \rangle, \langle p_2, \gamma_2 \rangle)$$

- Can represent more than one explicit path

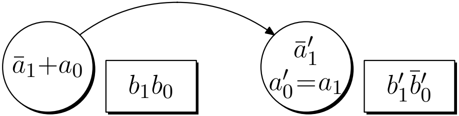


$$R(\langle a_0, a_1, b_0, b_1\rangle, \langle a'_0, a'_1, b'_0, b'_1\rangle) =$$
$$(\bar{a}_1 + a_0) \cdot (b_1 \cdot b_0) \ \cdot \ (\bar{a}'_1 \cdot (a'_0 = a_1)) \cdot (b'_1 \cdot \bar{b}'_0)$$

$$R(\langle p_0, \gamma_3\rangle, \langle p_2, \gamma_2\rangle), R(\langle p_1, \gamma_3\rangle, \langle p_2, \gamma_2\rangle)$$

Symbolic summary for path

- Can represent more than one explicit path

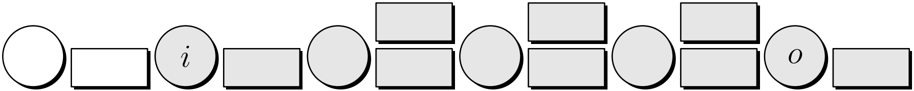


$$R(\langle a_0, a_1, b_0, b_1 \rangle, \langle a'_0, a'_1, b'_0, b'_1 \rangle) =$$

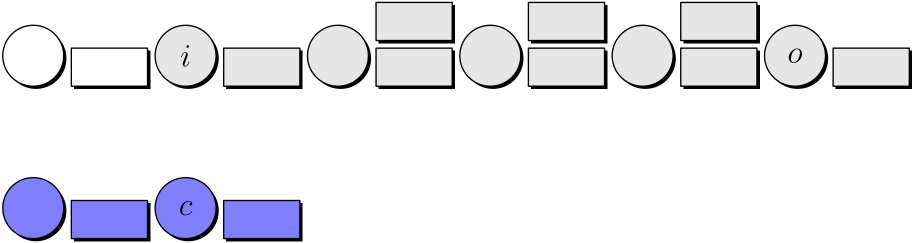
$$(\bar{a}_1 + a_0) \cdot (b_1 \cdot b_0) \ \cdot \ (\bar{a}'_1 \cdot (a'_0 = a_1)) \cdot (b'_1 \cdot \bar{b}'_0)$$

$R(\langle p_0, \gamma_3 \rangle, \langle p_2, \gamma_2 \rangle), R(\langle p_1, \gamma_3 \rangle, \langle p_2, \gamma_2 \rangle)$, and $R(\langle p_3, \gamma_3 \rangle, \langle p_3, \gamma_2 \rangle)$

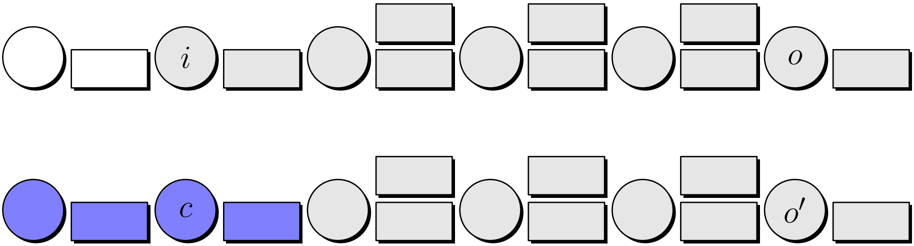
- Prefix of path constrains entry state



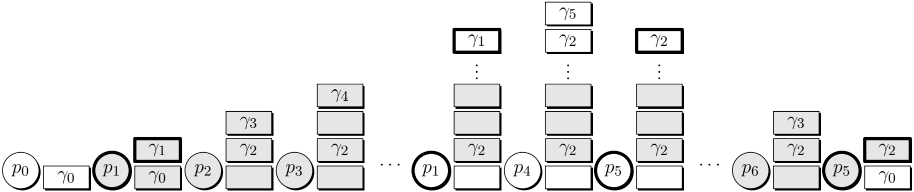
- Prefix of path constrains entry state
- Search encounters new "calling context"



- Prefix of path constrains entry state
- Search encounters new "calling context"
- QBF to determine whether entry state and calling context "compatible": $\forall s_c \exists s_i . s_c = s_i$



- Only a finite number of possible input/output pairs



- Maintain worklist of path formulas incident to nodes
- Remove from worklist if already *covered* by other formula
- Maintain set of summaries

$$R_{NEW} \subseteq R_{OLD}?$$

Symbolic Search & Summarization: When are we done?

- Maintain worklist of path formulas incident to nodes
- Remove from worklist if already *covered* by other formula
- Maintain set of summaries

$$R_{NEW} \subseteq R_{OLD}?$$

$$\forall \langle p_0, \gamma_0 \rangle, \langle p'_0, \gamma'_0 \rangle .\ \exists \langle p_1, \gamma_1 \rangle, \langle p'_1, \gamma'_1 \rangle .$$

$$R_{NEW}(\ \textcircled{p_0}\,\boxed{\gamma_0}\ ,\textcircled{p'_0}\,\boxed{\gamma'_0}\ ) = R_{OLD}(\ \textcircled{p_1}\,\boxed{\gamma_1}\ ,\textcircled{p'_1}\,\boxed{\gamma'_1}\ )$$

- *Universal Summary* provides a summary for *any arbitrary* entry state
- "calling context" is unconstrained

- *Universal Summary* provides a summary for *any arbitrary* entry state
- "calling context" is unconstrained

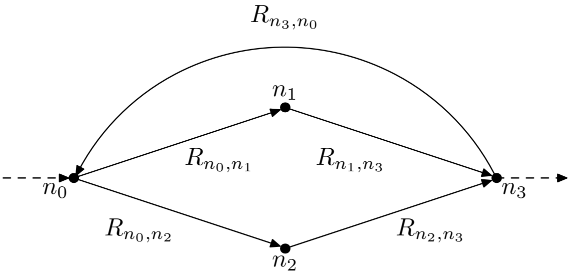
$$\Sigma_{\mathcal{U}}(\langle p, \gamma \rangle, \langle p', \gamma' \rangle)$$

$$\iff$$

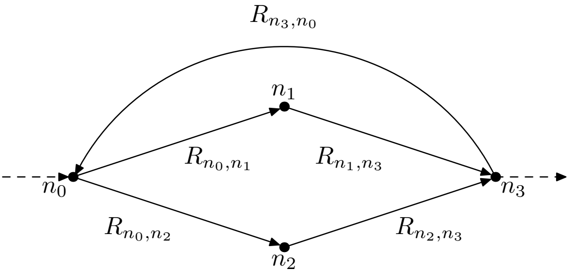
$$\forall \langle p, \gamma w \rangle.\ (\exists \langle p_1, w_1 \rangle, \ldots, \langle p_n, w_n \rangle.$$

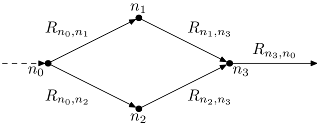
$$\langle p, \gamma \rangle \rightarrow \langle p_1, w_1 \rangle \rightarrow \ldots \rightarrow \langle p_n, w_n \rangle \rightarrow \langle p', \gamma' \rangle \wedge$$

$$\forall i \in \{1..n\}. |w_i| \geq 2)$$

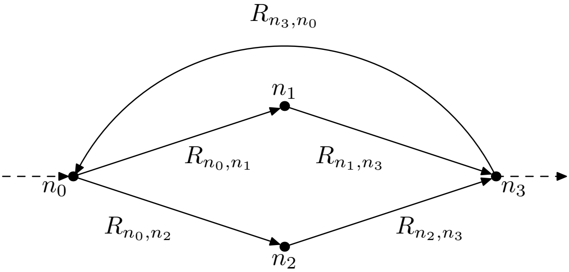

R_{n_3,n_0}
n_1
R_{n_0,n_1}
R_{n_1,n_3}
n_0
n_3
R_{n_0,n_2}
R_{n_2,n_3}
n_2

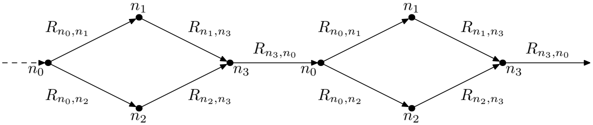
Constructing Universal Summaries using BMC



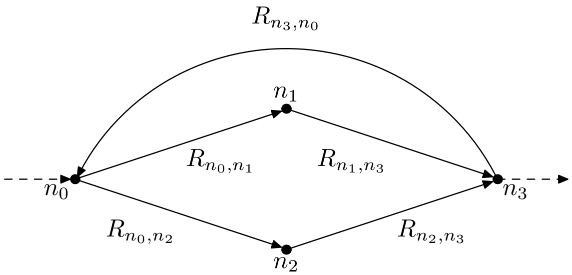


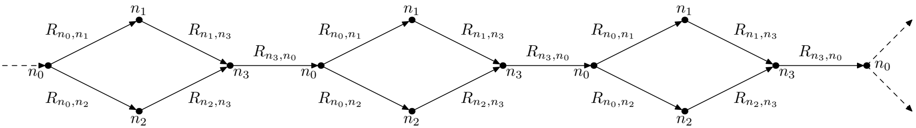
Constructing Universal Summaries using BMC



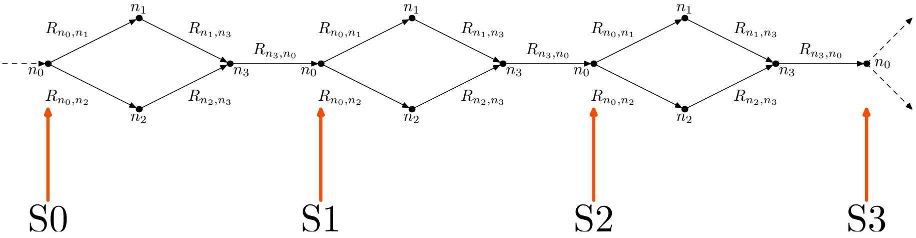


Constructing Universal Summaries using BMC





BMC: When to stop unrolling?



- unroll up to *longest loop-free path*
- all states in the path are *pairwise* different
- $\exists S0, S1, S2, S3 \ . \ S0 \neq S1 \wedge S0 \neq S2 \wedge S0 \neq S3 \wedge S1 \neq S2 \ldots$

- Start with *innermost* functions (no call to other function)
- Construct summaries in *top down* manner
- *Merge* all summaries obtained by unrolling

$$\bigvee_{i=1}^{k} \Sigma_i(\; p \; \gamma \;, p' \; \gamma' \;)$$

Universal Summaries

- Start with *innermost* functions (no call to other function)
- Construct summaries in *top down* manner
- *Merge* all summaries obtained by unrolling

$$\bigvee_{i=1}^{k} \Sigma_i(\boxed{p}\ \boxed{\gamma}, \boxed{p'}\ \boxed{\gamma'})$$

- Applicable in *all* calling contexts!

Results

Benchmark	#vars	Bebop	QBF-summaries	univ. summ.	violation
adddevice	434	4m37.4s	0m0.6s	0m1.8s	yes
nulldevice	434	4m34.0s	0m8.6s	0m1.4s	yes
pendedcompletedreq	86	0m30.9s	timeout	0m13.5s	yes
targetrelationneedsref	37	0m0.4s	0m0.5s	0m2.74s	no
markirppending	11	0m0.4s	0m3.0s	0m18.5s	no
wmiforward	15	0m0.7s	0m2.0s	0m15.3s	no
Terminator 1	74	timeout	1m55.9s	1m55.9s	yes
Terminator 2	60	88m22.6s	timeout	timeout	yes

- Advantages:
  - Universal Summaries good for bug finding
    - In CEGAR, for n iterations, $\geq n - 1$ of the abstractions have a "bug"
  - Eliminates many calls to QBF solver
- Disadvantages:
  - Does not work for programs with recursion: Fall back to QBF
  - Large universal summaries combined with QBF too hard for solver
  - Does not scale very well for "bug-free" programs