

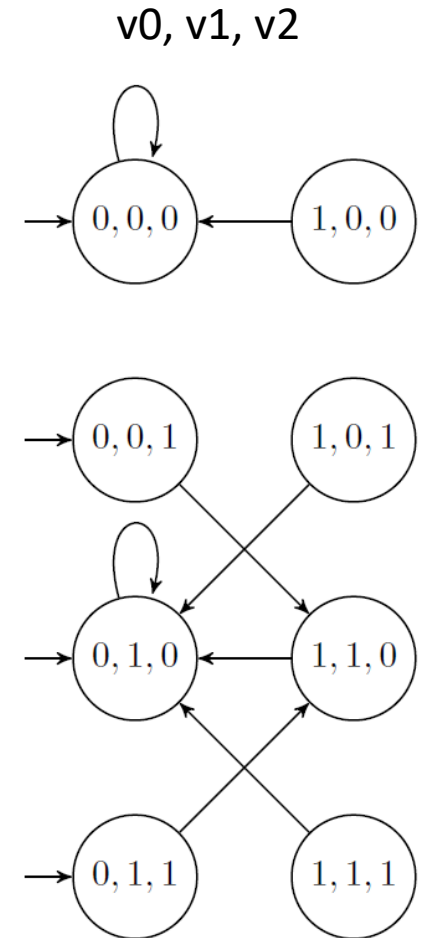
The initial value of the state variable v_0 of the circuit is *false*. The initial values of v_1 and v_2 are unknown.

Task 1a. [5 Points]

- State the formula S_0 that represents the set of initial states and the formula R that represents the transition relation of C .

Task 1b. [5 Points]

- Draw the Kripke Structure $M = (S, S_0, R, AP, L)$ that represents C .



taken from an anonymous student

SAT-Based Model Checking

Chapter 10

Outline

- Bounded Model Checking
- Verifying Reachability Properties with k -induction
- Model Checking with Inductive Invariants
- Model Checking with Craig Interpolants
- Property-Directed Reachability

Overview

- SAT solvers can solve propositional formulas. (See Chapter 9.)
- SAT solvers have become very fast

Techniques

- Bounded Model Checking: Is there a trace of length k that violates the property?
- k -induction: Can we prove the property inductively for *any* trace?
- Create an inductive invariant that is stronger than the property
 - Craig Interpolants
 - Property-Directed Reachability

In this chapter, we will only consider LTL formulas

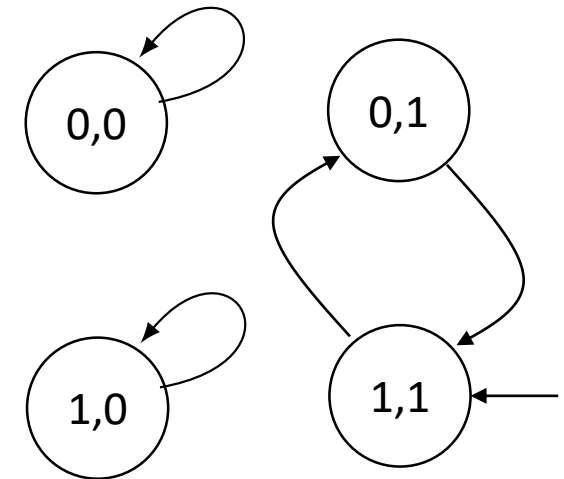
Preliminaries

Postimage

$\text{postimg}(S)$ = states reachable from S in one step

$\text{postimg}(\{ (0,1) \}) =$

$\text{postimg}(\{ (0,0), (1,1) \}) =$



Kripke structure

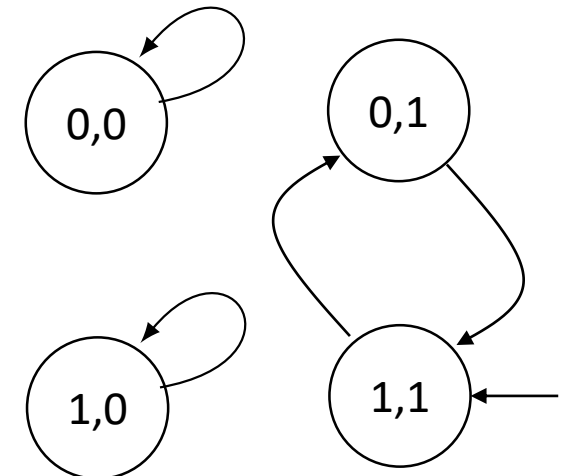
Paths

Initial states: $\mathcal{S}_0(x, y) = (x = 1 \wedge y = 1)$

Transitions: $\mathcal{R}(x, y, x', y') = (x' = (x + y) \bmod 2) \wedge (y' = y)$

$Path_1(S(V)) = S \wedge R(V, V') =$

$\exists V. \neg x \wedge y \wedge (x' = (x + y) \bmod 2) \wedge (y' = y) =$



Kripke structure

Paths

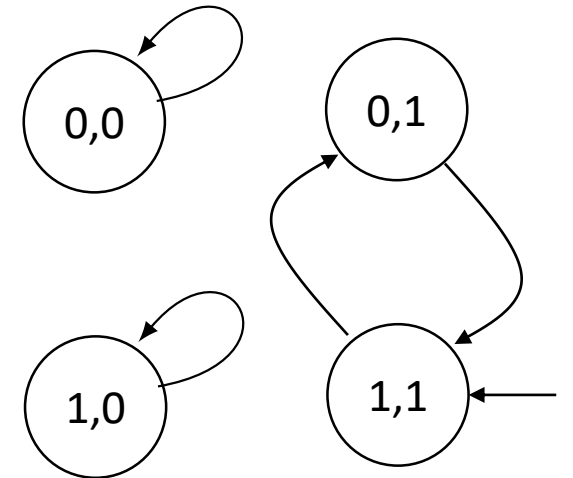
Initial states: $\mathcal{S}_0(x, y) = (x = 1 \wedge y = 1)$

Transitions: $\mathcal{R}(x, y, x', y') = (x' = (x + y) \bmod 2) \wedge (y' = y)$

$Path_1(S(V)) = S \wedge R(V, V') =$

$x \wedge y \wedge (x' = (x + y) \bmod 2) \wedge (y' = y) =$

$x \wedge y \wedge \neg x' \wedge y'$.



Kripke structure

Bounded Model Checking

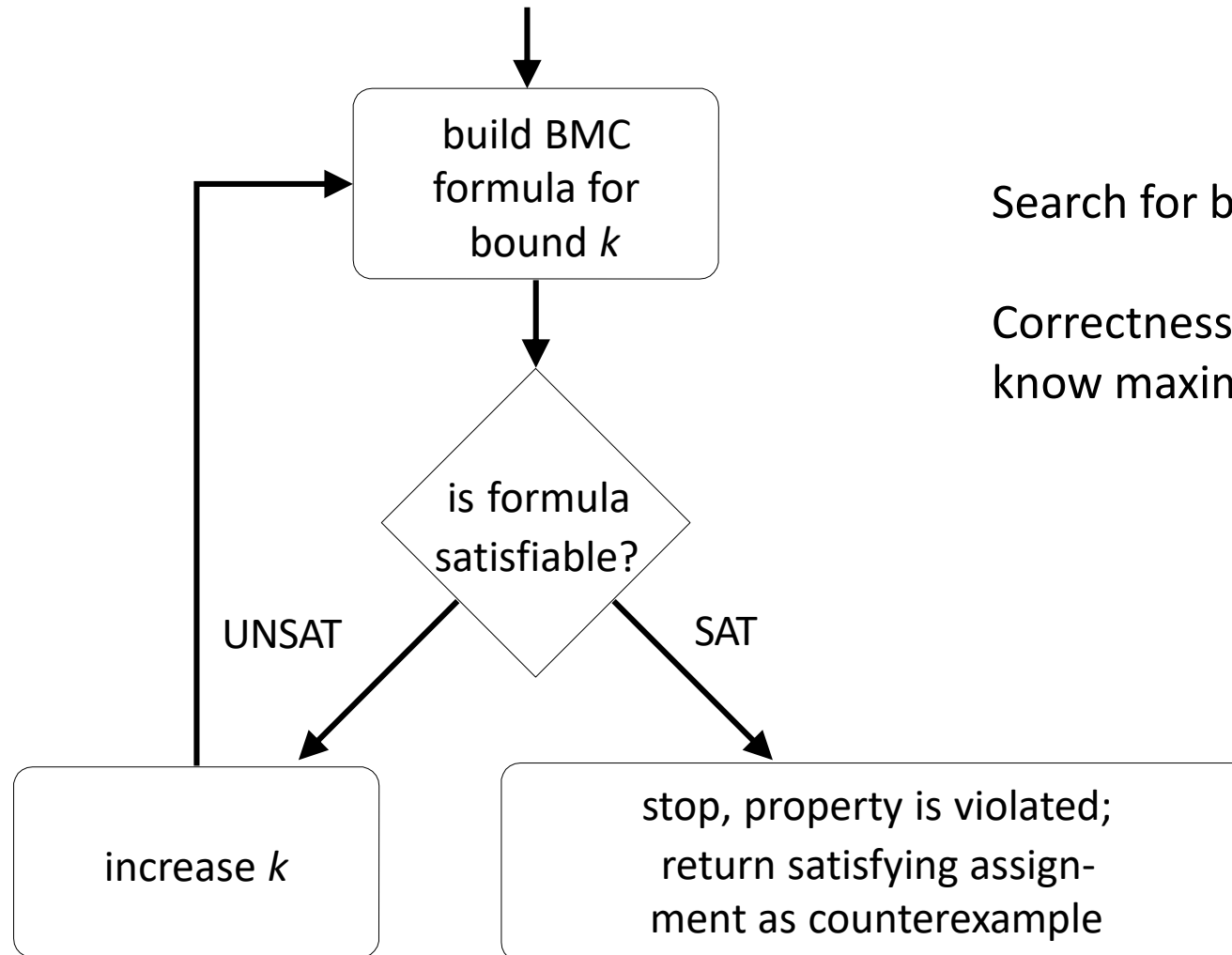
Computer Aided Verification Award 2018

Bounded Model Checking has revolutionized the way model checking is used and perceived. It has increased the capabilities of model checkers by orders of magnitude, turning them into a standard tool for hardware verification and a very important component of the toolkit available for software verification...

(1999)



Bounded Model Checking



Search for bugs within k steps.

Correctness can only be proven if we know maximal value for k

Reachability Properties

Property

p always holds iff we cannot reach a state with $\neg p$

Kripke Structure

$M = (S, S_0, R, AP, L)$ – represented symbolically (See Chapter 3)

State variables $V = \{v_1, \dots, v_n\}$

Reachability – Paths

$$\mathit{path}_0(s_0) =$$

$$\mathit{path}_1(s_0, s_1) =$$

$$\mathit{path}_2(s_0, \dots, s_2) =$$

$$\mathit{path}_k(s_0, \dots, s_k) =$$

Reachability – Paths

$$path_0(s_0) = S_0(s_0)$$

$$path_1(s_0, s_1) = S_0(s_0) \wedge R(s_0, s_1)$$

$$path_2(s_0, \dots, s_2) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2)$$

$$path_3(s_0, \dots, s_3) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \wedge R(s_2, s_3)$$

$$path_k(s_0, \dots, s_k) = S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$$

Reachability – Building the Formula

Reachability – Building the Formula

$$path_k(s_0, \dots, s_k) = S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$$

Path starts in initial state Exists transition from s_i to s_{i+1}

System is **incorrect** within k steps if

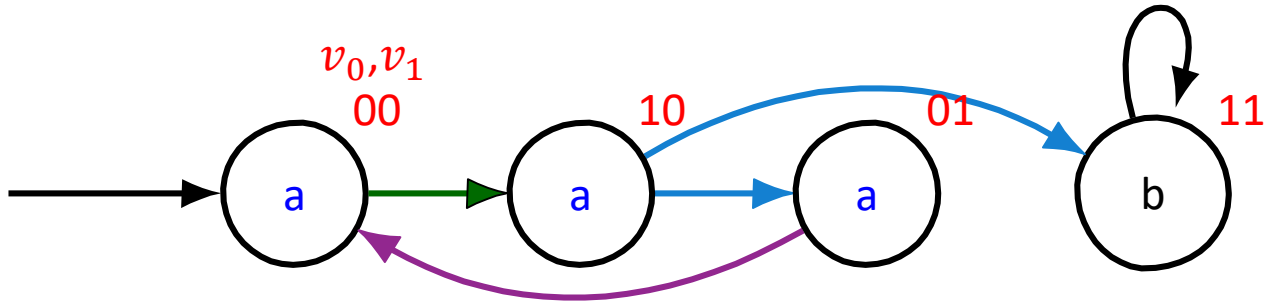
$$path_k(s_0, \dots, s_k) \wedge \bigvee_{i=0}^k \neg p(s_i)$$

There is a path to s_k One of the state violates p

Reachability – Correctness

Theorem 10.1. $path_{k(s_0, \dots, s_k)} \wedge \bigvee_{i=0}^k \neg p(s_i)$ is satisfiable iff there is a counterexample to $AG\ p$ of length $\leq k$.

Example



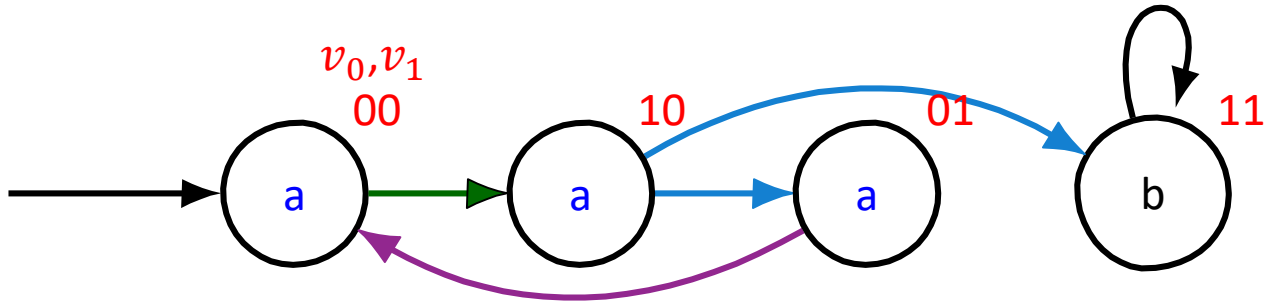
AG a

$S_0(v_0, v_1) = .$

$R(v_0, v_1, v'_0, v'_1) =$

$a(v_0, v_1) = .$

Example



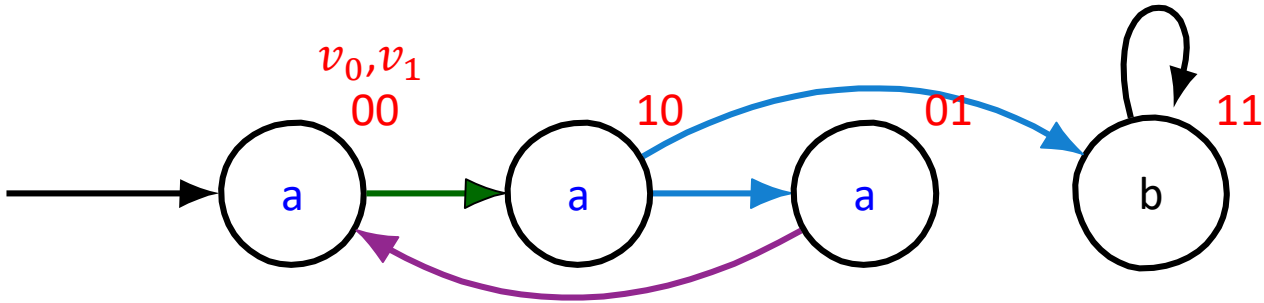
AG a

$$S_0(v_0, v_1) = \neg v_0 \wedge \neg v_1$$

$$R(v_0, v_1, v'_0, v'_1) = \neg v_0 \wedge \neg v_1 \wedge v'_0 \wedge \neg v'_1 \\ \vee v_0 \wedge \neg v_1 \wedge v'_1 \\ \vee \neg v_0 \wedge v_1 \wedge \neg v'_0 \wedge \neg v'_1 \\ \vee v_0 \wedge v_1 \wedge v'_0 \wedge v'_1$$

$$a(v_0, v_1) = \neg v_0 \vee \neg v_1$$

Example



$$S_0(v_0, v_1) = \neg v_0 \wedge \neg v_1$$

$$R(v_0, v_1, v'_0, v'_1) = \begin{aligned} &\neg v_0 \wedge \neg v_1 \wedge v'_0 \wedge \neg v'_1 \\ &\vee v_0 \wedge \neg v_1 \wedge v'_1 \\ &\vee \neg v_0 \wedge v_1 \wedge \neg v'_0 \wedge \neg v'_1 \\ &\vee v_0 \wedge v_1 \wedge v'_0 \wedge v'_1 \end{aligned}$$

$$a(v_0, v_1) = \neg v_0 \vee \neg v_1$$

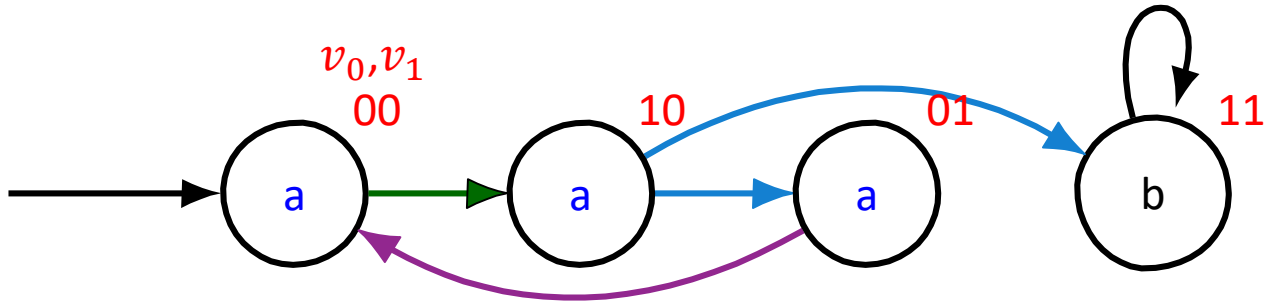
$$path_1(s_0, s_1, s_2) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2)$$

$$= \neg v_{0,0} \wedge \neg v_{1,0} \wedge$$

$$\left(\begin{array}{l} \neg v_{00} \wedge \neg v_{10} \wedge v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge \neg v_{10} \wedge v_{11} \\ \vee \neg v_{00} \wedge v_{10} \wedge \neg v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge v_{10} \wedge v_{01} \wedge v_{11} \end{array} \right)$$

$$\bigvee_{i=0}^k \neg a(s_i) = \neg(\neg v_{00} \vee \neg v_{10}) \vee \neg(\neg v_{01} \vee \neg v_{11})$$

Example



$$S_0(v_0, v_1) = \neg v_0 \wedge \neg v_1$$

$$R(v_0, v_1, v'_0, v'_1) = \begin{aligned} &\neg v_0 \wedge \neg v_1 \wedge v'_0 \wedge \neg v'_1 \\ &\vee v_0 \wedge \neg v_1 \wedge v'_1 \\ &\vee \neg v_0 \wedge v_1 \wedge \neg v'_0 \wedge \neg v'_1 \\ &\vee v_0 \wedge v_1 \wedge v'_0 \wedge v'_1 \end{aligned}$$

$$a(v_0, v_1) = \neg v_0 \vee \neg v_1$$

$$path_2(s_0, s_1, s_2) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2)$$

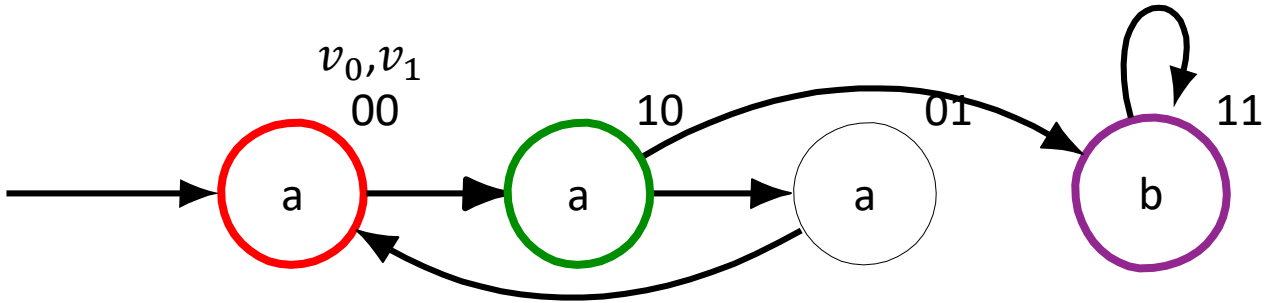
$$= \neg v_{0,0} \wedge \neg v_{1,0} \wedge$$

$$\left(\begin{array}{l} \neg v_{00} \wedge \neg v_{10} \wedge v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge \neg v_{10} \wedge v_{11} \\ \vee \neg v_{00} \wedge v_{10} \wedge \neg v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge v_{10} \wedge v_{01} \wedge v_{11} \end{array} \right) \wedge$$

$$\left(\begin{array}{l} \neg v_{01} \wedge \neg v_{11} \wedge v_{02} \wedge \neg v_{12} \\ \vee v_{01} \wedge \neg v_{11} \wedge v_{12} \\ \vee \neg v_{01} \wedge v_{11} \wedge \neg v_{02} \wedge \neg v_{12} \\ \vee v_{01} \wedge v_{11} \wedge v_{02} \wedge v_{12} \end{array} \right)$$

$$\bigvee_{i=0}^k \neg a(s_i) = \neg(\neg v_{00} \vee \neg v_{10}) \vee \neg(\neg v_{01} \vee \neg v_{11}) \vee \neg(\neg v_{02} \vee \neg v_{12})$$

Example



AG a

$$\begin{aligned}
 path_2(s_0, s_1, s_2) &= S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \\
 &= \neg v_{0,0} \wedge \neg v_{1,0} \wedge \\
 &\quad \left(\begin{array}{l} \neg v_{00} \wedge \neg v_{10} \wedge v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge \neg v_{10} \wedge v_{11} \\ \vee \neg v_{00} \wedge v_{10} \wedge \neg v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge v_{10} \wedge v_{01} \wedge v_{11} \end{array} \right) \wedge \\
 &\quad \left(\begin{array}{l} \neg v_{01} \wedge \neg v_{11} \wedge v_{02} \wedge \neg v_{12} \\ \vee v_{01} \wedge \neg v_{11} \wedge v_{12} \\ \vee \neg v_{01} \wedge v_{11} \wedge \neg v_{02} \wedge \neg v_{12} \\ \vee v_{01} \wedge v_{11} \wedge v_{02} \wedge v_{12} \end{array} \right)
 \end{aligned}$$

Satisfying assignment

i	0	1	2
v_{0i}	0	1	1
v_{1i}	0	0	1

$$\bigvee_{i=0}^k \neg a(s_i) = \neg(\neg v_{00} \vee \neg v_{10}) \vee \neg(\neg v_{01} \vee \neg v_{11}) \vee \neg(\neg v_{02} \vee \neg v_{12})$$

Try it with Z3!

```
(declare-const v00 Bool)
(declare-const v10 Bool)
(declare-const v01 Bool)
(declare-const v11 Bool)
(declare-const v02 Bool)
(declare-const v12 Bool)

(define-fun S0 ((v0 Bool) (v1 Bool)) Bool
  (and (not v0) (not v1)))

(define-fun R ((v0 Bool) (v1 Bool) (w0 Bool) (w1 Bool))
  Bool
  (or
    (and (not v0) (not v1) w0 (not w1))
    (and v0 (not v1) w1)
    (and (not v0) v1 (not w0) (not w1))
    (and v0 v1 w0 w1))
  )

(define-fun a ((v0 Bool) (v1 Bool)) Bool
  (or (not v0) (not v1))
  )

(define-fun path1 ((v00 Bool) (v10 Bool) (v01 Bool) (v11
  Bool)) Bool
  (and (S0 v00 v10)
    (R v00 v10 v01 v11)
  )
  )
```

```
(define-fun path2 ((v00 Bool) (v10 Bool) (v01 Bool) (v11
  Bool) (v02 Bool) (v12 Bool)) Bool
  (and (S0 v00 v10)
    (R v00 v10 v01 v11)
    (R v01 v11 v02 v12)
  )
  )

; k = 1
; (assert
;   (and (path1 v00 v10 v01 v11)
;     (or (not (a v00 v10))
;       (not (a v01 v11))
;     )
; )
;)

; k = 2
; (assert
;   (and (path2 v00 v10 v01 v11 v02 v12)
;     (or (not (a v00 v10))
;       (not (a v01 v11))
;       (not (a v02 v12))
;     )
; )
;)

(check-sat)
(get-model)
```

<https://rise4fun.com/Z3>

Another Example

Does modulo-8 counter of Chapter 3.5 ever reach 4?

$$\neg p = \neg v_0 \wedge \neg v_1 \wedge v_2.$$

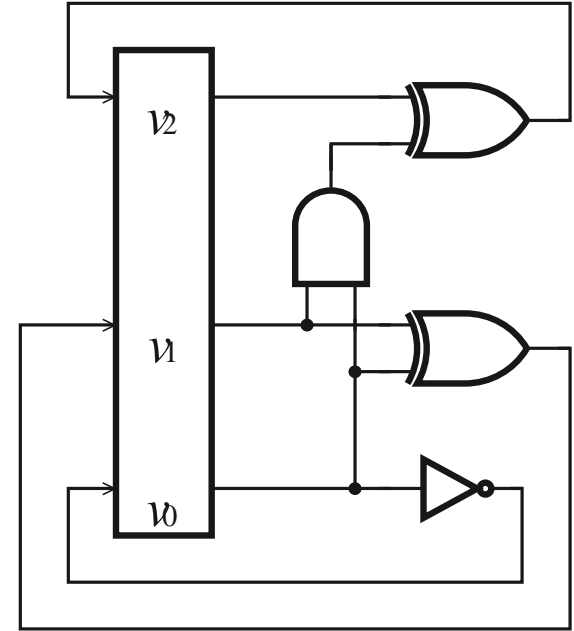
$$S_0(V) = \neg v_0 \wedge \neg v_1 \wedge \neg v_2.$$

$$R(V, V') = (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \oplus v_1) \wedge (v'_2 \leftrightarrow (v_0 \wedge v_1) \oplus v_2).$$

$k = 0$:

$$S_0(v) \quad \neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge$$

$$\neg p(v) \quad \neg v_0 \wedge \neg v_1 \wedge v_2.$$



Another Example

Does module-8 counter of Chapter 3.5 ever reach 4?

$$\neg p = \neg v_0 \wedge \neg v_1 \wedge v_2.$$

$$S_0(V) = \neg v_0 \wedge \neg v_1 \wedge \neg v_2.$$

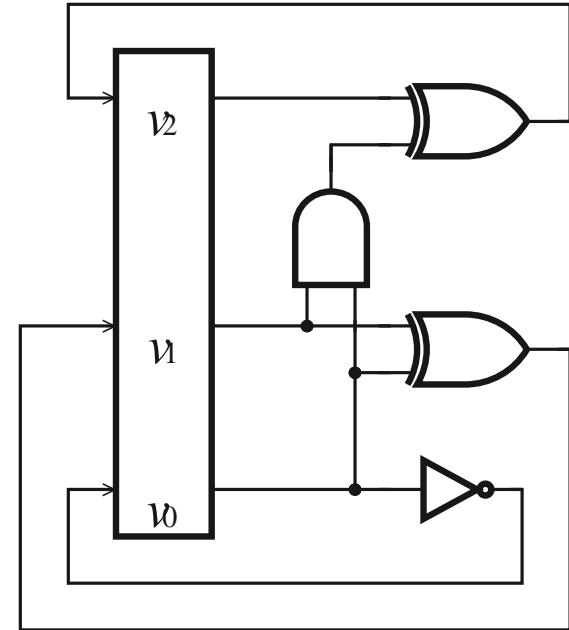
$$R(V, V') = (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \oplus v_1) \wedge (v'_2 \leftrightarrow (v_0 \wedge v_1) \oplus v_2).$$

$k = 1$:

$$S_0(v) \quad \neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge$$

$$R(v, v') \quad (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \oplus v_1) \wedge (v'_2 \leftrightarrow (v_0 \wedge v_1) \oplus v_2) \wedge$$

$$(\underbrace{\neg v_0 \wedge \neg v_1 \wedge v_2}_{\neg p(v)} \vee \underbrace{\neg v'_0 \wedge \neg v'_1 \wedge v'_2}_{\neg p(v')}).$$



Another Example

Does module-8 counter of Chapter 3.5 ever reach 4?

$$\neg p = \neg v_0 \wedge \neg v_1 \wedge v_2.$$

$$S_0(V) = \neg v_0 \wedge \neg v_1 \wedge \neg v_2.$$

$$R(V, V') = (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \oplus v_1) \wedge (v'_2 \leftrightarrow (v_0 \wedge v_1) \oplus v_2).$$

k = 2:

$$S_0(v) \quad \neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge$$

$$R(v, v') \quad (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \oplus v_1) \wedge (v'_2 \leftrightarrow (v_0 \wedge v_1) \oplus v_2) \wedge$$

$$R(v', v'') \quad (v''_0 \leftrightarrow \neg v'_0) \wedge (v''_1 \leftrightarrow v'_0 \oplus v'_1) \wedge (v''_2 \leftrightarrow (v'_0 \wedge v'_1) \oplus v'_2) \wedge$$

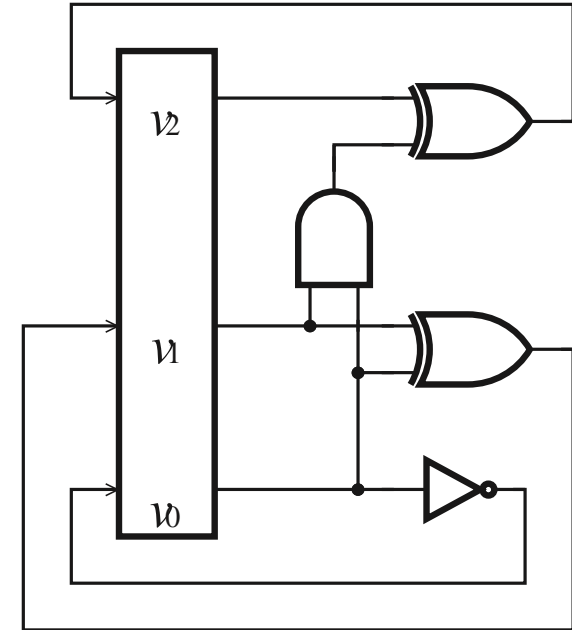
$$(\neg v_0 \wedge \neg v_1 \wedge v_2 \vee \neg v'_0 \wedge \neg v'_1 \wedge v'_2 \vee \neg v''_0 \wedge \neg v''_1 \wedge v''_2.)$$

$\neg p(v)$

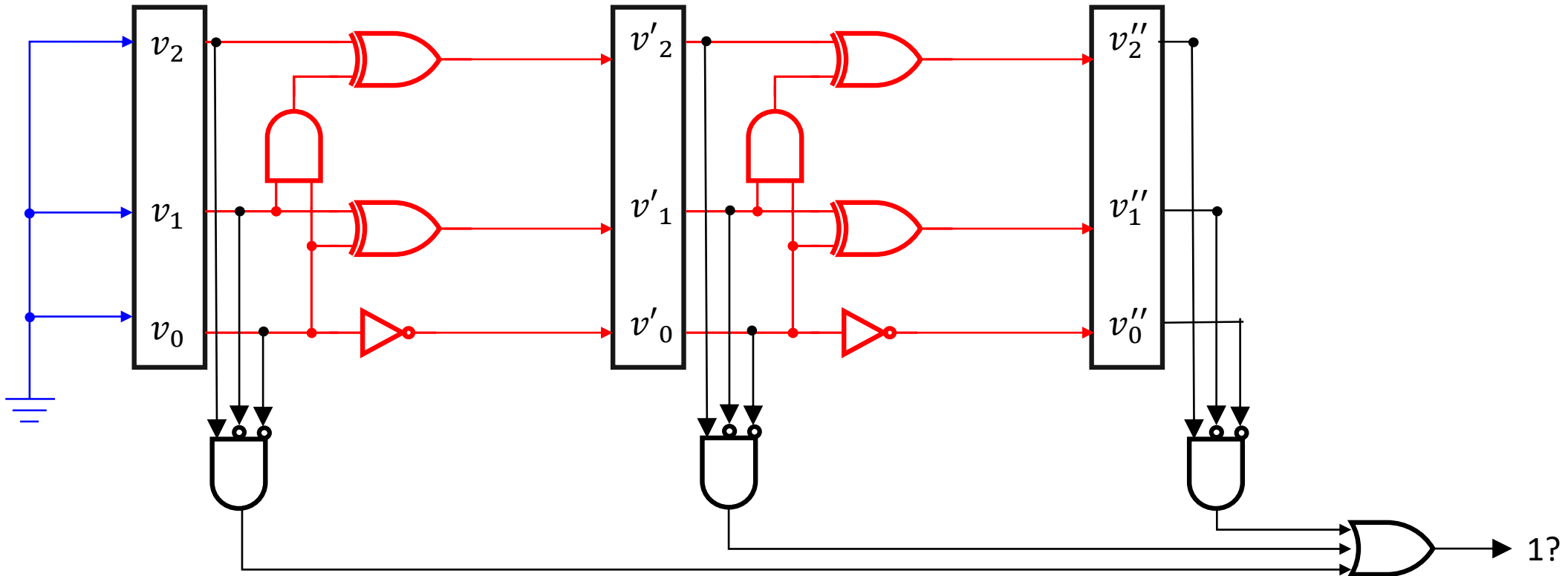
$\neg p(v')$

Model Checking

$\neg p(v'')$



The Electrical Engineer's Viewpoint



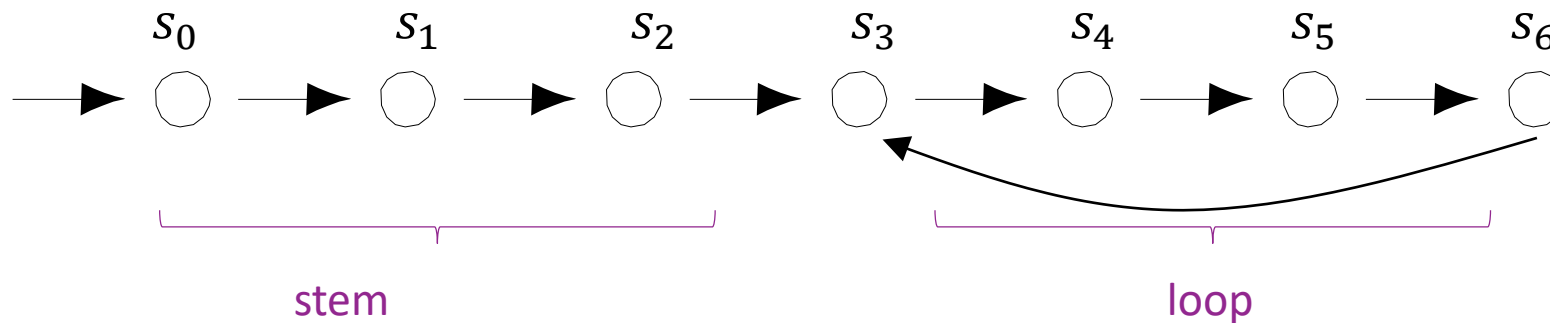
Eventuality Properties

Property

Suppose $\phi = \mathbf{AF} p$

Counterexamples fulfill $\mathbf{EG} \neg p$

Counterexamples have *Lasso Shape* (See Chapter 4.)



Completeness

BMC finds bugs of length $\leq k$.

Longer counterexamples may exist!

Is there a k big enough to exclude any counterexample?

Def. $M \models_k \phi$: all computations of length k satisfy ϕ .

Completeness threshold: Number CT such that $M \models_{CT} \phi \Rightarrow M \models \phi$

If completeness threshold known, stop BMC when $k = CT$

Ideas for finding CT ?

Completeness Threshold

Finding smallest CT is as difficult as model checking!

- Smallest CT is size of shortest counterexample, or 0 if the property is satisfied

Simple values for CT

- Number of state of M is bound on CT
- Diameter (longest simple path between two states) is bound on CT

These are typically really large numbers

Verifying Reachability Properties with k -induction



Mary Sheeran, Koen Claessen, Per Bjesse,
2000

Motivation

- Completeness thresholds usually very large
- Can we **prove** a property with fewer unrollings?
- **Idea: Use induction.**

Base: Prove $Q(0)$

Induction: Prove $Q(n - 1) \Rightarrow Q(n)$

Conclusion: $\forall n. Q(n)$

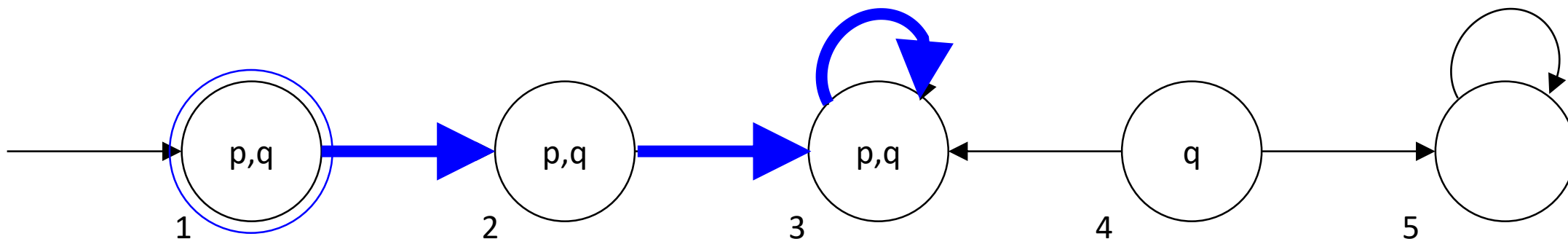
Caveat: Property may be true, but not inductive (see below)

Induction

Let's prove **AG** p on the following structure.

Take arbitrary path π

- **Base case:** $\pi(0) \models p$ true: $q_1 \models p$
- **Induction:** if $\pi(n - 1) \models p$ then $\pi(n) \models p$ true: any successor of a p -state is a p -state
- **Conclusion:** for any path π we have $\forall n. \pi(n) \models p$

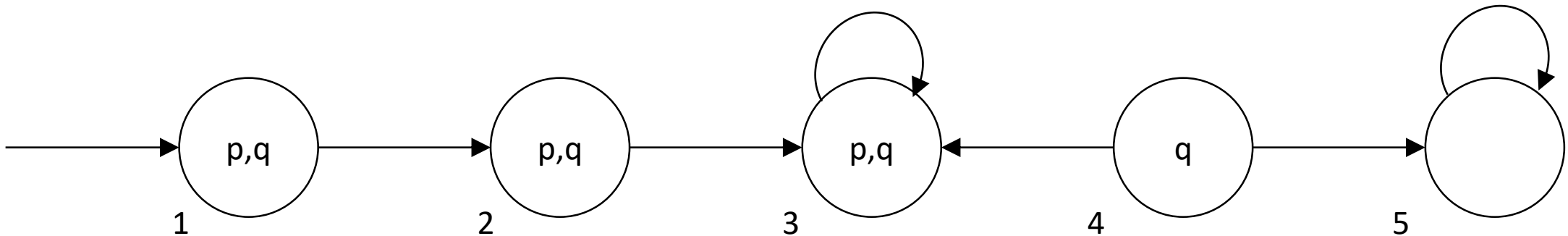


Satisfiability

Let's prove $AG\ p$ on the following structure. How can these properties be violated?

Take arbitrary path π

- **Base case:** $\pi(0) \models p$ $S_o(s) \wedge \neg p(s)$ Unsatisfiable
- **Induction:** if $\pi(n - 1) \models p$ then $\pi(n) \models p$ $p(s) \wedge R(s, s') \wedge \neg p(s')$ Unsatisfiable
- **Conclusion:** for any path π we have $\forall n. \pi(n) \models p$

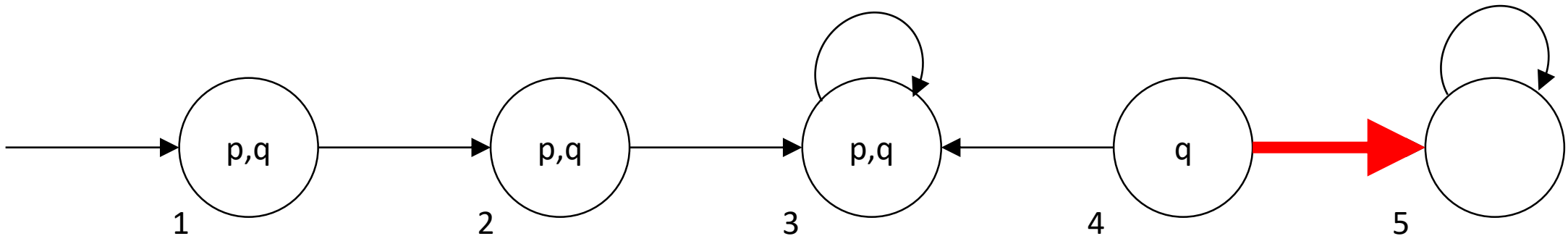


A Problem

Let's prove **AG** q on the following structure.

Take arbitrary path π

- **Base case:** $\pi(0) \models q$
- **Induction:** if $\pi(n - 1) \models q$ then $\pi(n) \models q$ **not true!**
- ~~**Conclusion:** for any path π we have $\forall n. \pi(n) \models q$~~ **not all true properties are inductive**



k -induction

Base:

Induction:

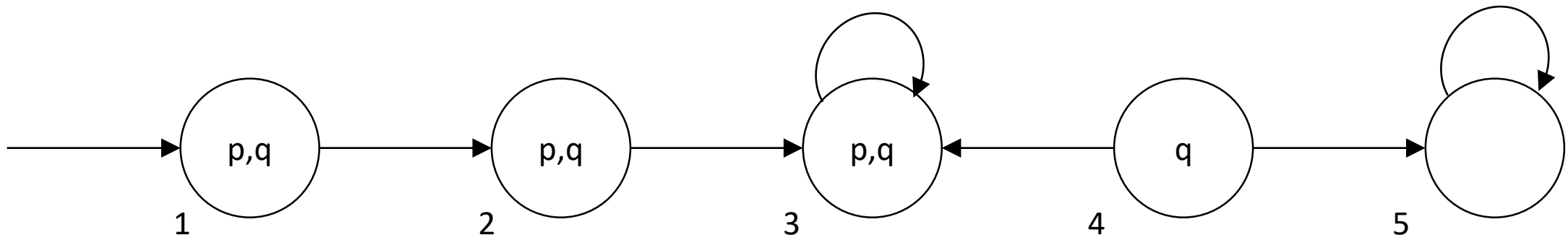
Conclusion: $\forall n. Q(n)$

In our setting:

Base. all paths of length k from S_0 are labeled q

Induction. all paths of length k labeled with all qs are followed by a q

Conclusion. All paths from S_0 are labeled q



k -induction

Base: Prove $Q(0) \wedge \dots \wedge Q(k - 1)$

Induction: Prove $Q(n - k) \wedge \dots \wedge Q(n - 1) \Rightarrow Q(n)$

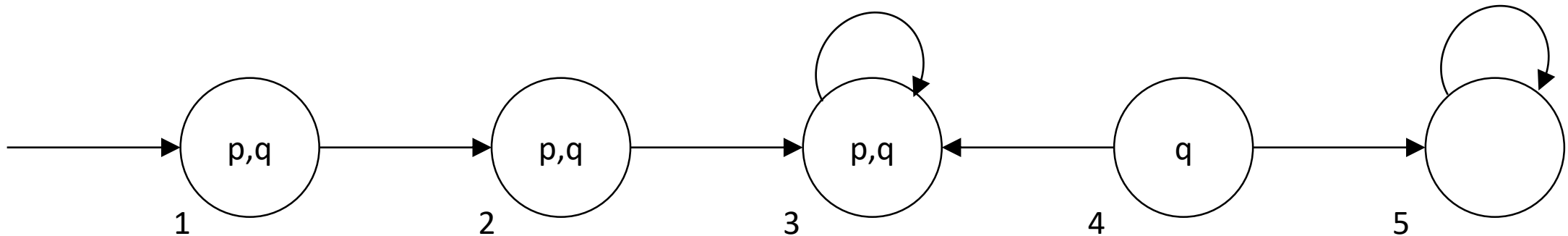
Conclusion: $\forall n. Q(n)$

In our setting:

Base. all paths of length k from S_0 are labeled q

Induction. all paths of length k labeled with all qs are followed by a q

Conclusion. All paths from S_0 are labeled q



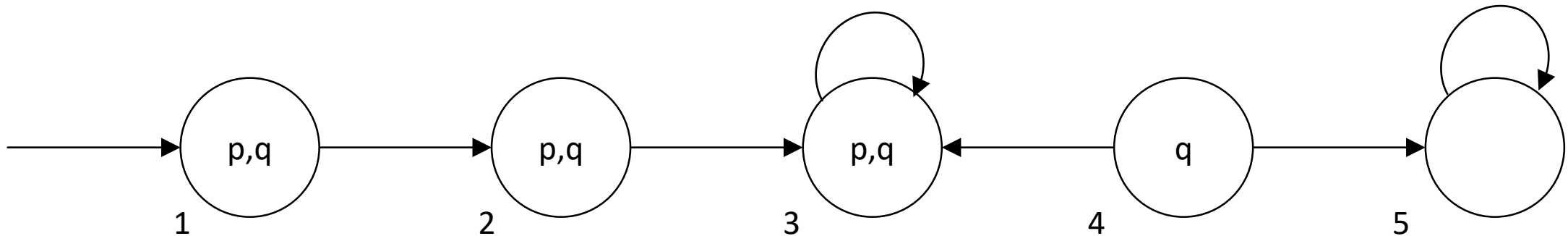
Prove $AG\ q$ using 2-induction

Base: Consider all paths of length 2 from q_1 : $q_1 \models q$ and $q_2 \models q$.

Induction: Do all successors of paths of length 2 labeled q, q fulfill q ?

- (q_1, q_2)
- (q_2, q_3)
- (q_3, q_4)
- (q_4, q_4)

Conclusion: for any path π we have $\forall n. \pi(n) \models p$



k-induction as Satisfiability

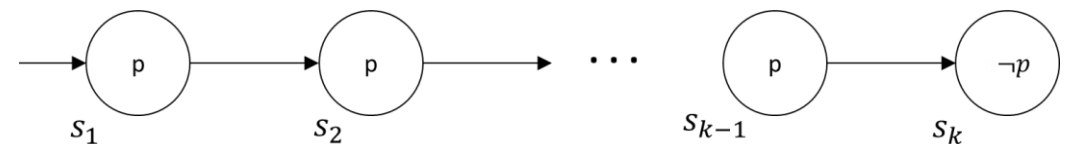
Base. all paths of length k from S_0 are labeled p

This is BMC! $S_0(s_0) \wedge \bigwedge_{i=0}^{k-2} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k-1} \neg p(s_i)$

Induction. all paths of length k labeled with all p 's are followed by a p

$$\bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{k-1} p(s_i) \wedge \neg p(s_k)$$

Formula satisfiable iff there is a counterexample



k-induction

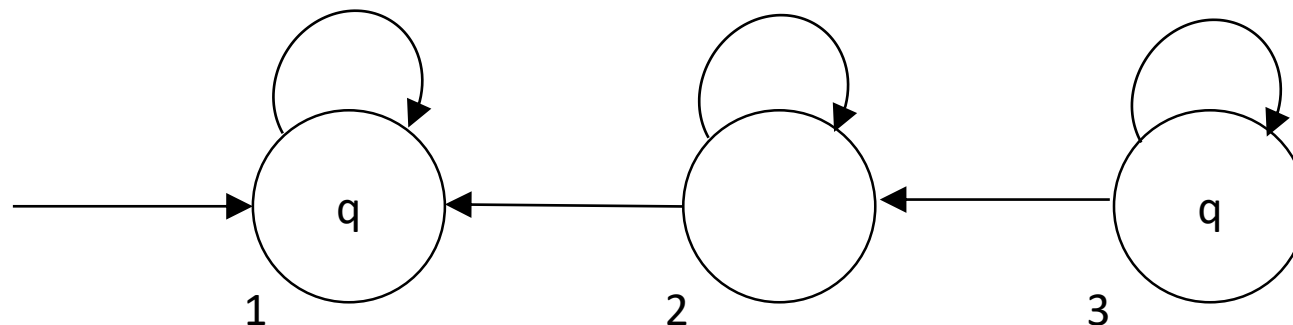
```
while(k=0; ; k++){  
    build BMC formula  $\phi$   
    if  $\phi$  SAT return “bug!”  
  
    build k-induction formula  $\psi$   
    if  $\phi$  UNSAT return “correct!”  
}
```


k-induction is not Complete

System satisfies **AG** q , but induction step fails for any k

Base. all paths of length k from S_0 are labeled q

Induction. all paths of length k labeled with all qs are followed by a q . **FALSE**

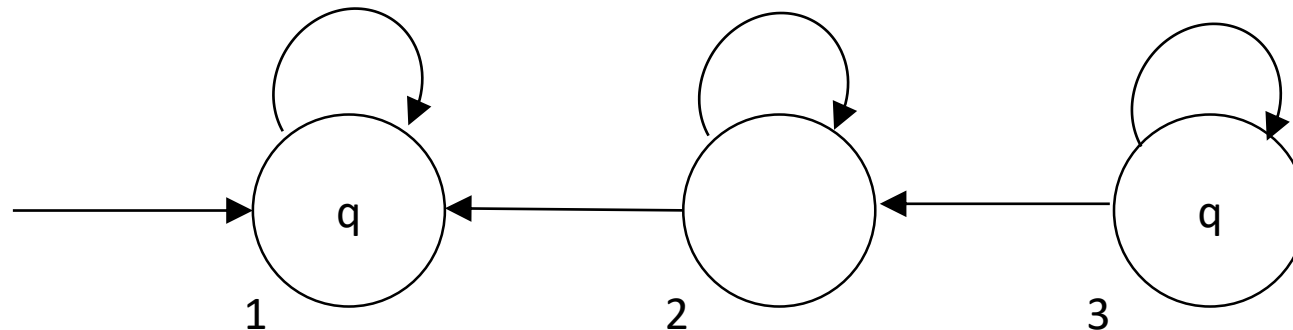


Making k-induction Complete

Induction. all **noncyclic** paths of length k labeled with all qs are followed by a q

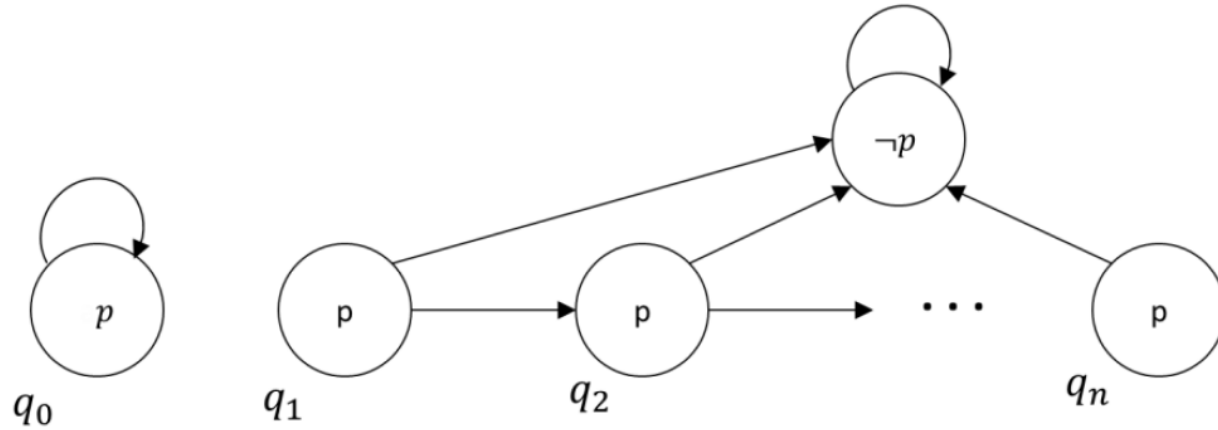
$$\bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{k-1} p(s_i) \wedge \neg p(s_k) \wedge$$

$$\bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^k s_i \neq s_j$$



Consider the following synchronous Kripke structure K .

HW2



We wish to prove that p is always true.

Task 2a. [5 points]

Suppose that q_1 is the initial state. Suppose you are given formulas R, S_0 , and p for the transition relation, the initial states and the property, resp.

- What is the smallest k such that BMC finds a counterexample?
- Show the BMC formula, using R, S_0 , and p .
- Is the formula satisfiable? Explain.

Task 2c. [5 points]

Suppose that q_0 is the initial state. The new formula for the initial states is S'_0 .

- What is the smallest k such that k -induction can prove the property correct?
- Suppose $n=2$. Choose an appropriate k and show the k -induction formula, using R, S'_0 , and p .
- Is the formula satisfiable? Explain.

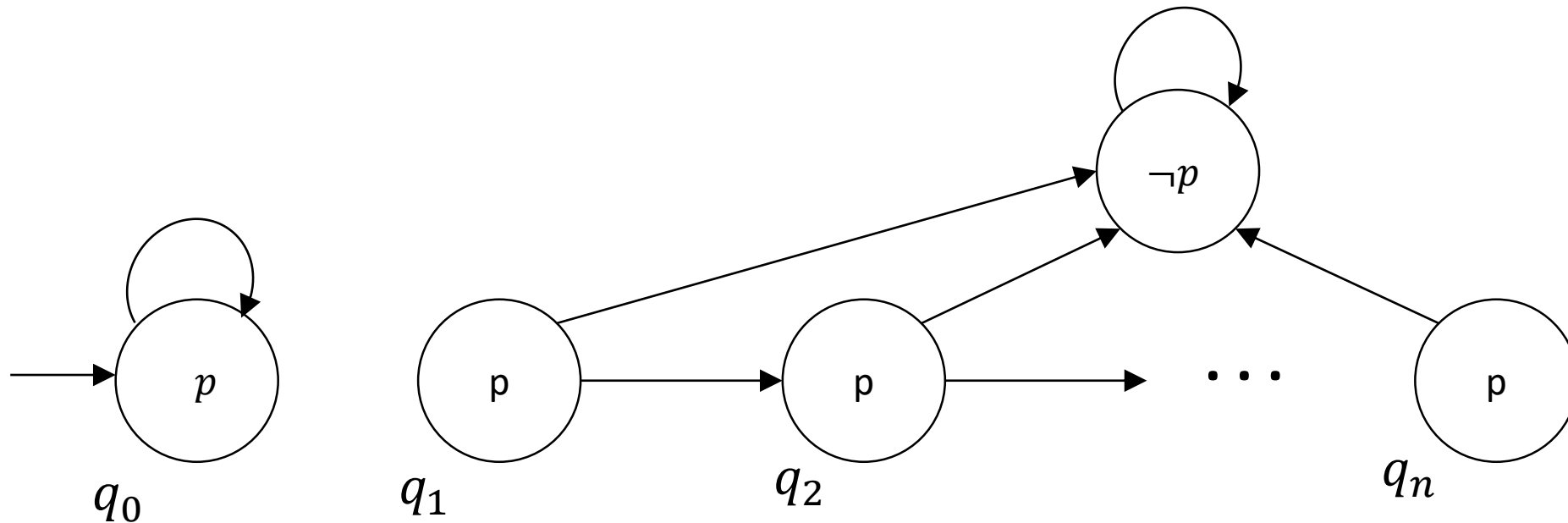
Model Checking with Inductive Invariants

Problems with k -induction

Problem: Sometimes k is very large

In the following machine, you need $k = n$ to prove that $\mathbf{AG} p$ holds.

Idea: Automatically find better inductive invariants.



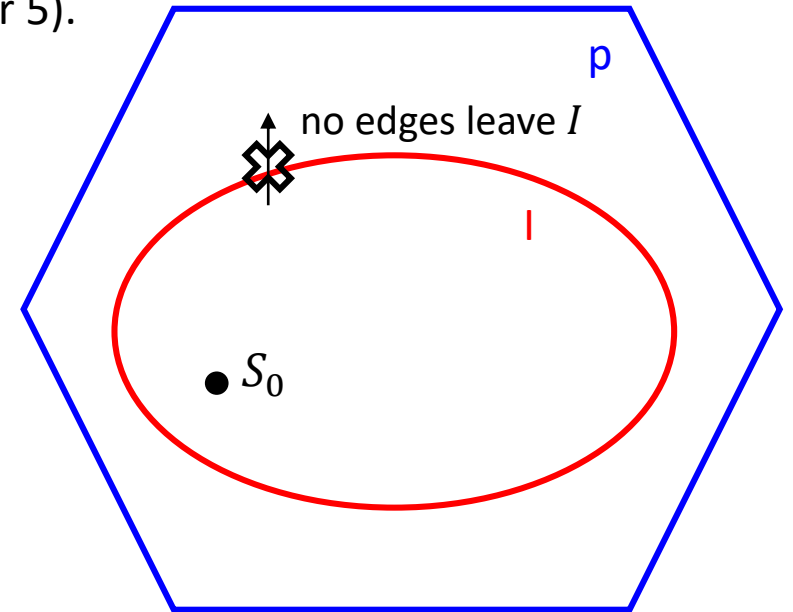
Inductive Invariant

Remember $postimage(Q) = \{s' \mid \exists s. R(s, s')\}$ (see Chapter 5).

Definition. $I \subseteq S$ is an **inductive invariant** for $AG\ p$ if

1. $S_0 \subseteq I$
2. $postimage(I) \subseteq I$
3. $\forall s \in I. s \models p$

If there is an inductive invariant for $AG\ p$, then $AG\ p$ holds.



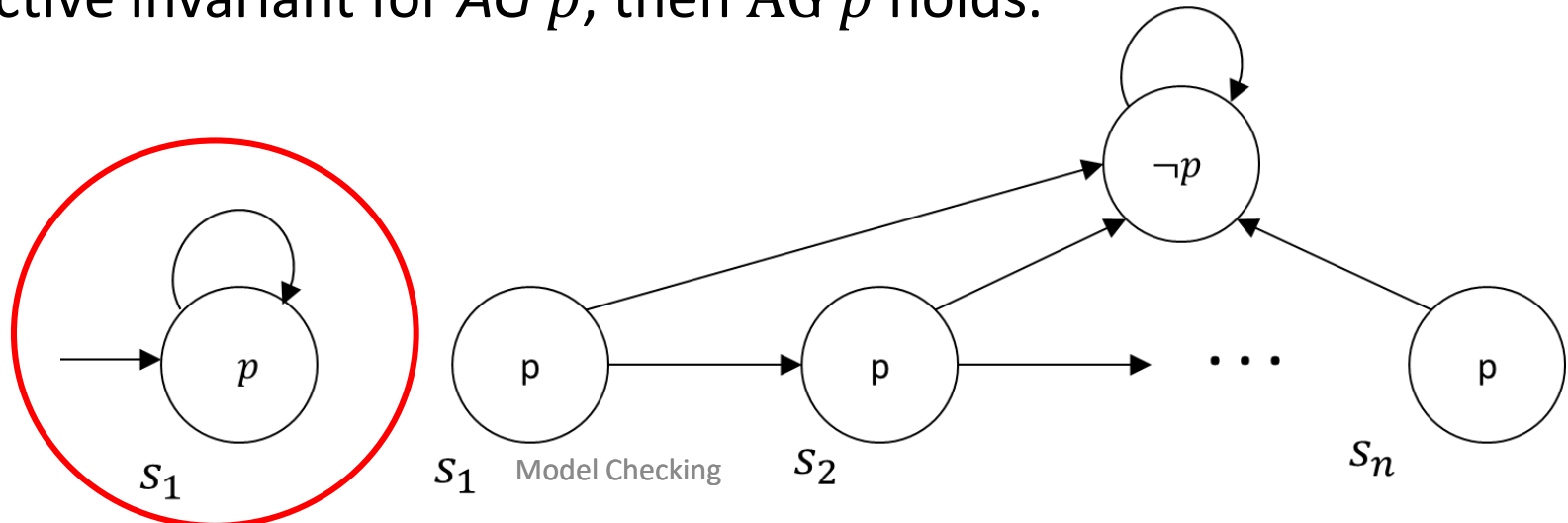
Inductive Invariant

Remember $postimage(Q) = \{s' \mid \exists s. R(s, s')\}$ (see Chapter 5).

Definition. $I \subseteq S$ is an **inductive invariant** for $AG\ p$ if

1. $S_0 \subseteq I$
2. $postimage(I) \subseteq I$
3. $\forall s \in I. s \models p$

If there is an inductive invariant for $AG\ p$, then $AG\ p$ holds.



Model Checking with Craig Interpolants

Ken McMillan, 2003

2010 CAV Award: “has significantly influenced both academic research and industrial practice, and has dramatically changed the scale of systems that can be analyzed by model checking.”



Kenneth McMillan

Interpolants as Inductive Invariants

- BMC finds bugs (and absence of bugs up to k steps)
- How to Show Correctness?
 - k -induction
 - Interpolants
- Find Interpolants I such that
 - States reachable in k steps are in I
 - no bad states are in I
- Interpolants are (good) overapproximation of post-image computation

Interpolant



William Craig, 1957

Definition. Given formulas A, B such that $A \wedge B = \perp$, an **interpolant** is a formula I such that

1. $A \rightarrow I$
2. $I \wedge B \equiv \perp$
3. I only uses symbols that occur both in A and in B

Example. Let

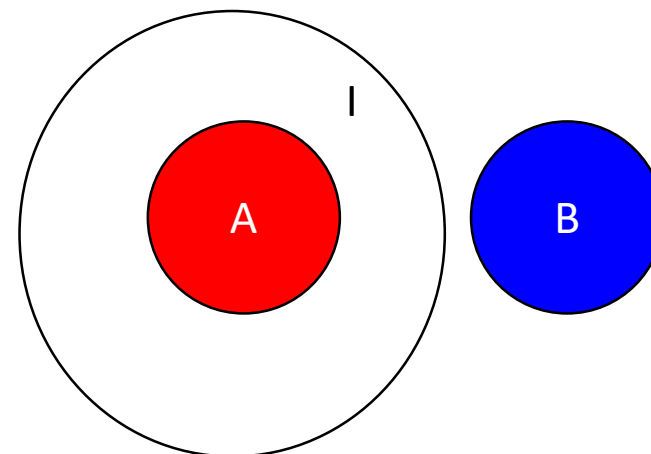
$$A = (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge a_2,$$

$$B = (\neg a_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \neg a_4.$$

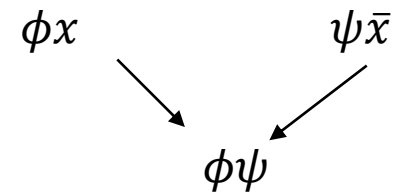
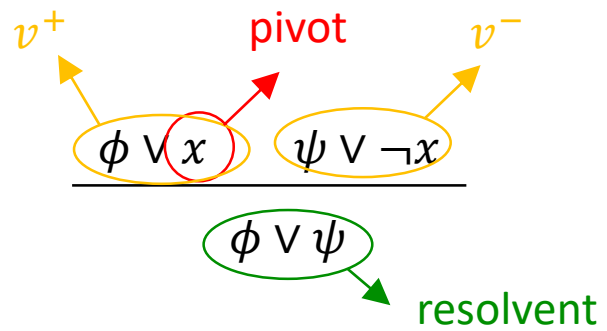
$A \wedge B$ is not satisfiable.

$\neg a_3 \wedge a_2$ is an interpolant:

1. $((a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge a_2) \rightarrow (\neg a_3 \wedge a_2)$
2. $(\neg a_3 \wedge a_2) \wedge ((\neg a_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \neg a_4) \equiv \perp$
3. a_2 and a_3 occur in A and in B



Resolution (Chap 9)



Interpolants from Resolution Proofs

For clause C , $C|B$ is obtained by removing literals not in B

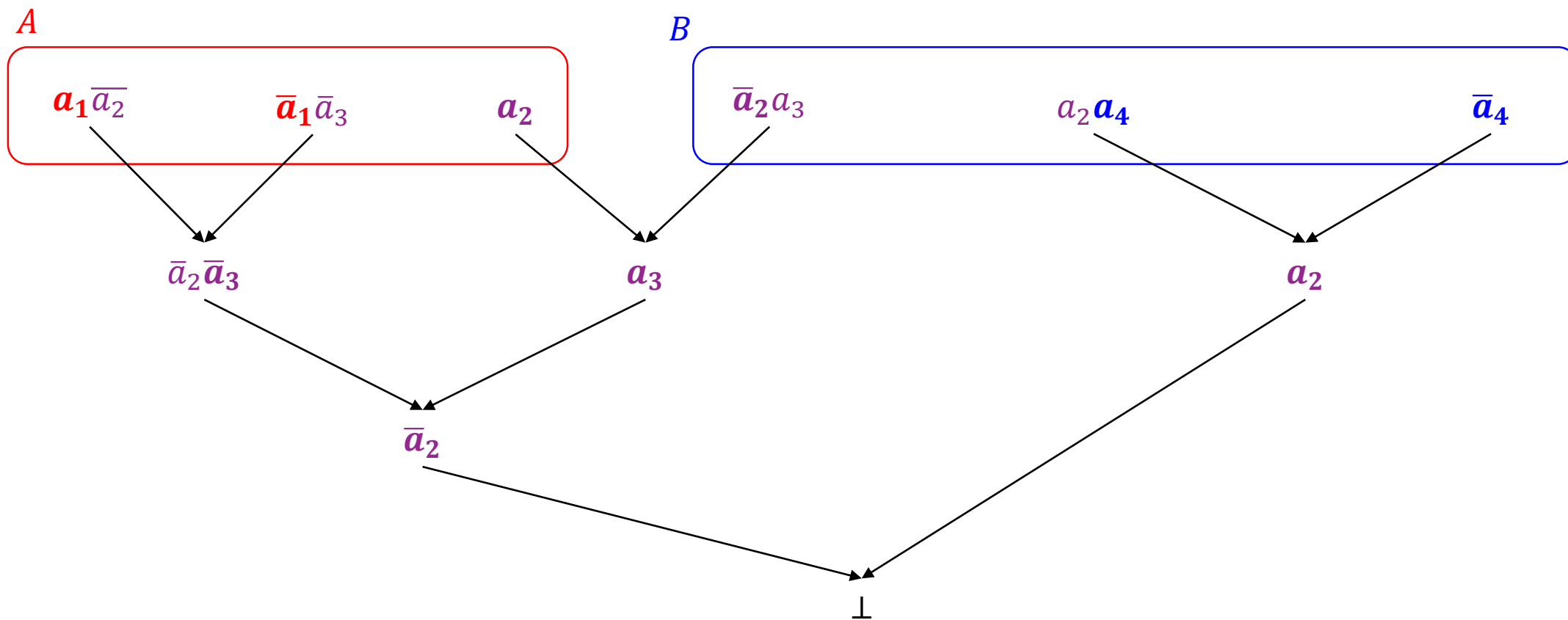
Algorithm. Go through resolution proof top-down.

1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$

Interpolation Example

Algorithm. Go through resolution proof top-down.

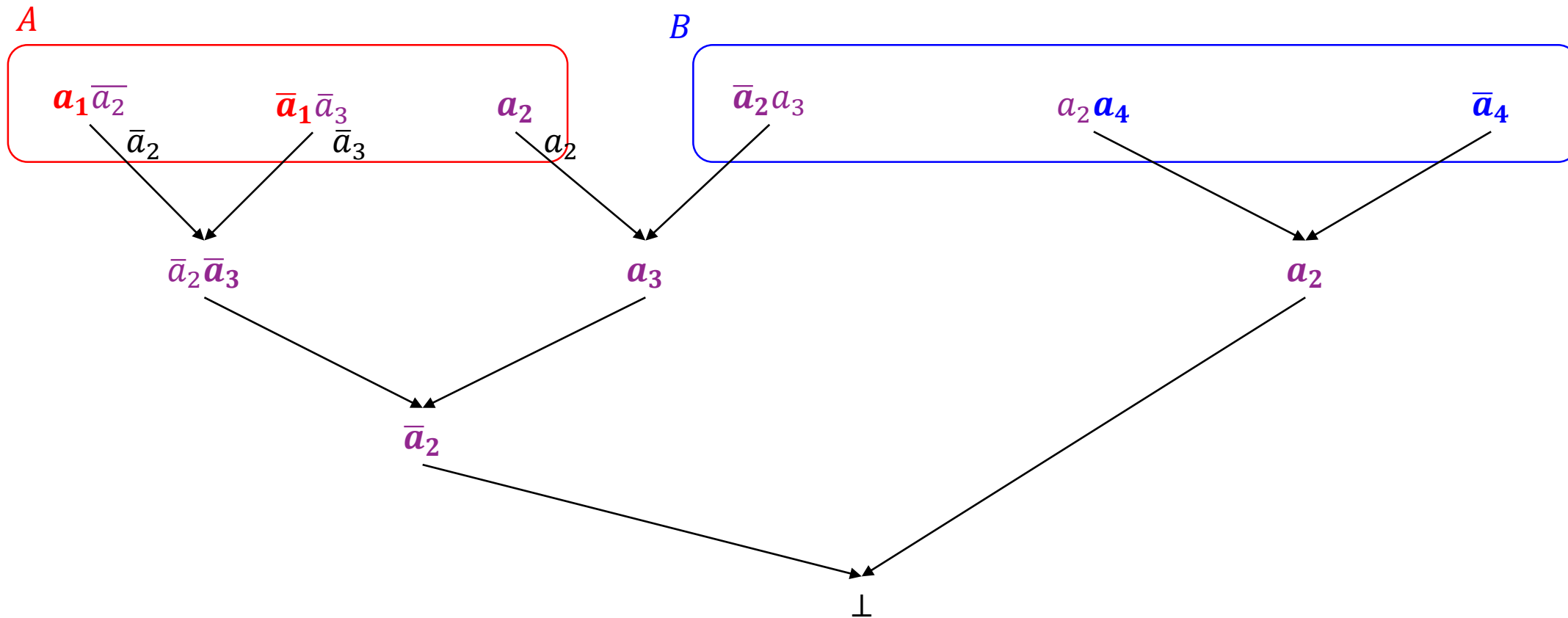
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

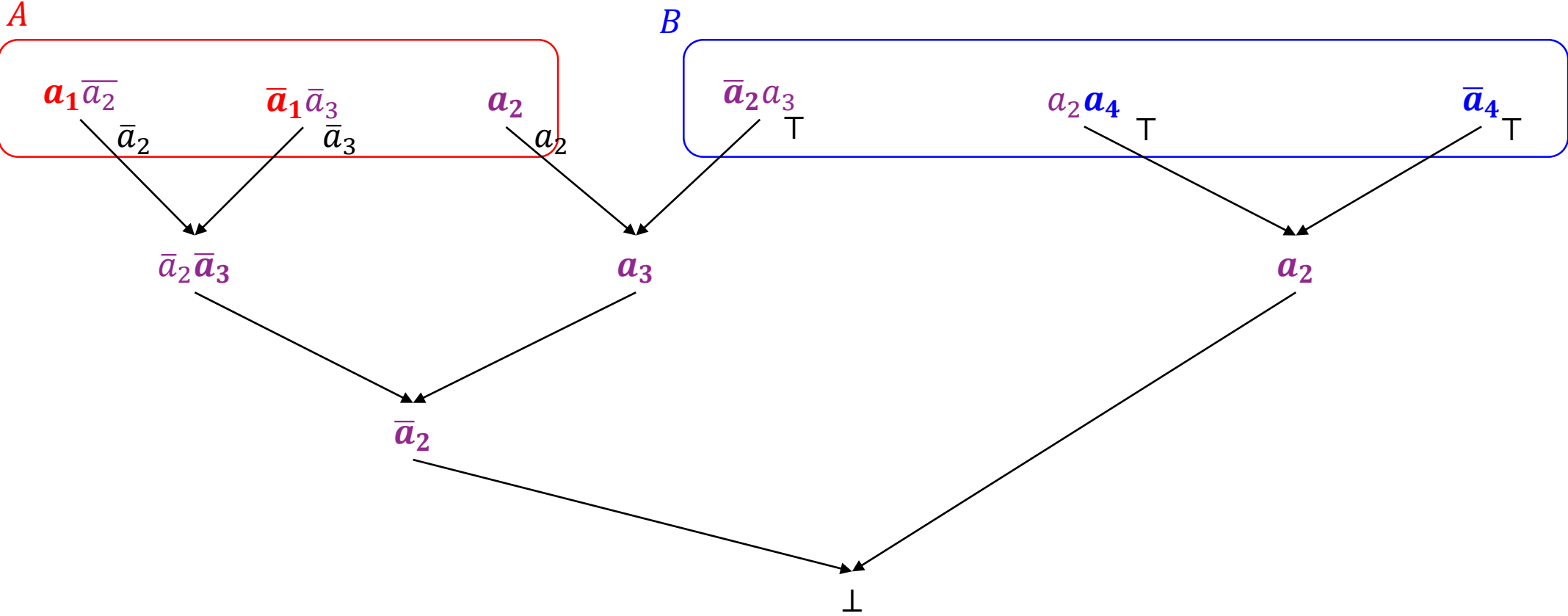
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

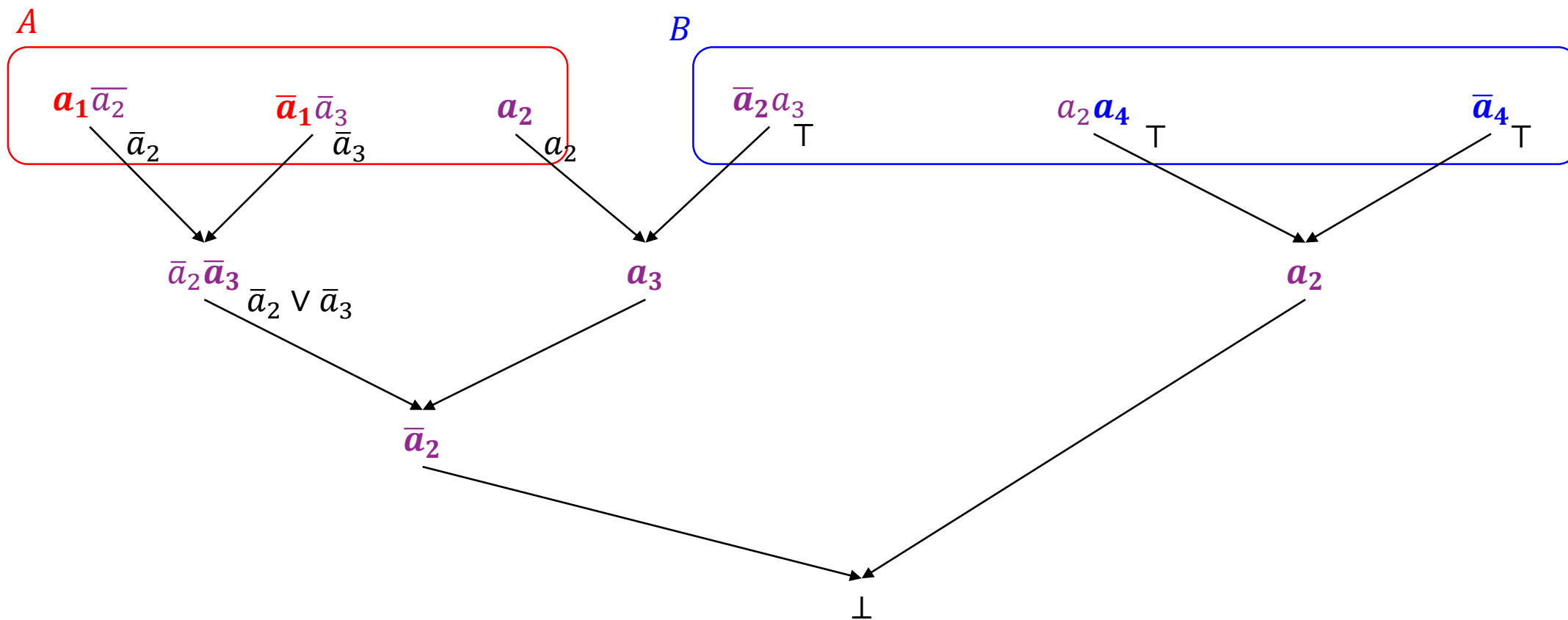
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

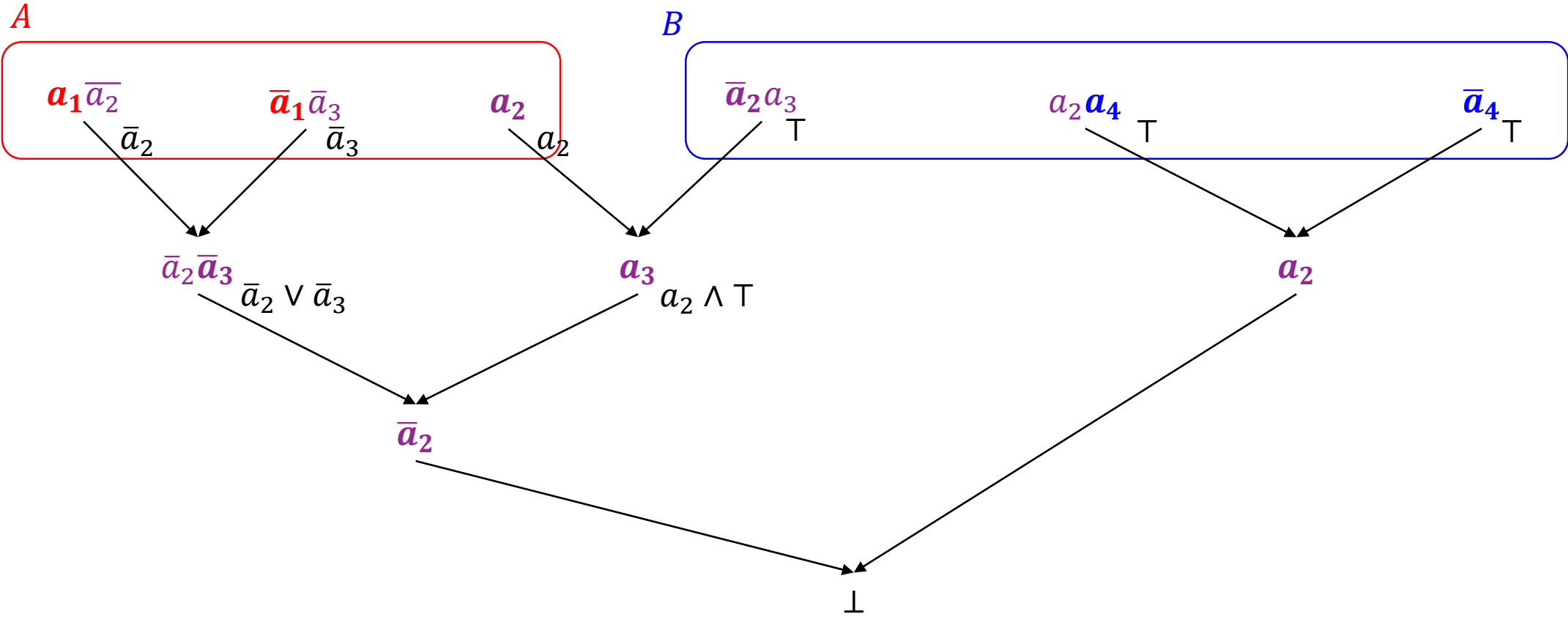
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

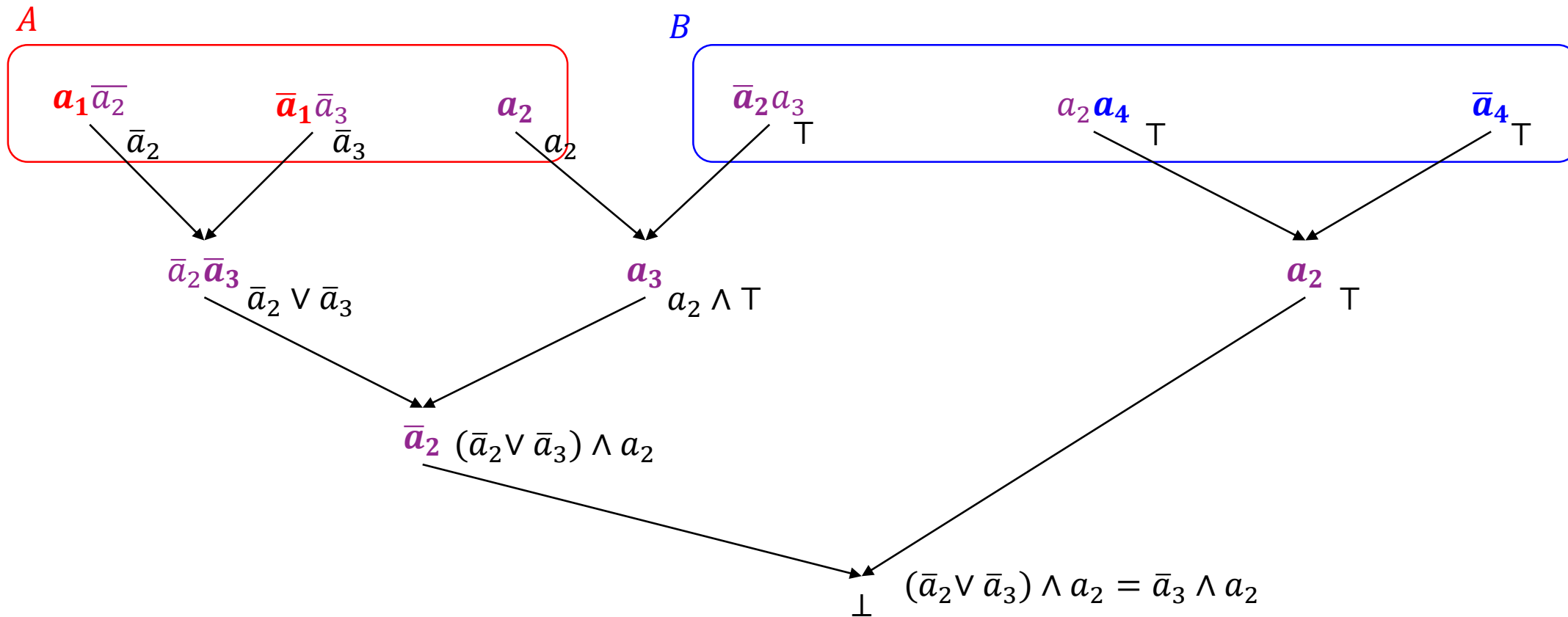
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Reachability Checking with Interpolation

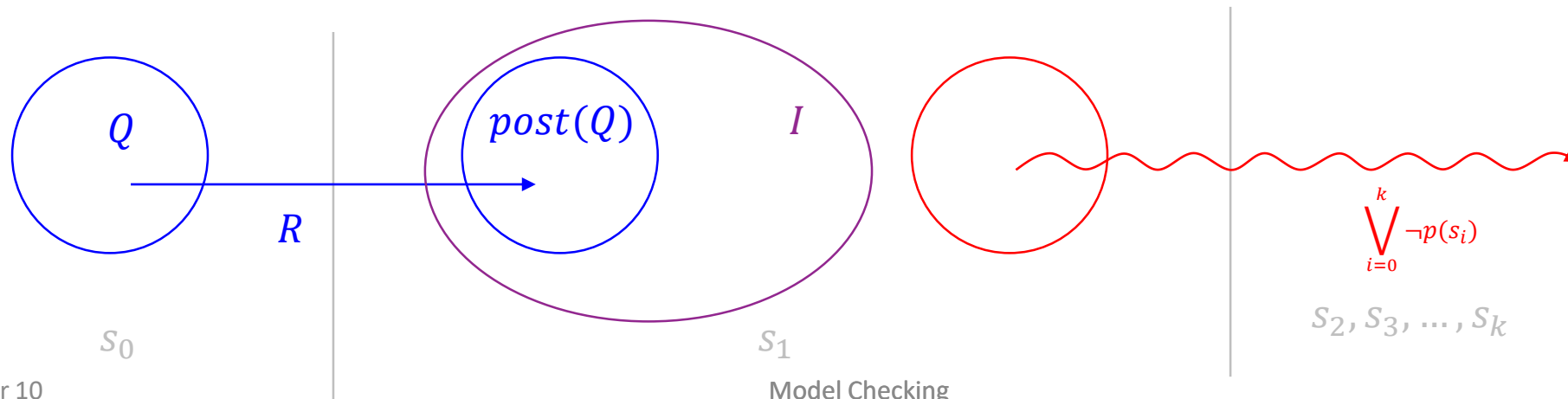
Recall BMC check for $\neg \mathbf{AG}p$:

$$S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^k \neg p(s_i).$$

Let's start from Q and assume $Q \models p$

$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

Suppose ϕ unsatisfiable, $I(s_1)$ is an interpolant



Reachability Checking with Interpolation

Recall BMC check for $\neg \mathbf{AG}p$:

$$S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^k \neg p(s_i).$$

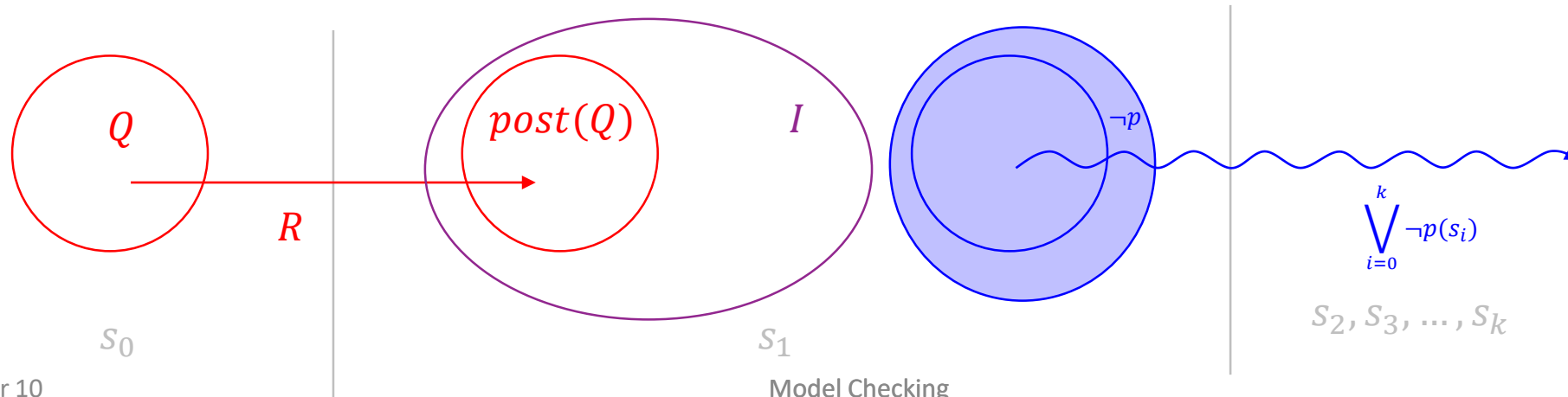
Let's start from Q and assume $Q \models p$

$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

Suppose ϕ unsatisfiable, $I(s_1)$ is an interpolant.

Note 1: $B \rightarrow \neg p(s_1)$
so $I(s_1) \wedge \neg p(s_1) = \perp$

Note 2: $I \supseteq \text{post}(Q)$



Interpolant Reachability Idea

$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^k \neg p(s_i).$$

1. If ϕ is satisfiable, $\neg p$ is **reachable**
2. Interpolant is approximation to post-image, like in BDD-based model checking.
 - If $\neg p$ is never part of I , it **cannot be reached**
3. If $\neg p$ is eventually part of I , increase k to increase precision of approximation.

Procedure terminates when k is diameter of system (or earlier!)

Algorithm

procedure CraigReachability(model M , $p \in AP$)

if $S_0 \wedge \neg p$ is SAT return " $M \not\models AG p$ ";

$k := 1$;

$Q := S_0(s_0)$;

while true **do**

$A := Q(s_0) \wedge R(s_0, s_1)$;

$B := \bigvee_{i=1}^k R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i)$;

if $A \wedge B$ is SAT **then**

if $Q = S_0$ then return " $M \not\models AG p$ ";

 Increase k

$Q := S_0(s_0)$;

else

 compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return " $M \models AG p$ ";

$Q := Q \vee I(s_0)$;

end if

end while

end procedure

// $\neg p$ can be reached from Q

// $\neg p$ can be reached from S_0

// Not sure if path to $\neg p$ is real. Increase precision

// Reached the fixpoint of overapproximated reachability?

// Another step of overapproximated reachability?

if $A \wedge B$ is SAT **then**

if $Q = S_0$ then return “ $M \not\models \text{AG } p$ ”;

 increase k

$Q := S_0(s_0)$;

else

 compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return “ $M \models \text{AG } p$ ”;

$Q := Q \vee I(s_0)$;

end if

10.4.4 Correctness

If CraigReachability returns “ $M \models AG p$ ” then $M \models AG p$

Let Q_i denote Q at iteration i . For all i , $Q_i \leftarrow postimage^i(Q_0)$. If $I \rightarrow Q_i$, we have reached a fixedpoint $Q^* = Q_i$ so $Q^* \leftarrow postimage^*(Q_0)$. Now because $Q_i \wedge \neg p = \perp$, we have $postimage^*(Q_0) \wedge \neg p = \perp$.

If CraigReachability returns “ $M \not\models AG p$ ” then $M \not\models AG p$

$A \wedge B$ encodes a path from Q_0 to $\neg p$.

CraigReachability terminates

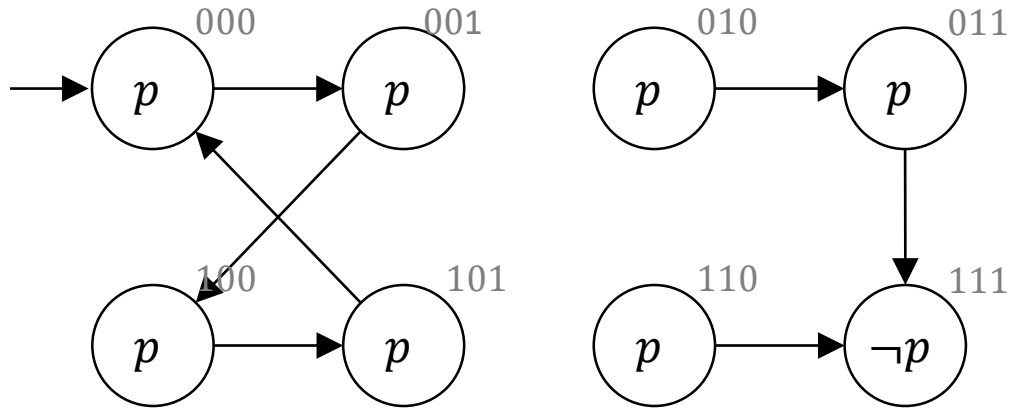
Note that k increases.

If $M \not\models AG p$, there is a path of length l to $\neg p$ and we will find it when $l = k$.

Suppose $M \models AG p$. If k is the diameter of the graph, no I and thus no Q_i can contain a state that reaches $\neg p$. Thus, $A \wedge B$ is never SAT and the algorithm terminates because the Q_i cannot grow forever.

$x_1x_2x_3$

Example $AG p$



$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

if $A \wedge B$ is SAT then

if $Q = S_0$ then return “ $M \not\models AG p$ ”;

 increase k

$Q := S_0(s_0)$;

else

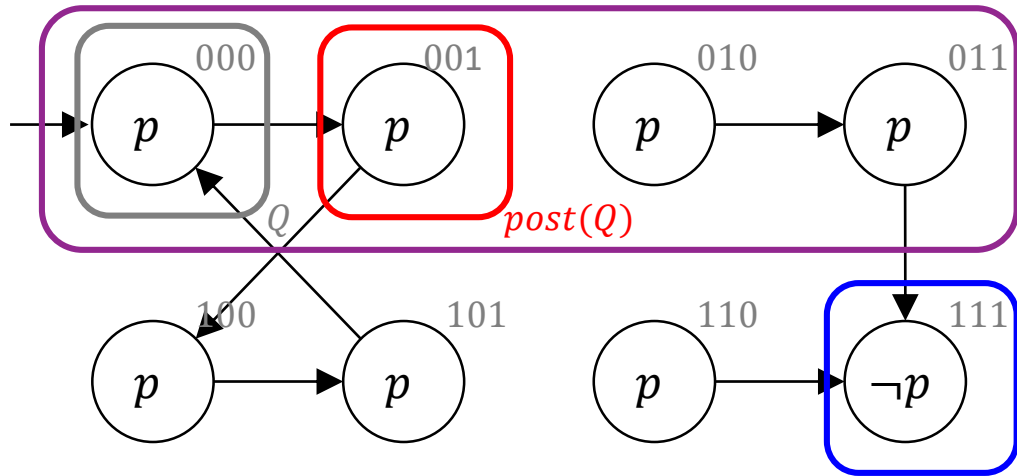
 compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return “ $M \models AG p$ ”;

$Q := Q \vee I(s_0)$;

$x_1x_2x_3$

Example $AG p$



$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

$$k = 1.$$

$$Q = \neg x_1 \wedge \neg x_2 \wedge \neg x_3 = \{000\}.$$

ϕ is UNSAT

Invariant checks first bit: $I = \neg x_1$

if $A \wedge B$ is SAT then

if $Q = S_0$ then return " $M \not\models AG p$ ";

increase k

$Q := S_0(s_0)$;

else

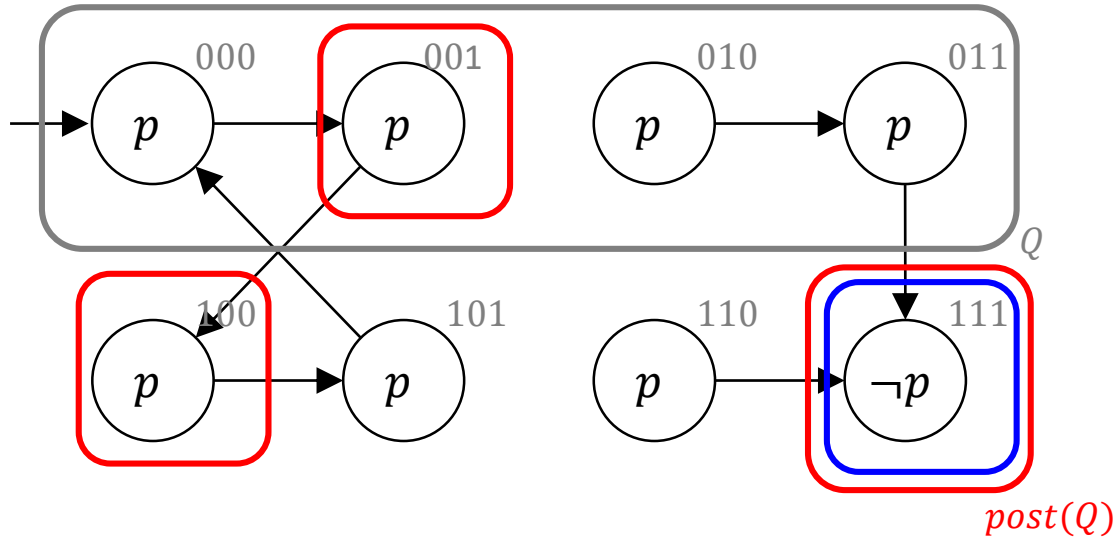
compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return " $M \models AG p$ ";

$Q := Q \vee I(s_0)$;

$x_1x_2x_3$

Example $AG p$



$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

$$k = 1.$$

$$Q = \neg x_1 = \{000, 001, 010, 011\}.$$

ϕ is SAT

if $A \wedge B$ is SAT then

if $Q = S_0$ then return " $M \not\models AG p$ ";

increase k

$Q := S_0(s_0)$;

else

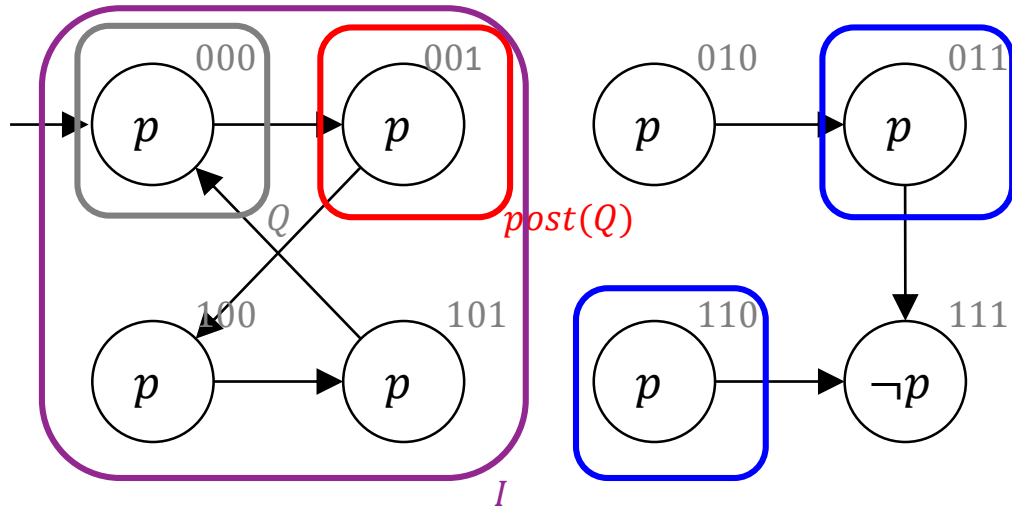
compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return " $M \models AG p$ ";

$Q := Q \vee I(s_0)$;

$x_1x_2x_3$

Example $AG p$



$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

$$k = 2.$$

$$Q = \neg x_1 \wedge \neg x_2 \wedge \neg x_3 = \{000\}.$$

ϕ is UNSAT

Invariant checks 2nd bit: $I = \neg x_1$

if $A \wedge B$ is SAT then

if $Q = S_0$ then return " $M \not\models AG p$ ";

increase k

$Q := S_0(s_0)$;

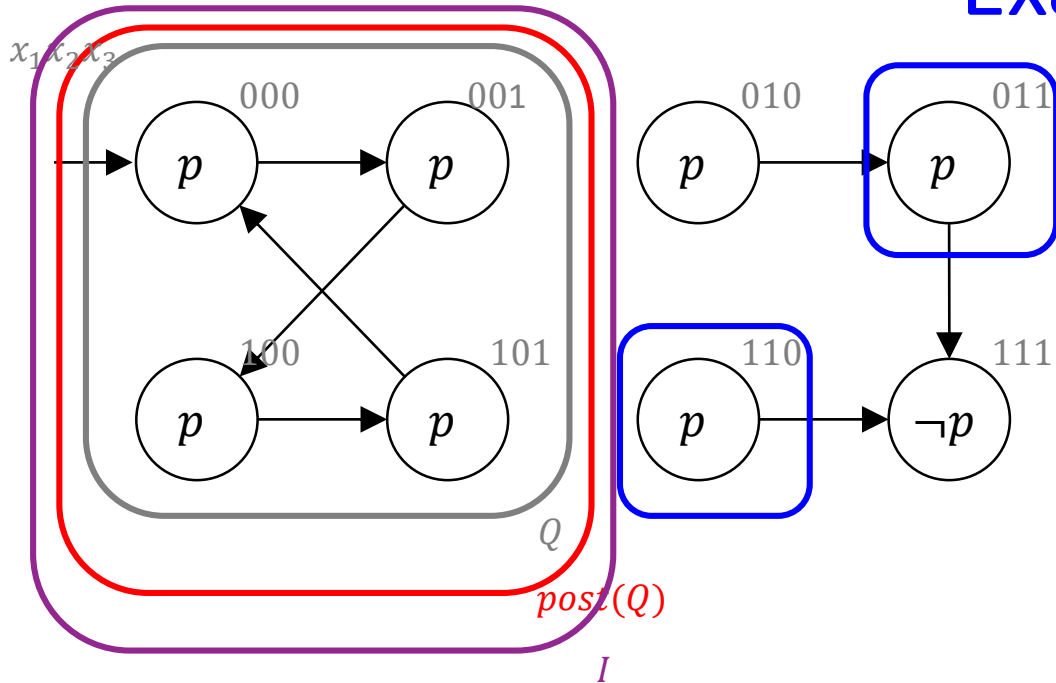
else

compute interpolant I for A and B ;

if $I(s_0) \rightarrow Q$ then return " $M \models AG p$ ";

$Q := Q \vee I(s_0)$;

Example $AG p$



$$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^k \neg p(s_i).$$

$$k = 2.$$

$$Q = \neg x_2 = \{000, 001, 100, 101\}$$

ϕ is UNSAT

$$I = \neg x_2 = Q.$$

Algorithm terminates.

if $A \wedge B$ is SAT then

 if $Q = S_0$ then return " $M \not\models AG p$ ";

 increase k

$Q := S_0(s_0)$;

else

 compute interpolant I for A and B ;

 if $I(s_0) \rightarrow Q$ then return " $M \models AG p$ ";

$Q := Q \vee I(s_0)$;

Property-Directed Reachability



Aaron Bradley

PDR

Property-Directed Reachability or IC3

- Makes no copies of transition relation – memory efficient
- Overapproximate postimage (like interpolation)

Frames

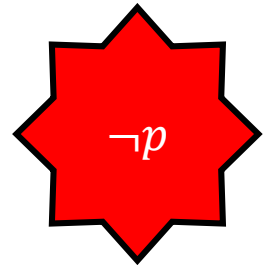
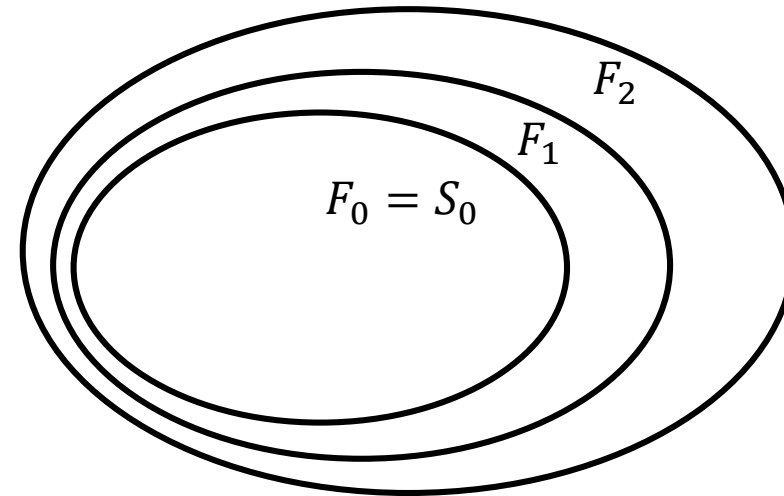
The **frames** F_0, \dots, F_k are a subset of states such that

Inv1. $S_0 \subseteq F_0$

Inv2. $F_i \subseteq F_{i+1}$

Inv3. $\forall s \in F_i. s \models p$

Inv4. $postimage(F_i) \subseteq F_{i+1}$



Frames

The **frames** F_0, \dots, F_k are a subset of states such that

Inv1. $S_0 \subseteq F_0$

Inv2. $F_i \subseteq F_{i+1}$

Inv3. $\forall s \in F_i. s \models p$

Inv4. $postimage(F_i) \subseteq F_{i+1}$

If $F_i = F_{i+1}$ then F_i an inductive invariant.

Proof:

1. $S_0 \subseteq F_0 \subseteq \dots \subseteq F_i$
2. $postimage(F_i) \subseteq F_{i+1} = F_i$
3. $\forall s \in F_i. s \models p$

Definition. $I \subseteq S$ is an **inductive invariant** for $AG p$ if

1. $S_0 \subseteq I$
2. $postimage(I) \subseteq I$
3. $\forall s \in I. s \models p$

PDR

```
function PDR(model  $M$ ,  $p \in AP$ )  
  if  $s_0 \wedge \neg p$  is SAT return “ $M \not\models \mathbf{AG} p$ ”;  
   $F_0 := S_0$ ;  $k := 0$ ; // Inv1 – Inv4 hold  
  while true do  
    extendFrontier( $M$ ,  $k$ );  
    propagateClauses( $M$ ,  $k$ );  
    if  $\exists i. F_i = F_{i+1}$  then return “ $M \models \mathbf{AG} p$ ”;  
     $k := k + 1$ ;  
  end while  
end function
```

extendFrontier

function extendFrontier(model M , $k \in \mathbb{N}$)

$F_{k+1} := \{s \mid p \in L(s)\};$

while $F_k \wedge R \wedge \neg p'$ is SAT **do**

$s' :=$ state labeled $\neg p$ extracted from satisfying assignment

$s :=$ predecessor of s' extracted from satisfying assignment

 removeCTI(M, s, k) // removes s from F_{k+1}

end while

 // inv1 – inv 4 hold

end function

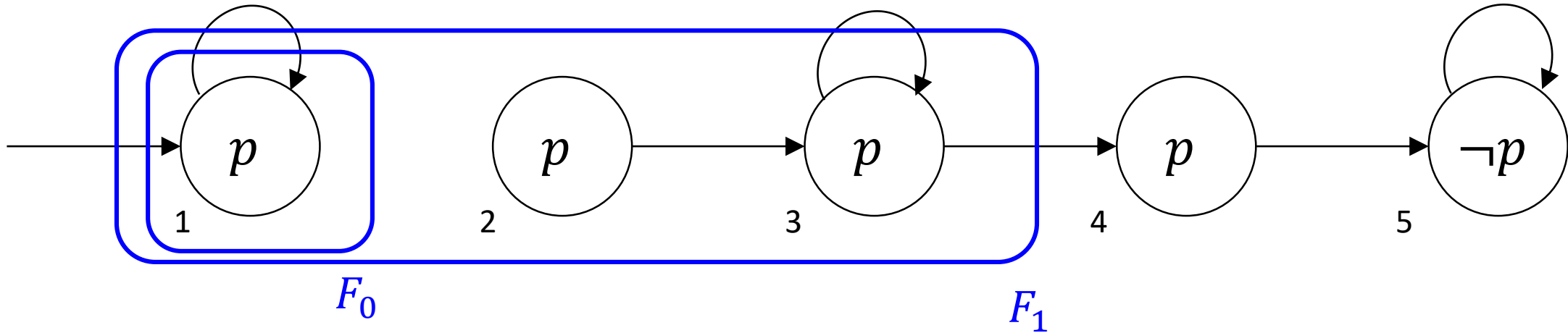
removeCTI

```
procedure removeCTI(model  $M$ ,  $s \in S$ ,  $i \in \mathbf{N}$ )  
  if  $S_0 \wedge s$  is SAT then abort “ $M \not\models \mathbf{AG} p$ ”;  
  while  $F_i \wedge R \wedge \neg s \wedge s'$  is SAT do  
    for  $l$  from 0 to  $i$   
       $F_l := F_l \wedge \neg s$   
    end for  
     $t :=$  predecessor of  $s$ , extracted from SAT witness  
    removeCTI( $M$ ,  $t$ ,  $i - 1$ )  
  end while  
end procedure
```

propagateClauses

```
procedure propagateClauses( $k \in \mathbf{N}$ )  
  for  $i$  from 1 to  $k$   
    for every clause  $c \in F_i$   
      if  $F_i \wedge T \wedge \neg c'$  then  
         $F_{i+1} := F_{i+1} \wedge c;$   
      end if  
    end for  
  end for  
end procedure
```

PDR Example: Property Holds

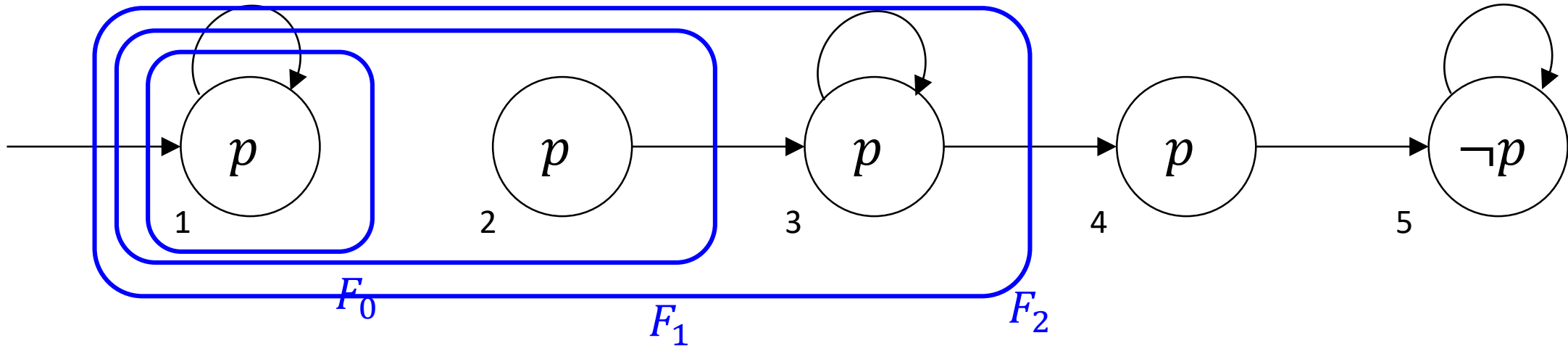


$k = 1$

```

function PDR(model  $M$ ,  $p \in AP$ )
  if  $s_0 \wedge \neg p$  is SAT return " $M \not\models \mathbf{AG} p$ ";
   $F_0 := S_0$ ;  $k := 0$ ;
  while true do
    extendFrontier( $M$ ,  $k$ );
    propagateClauses( $M$ ,  $k$ );
    if  $\exists i. F_i = F_{i+1}$  then return " $M \models \mathbf{AG} p$ ";
     $k := k + 1$ ; // Showing state here
  end while
end function
  
```

PDR Example: Property Holds

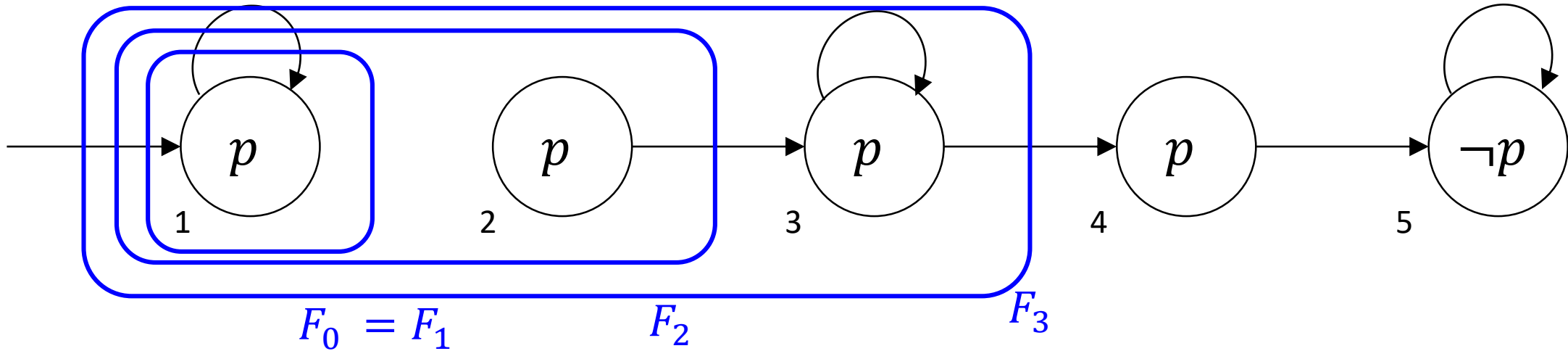


$k = 2$

```

function PDR(model  $M$ ,  $p \in AP$ )
  if  $s_0 \wedge \neg p$  is SAT return " $M \not\models \mathbf{AG} p$ ";
   $F_0 := S_0$ ;  $k := 0$ ;
  while true do
    extendFrontier( $M$ ,  $k$ );
    propagateClauses( $M$ ,  $k$ );
    if  $\exists i. F_i = F_{i+1}$  then return " $M \models \mathbf{AG} p$ ";
     $k := k + 1$ ; // Showing state here
  end while
end function
  
```

PDR Example: Property Holds



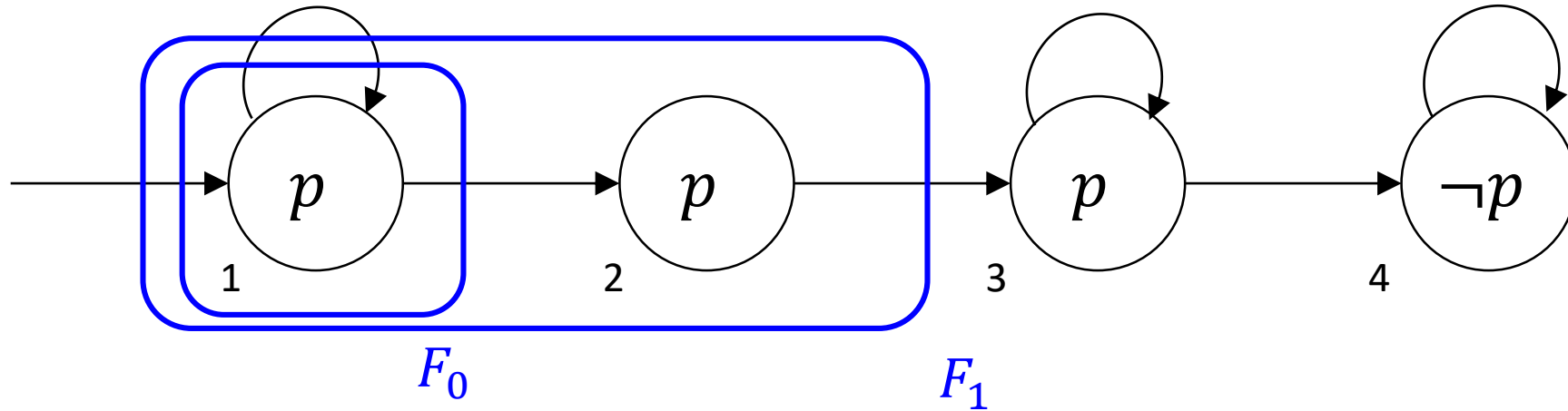
$k = 3$

$F_0 = F_1$. Property satisfied

```

function PDR(model  $M$ ,  $p \in AP$ )
  if  $s_0 \wedge \neg p$  is SAT return " $M \not\models \mathbf{AG} p$ ";
   $F_0 := S_0$ ;  $k := 0$ ;
  while true do
    extendFrontier( $M$ ,  $k$ );
    propagateClauses( $M$ ,  $k$ );
    if  $\exists i. F_i = F_{i+1}$  then return " $M \models \mathbf{AG} p$ ";
     $k := k + 1$ ; // Showing state here
  end while
end function
  
```

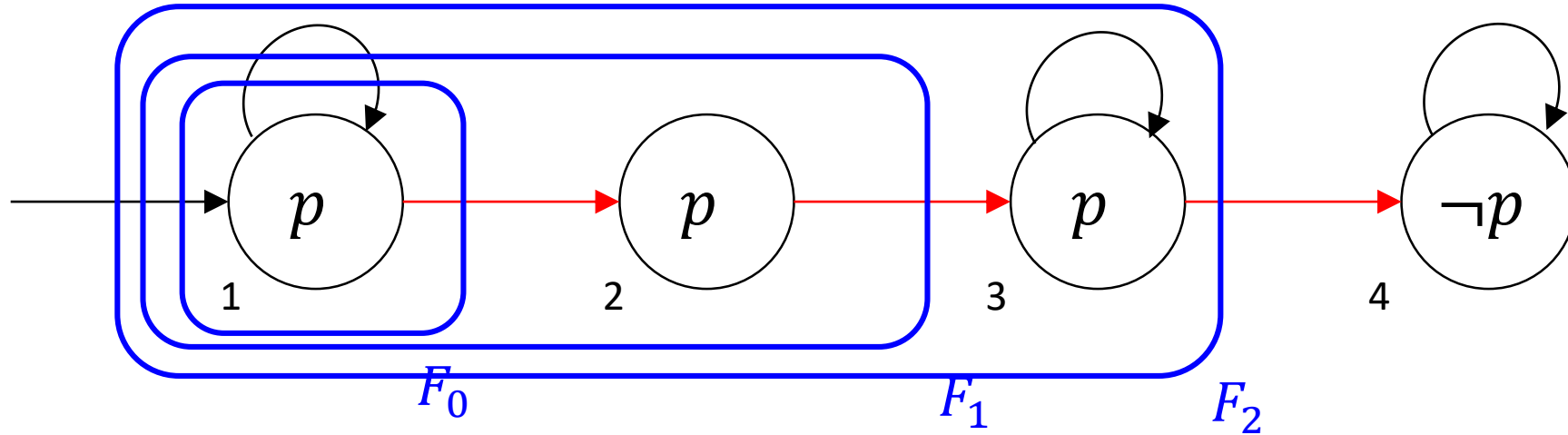

PDR Example: Property Violated



```

function PDR(model  $M$ ,  $p \in AP$ )
  if  $s_0 \wedge \neg p$  is SAT return " $M \not\models \mathbf{AG} p$ ";
   $F_0 := S_0$ ;  $k := 0$ ;
  while true do
    extendFrontier( $M$ ,  $k$ );
    propagateClauses( $M$ ,  $k$ );
    if  $\exists i. F_i = F_{i+1}$  then return " $M \models \mathbf{AG} p$ ";
     $k := k + 1$ ; // Showing state here
  end while
end function
  
```

PDR Example: Property Violated



$k = 1$

```

procedure removeCTI(model  $M$ ,  $s \in S$ ,  $i \in \mathbf{N}$ )
  if  $S_0 \wedge s$  is SAT then abort " $M \not\models \mathbf{AG} p$ ";
  while  $F_i \wedge R \wedge \neg s \wedge s'$  is SAT do
    for  $l$  from 0 to  $i$ 
       $F_l := F_l \wedge \neg s$ 
    end for
     $t :=$  predecessor of  $s$ , extracted from SAT witness
    removeCTI( $M$ ,  $t$ ,  $i - 1$ )
  end while
end procedure
  
```