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

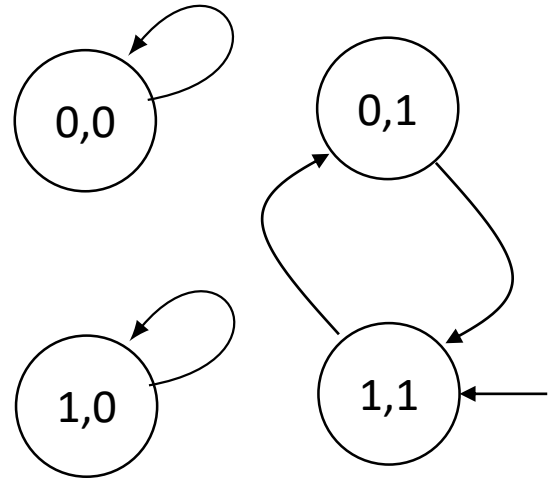
Preliminaries

Postimage

postimg(S) = states reachable from S in one step

postimg({ (0,1) }) =

postimg({ (0,0), (1,1)}) =



Kripke structure

Paths

Initial states:

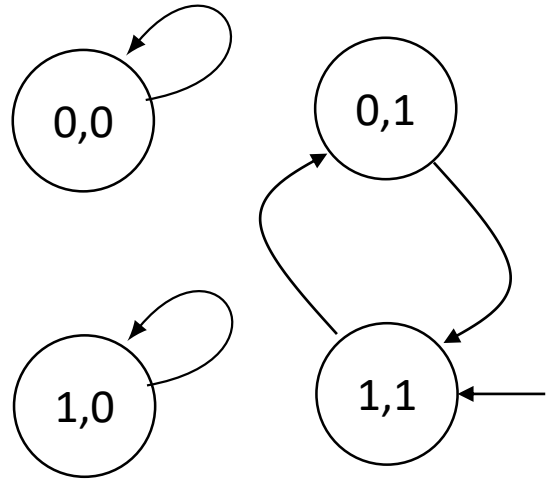
$$\mathcal{S}_0(x, y) = (x = 1 \wedge y = 1)$$

Transitions:

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

Paths:

$$Path_1(V, V') = \mathcal{S}_0(V) \wedge \mathcal{R}(V, V')$$



Kripke structure

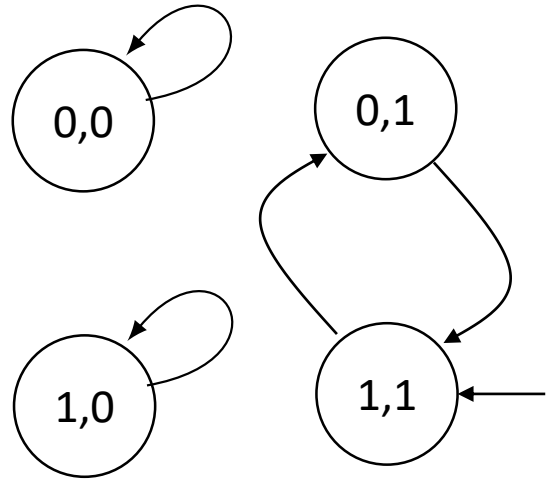
Paths

Initial states:

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

Transitions:

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



Kripke structure

Paths:

$$Path_1(V, V') = \mathcal{S}_0(V) \wedge \mathcal{R}(V, V') =$$
$$= x \wedge y \wedge (x' = (x + y)\ mod\ 2) \wedge (y' = y)$$

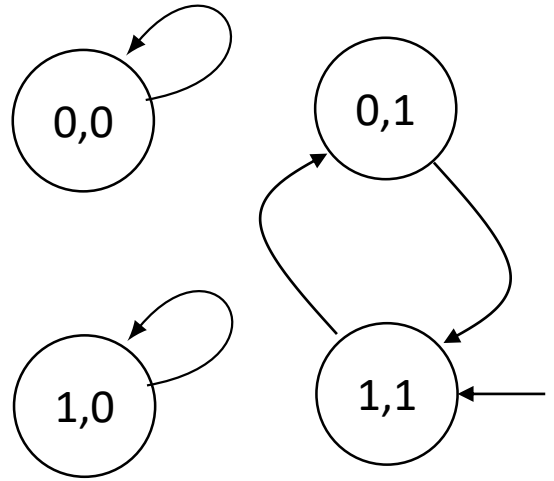
Paths

Initial states:

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

Transitions:

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



Kripke structure

Paths:

$$Path_1(V, V') = \mathcal{S}_0(V) \wedge \mathcal{R}(V, V') =$$
$$= x \wedge y \wedge (x' = (x + y)\ mod\ 2) \wedge (y' = y) =$$
$$= x \wedge y \wedge \neg x' \wedge y$$

Bounded Model Checking

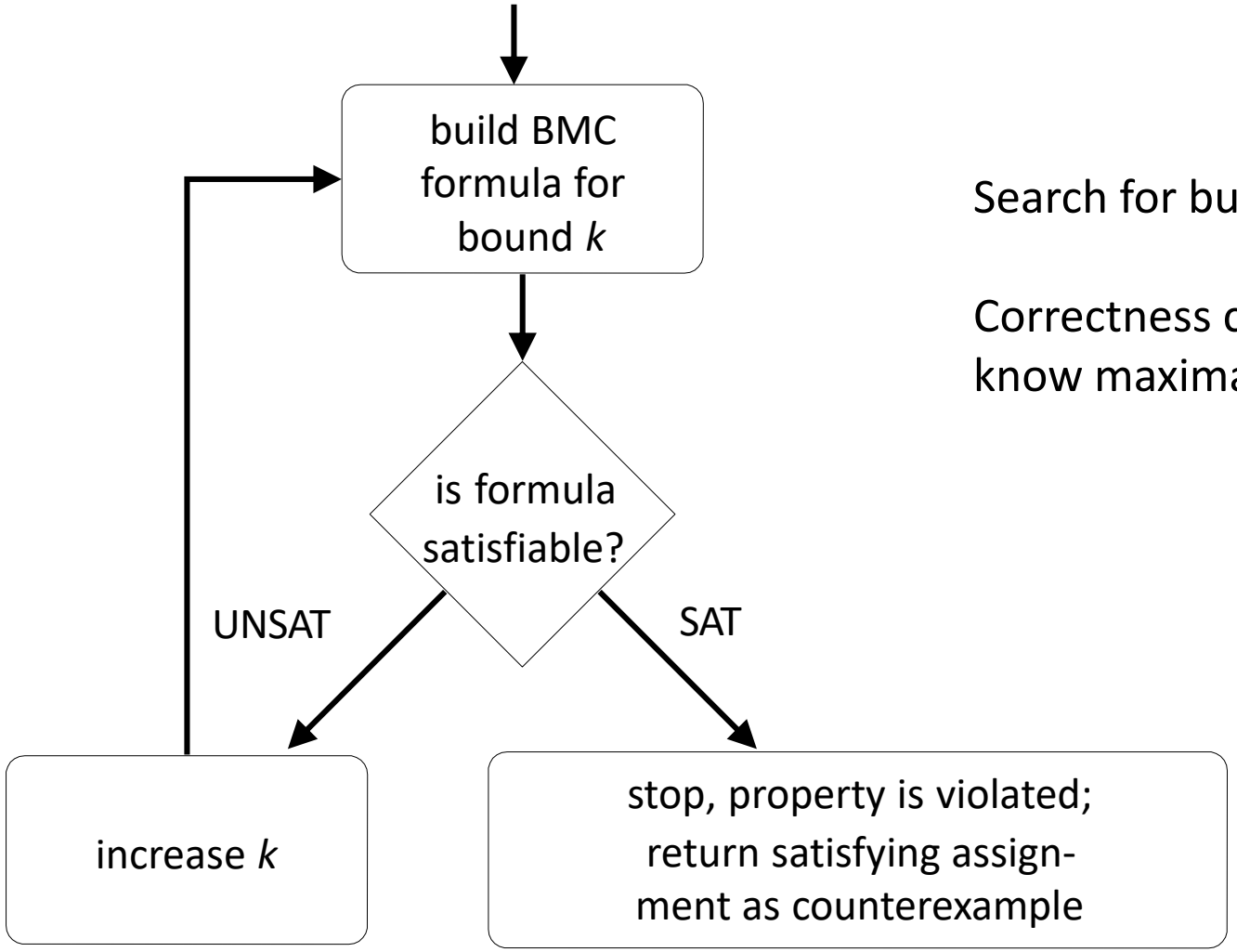
Computer Aided Verification Award 2018

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)

Flavio Lerda, Daniel Kroening, Armin Biere, Alessandro Cimatti,
Not pictured: Edmund M. Clarke, Yunshan Zhu

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 $\mathrm{V} = \{v_1, \ldots, v_n\}$

Reachability – Paths

$$path_0(s_0) =$$

$$path_1(s_0, s_1) =$$

$$path_2(s_0, \ldots, s_2) =$$

$$path_k(s_0, \ldots, s_k) =$$

Reachability – Paths

$$path_0(s_0) = S_0(s_0)$$
$$path_1(s_0, s_1) = S_0(s_0) \wedge R(s_0, s_1)$$
$$path_2(s_0, \ldots, s_2) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2)$$
$$path_3(s_0, \ldots, s_3) = S_0(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \wedge R(s_1, s_3)$$

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

Reachability – Building the Formula

Reachability – Building the Formula

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

Path starts in initial state

Exists transition from s_i to s_{i+1}

System is **incorrect** within k steps if

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

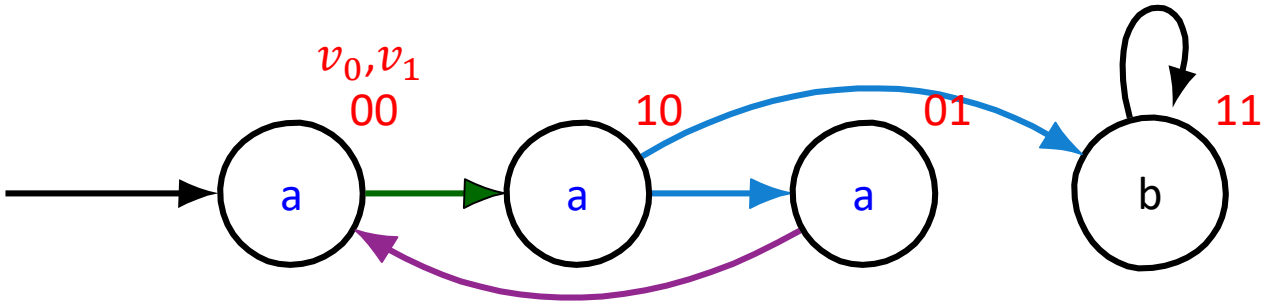
There is a path to s_k

One of the state violates p

Reachability – Correctness

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

Example



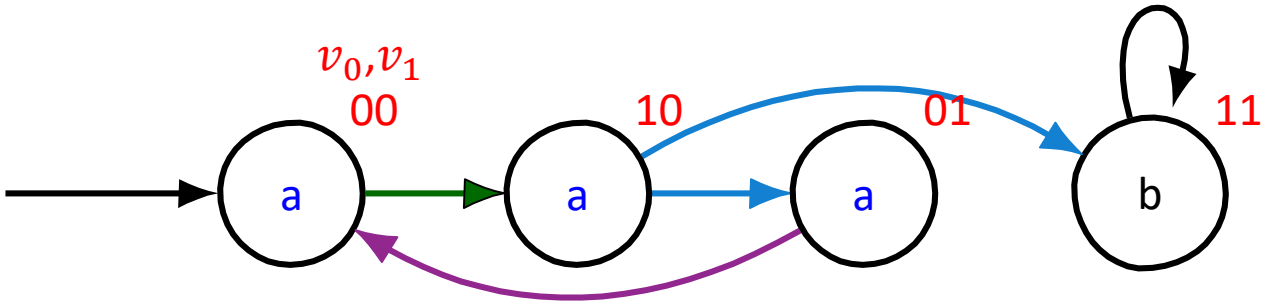
AG a

$S_0(v_0, v_1) =.$

$R(v_0, v_1, v_0', v_1')$ =

$a(v_0, v_1) =.$

Example



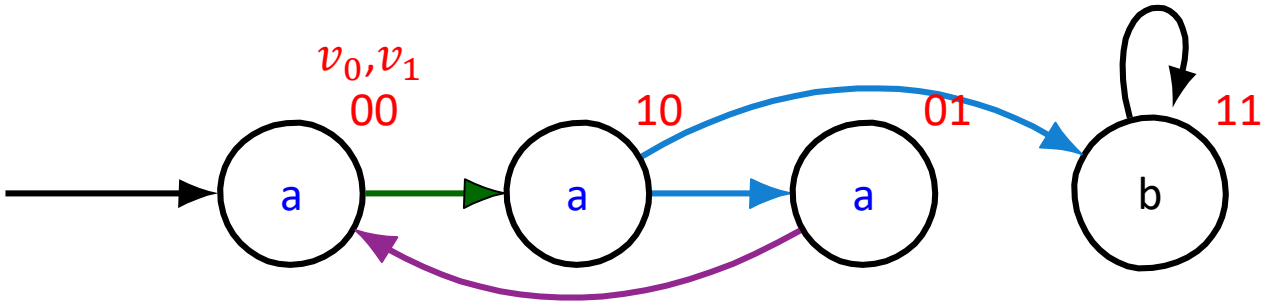
AG a

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

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

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

Example



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

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

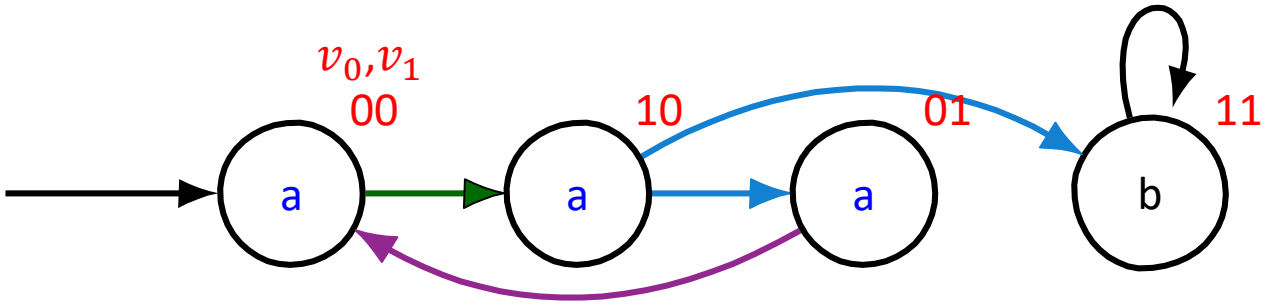
$$a(v_0, v_1) = \neg v_0 \vee \neg v_1$$

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

$$= \neg v_{0,0} \wedge \neg v_{1,0} \wedge \left(\begin{array}{c} \neg v_{00} \wedge \neg v_{10} \wedge v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge \neg v_{10} \wedge v_{11} \\ \vee \neg v_{00} \wedge v_{10} \wedge \neg v_{01} \wedge \neg v_{11} \\ \vee v_{00} \wedge v_{10} \wedge v_{01} \wedge v_{11} \end{array} \right)$$

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

Example



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

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

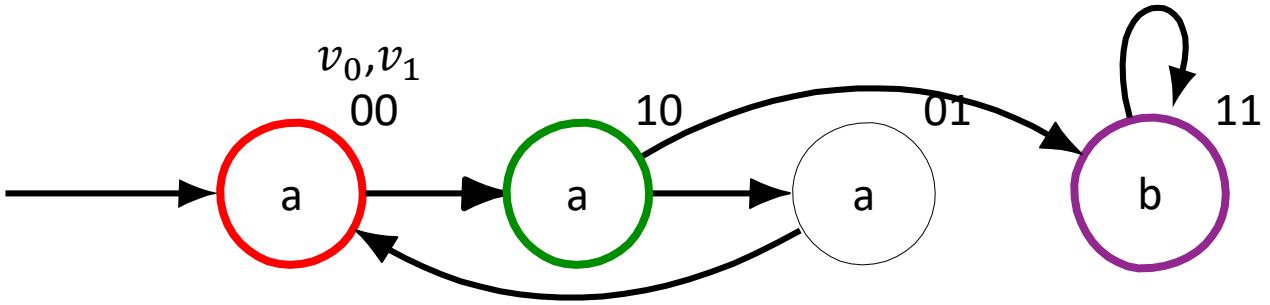
$$a(v_0, v_1) = \neg v_0 \vee \neg v_1$$

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

$$= \neg v_{0,0} \wedge \neg v_{1,0} \wedge \begin{pmatrix} \neg v_{00} \wedge \neg v_{10} \wedge v_{01} \wedge \neg v_{11} \\ \vee\ v_{00} \wedge \neg v_{10} \wedge v_{11} \\ \vee\ \neg v_{00} \wedge v_{10} \wedge \neg v_{01} \wedge \neg v_{11} \\ \vee\ v_{00} \wedge v_{10} \wedge v_{01} \wedge v_{11} \end{pmatrix} \wedge \begin{pmatrix} \neg v_{01} \wedge \neg v_{11} \wedge v_{02} \wedge \neg v_{12} \\ \vee\ v_{01} \wedge \neg v_{11} \wedge v_{12} \\ \vee\ \neg v_{01} \wedge v_{11} \wedge \neg v_{02} \wedge \neg v_{12} \\ \vee\ v_{01} \wedge v_{11} \wedge v_{02} \wedge v_{12} \end{pmatrix}$$

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

Example



AG a

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

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

Satisfying assignment

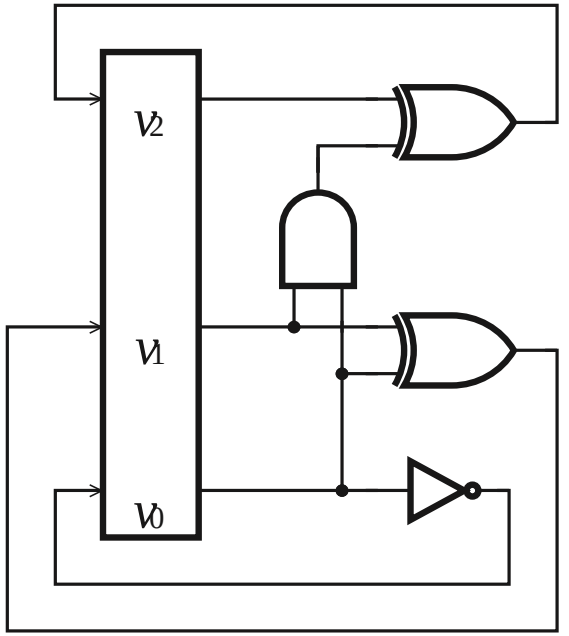
i	0	1	2
v_{0i}	0	1	1
v_{1i}	0	0	1

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 or
https://compsys-tools.ens-lyon.fr/z3/index.php

Another Example



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

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

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

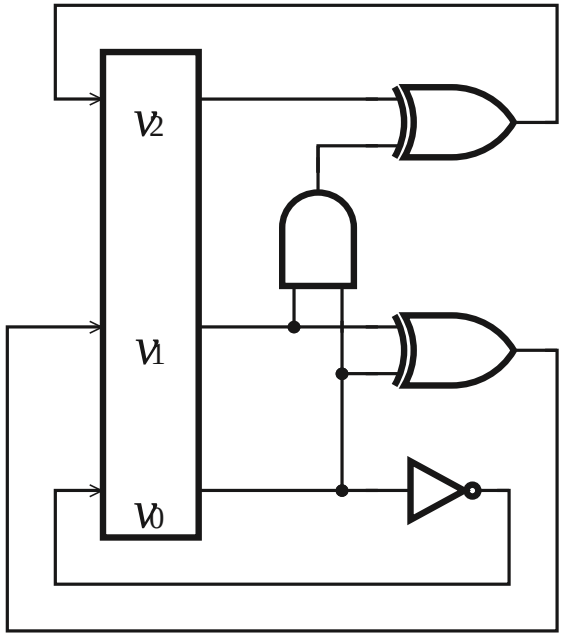
$R(V,V') = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \oplus v_1) \wedge (v_2' \leftrightarrow (v_0 \wedge v_1) \oplus v_2)$.

k =0:

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

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

Another Example



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

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

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

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

$k = 1$:

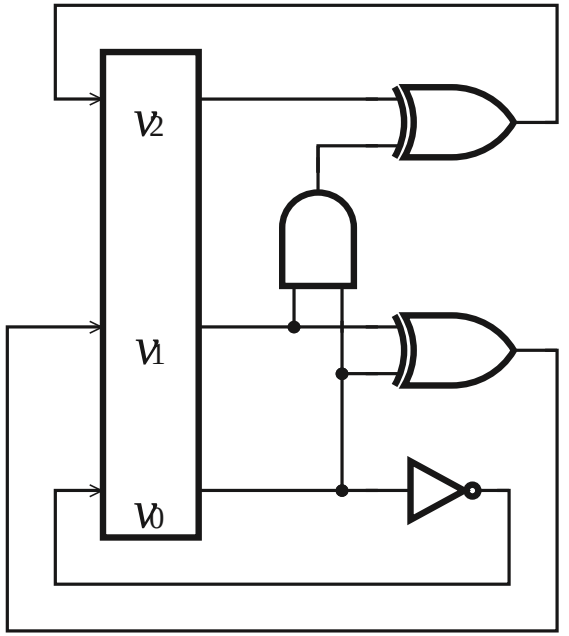
$S_0(v)$ $\neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge$

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

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

$\neg p(v)$ $\neg p(v')$

Another Example



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

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

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

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

$k = 2$:

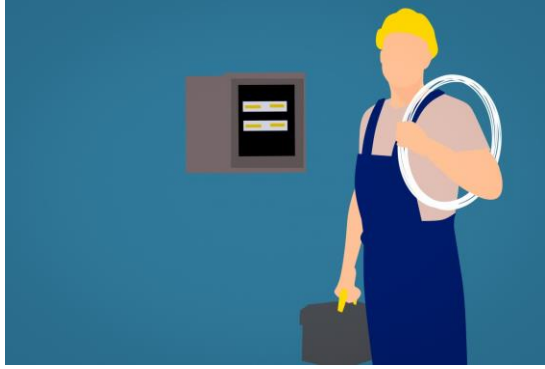
$S_0(v)$ $\neg v_0 \wedge \neg v_1 \wedge \neg v_2 \wedge$

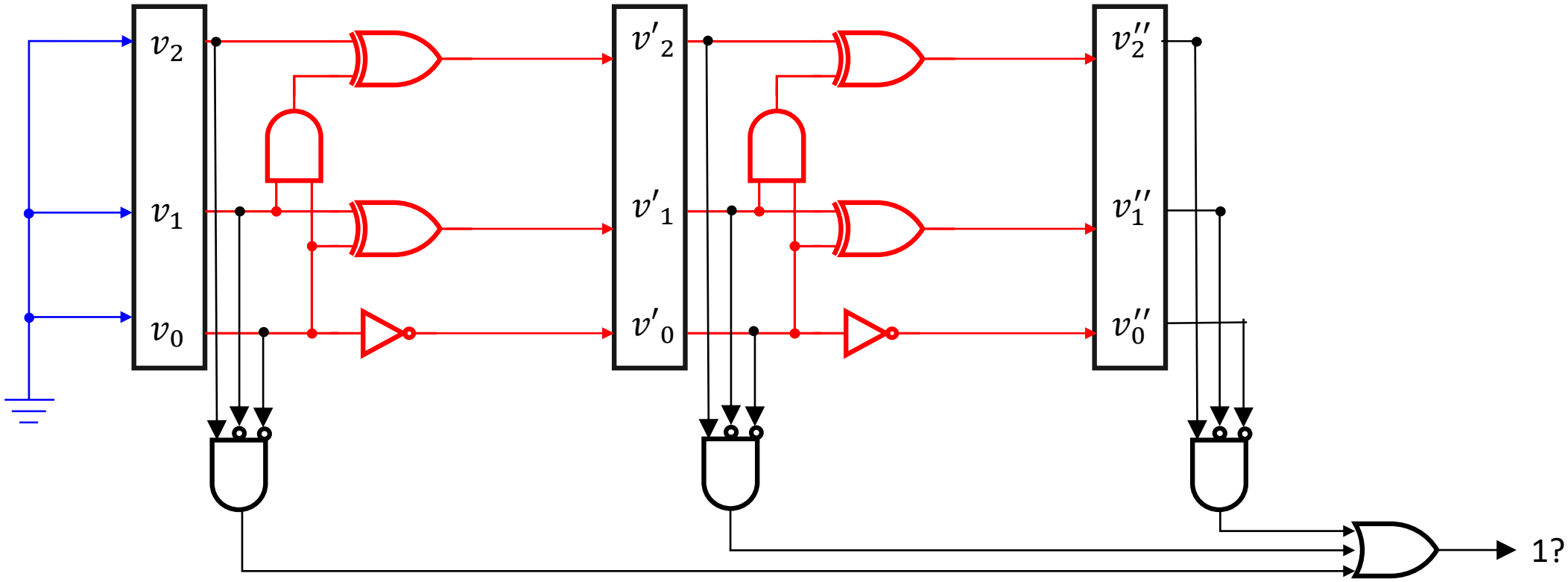
$R(v, v')$ $(v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \oplus v_1) \wedge (v_2' \leftrightarrow (v_0 \wedge v_1) \oplus v_2) \wedge$

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

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

The Electrical Engineer's Viewpoint





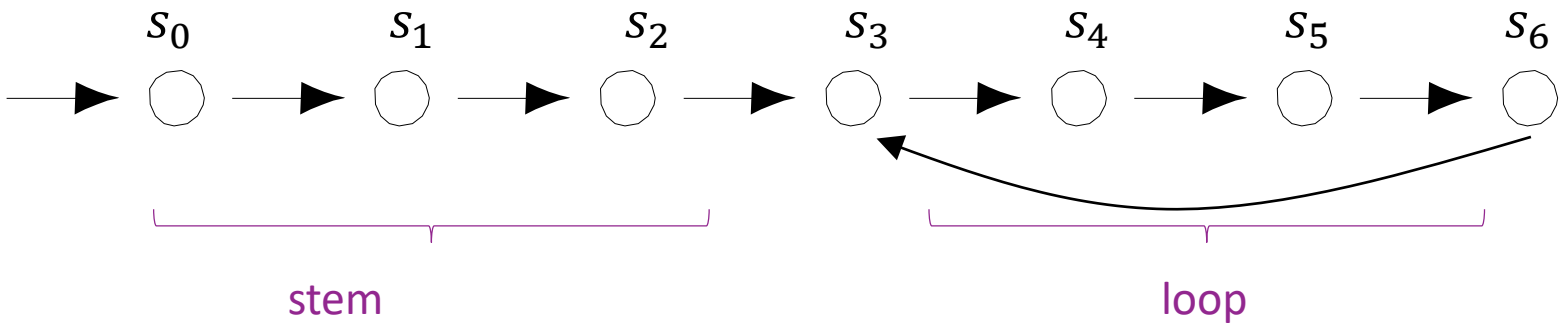
Eventuality Properties

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

$$New(V_0, \ldots, V_k) = S_0(V_0) \wedge \bigwedge_{i=0}^{k-1} (R(V_i, V_{i+1}) \wedge \bigwedge_{j<i} V_i \neq V_j)$$

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: $\forall n. Q(n)$

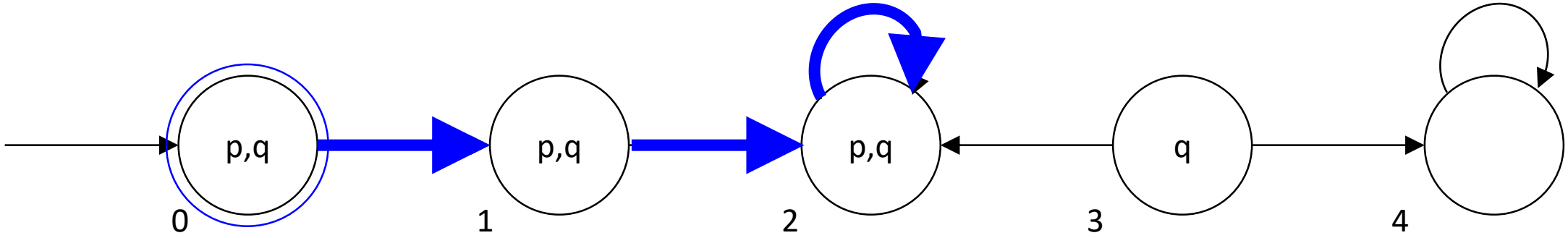
Caveat: Property may be true, but not inductive (see below)

Induction

Let's prove $\mathbf{AG}\ p$ on the following structure.

Take arbitrary path π

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

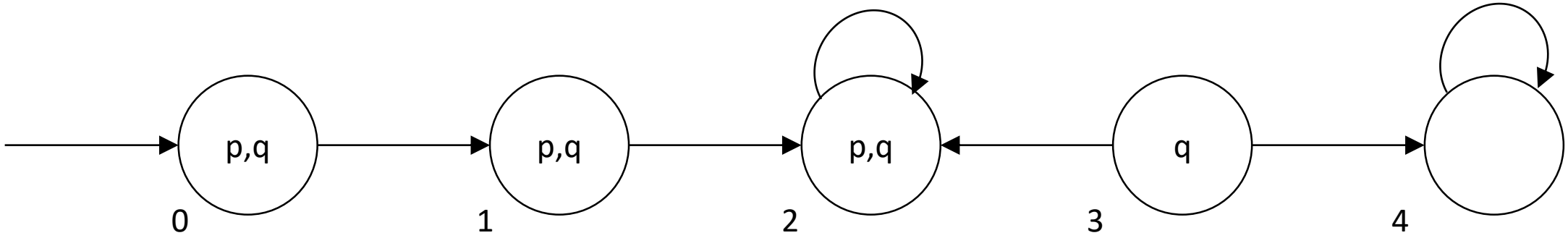


Satisfiability

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

Take arbitrary path π

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

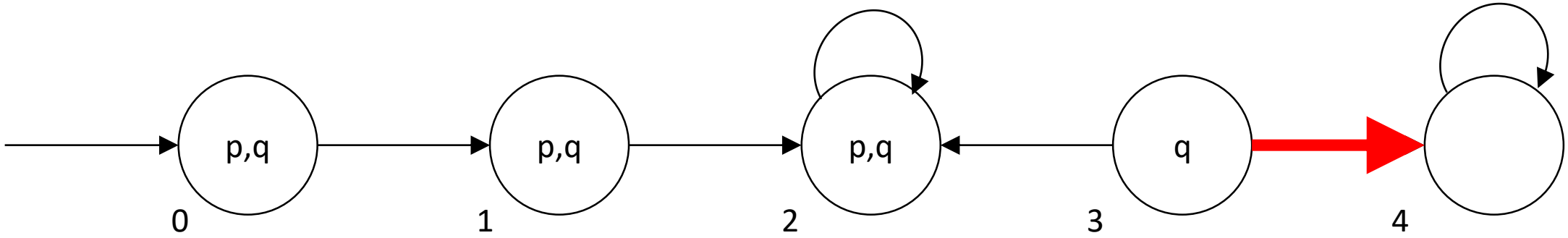


A Problem

Let's prove $\mathbf{AG}\ q$ on the following structure.

Take arbitrary path π

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



k-induction

Base:

Induction:

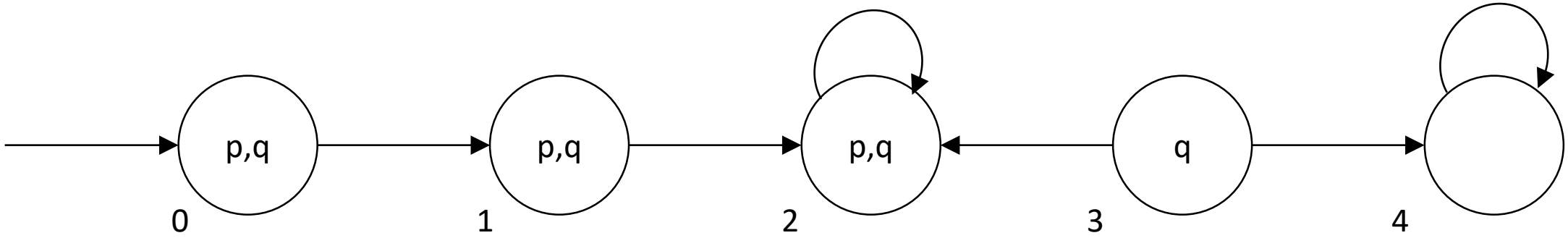
Conclusion: $\forall n.\, Q(n)$

In our setting:

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

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

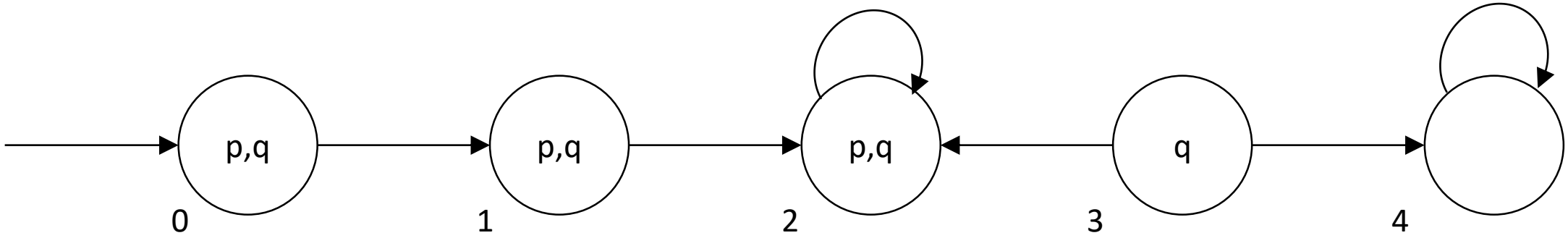
Conclusion. All paths from S_0 are labeled *q*



k-induction

Base: Prove $Q(0) \wedge \cdots \wedge \mathrm{Q}(k)$
Induction: Prove $\mathrm{Q}(\mathrm{n}-\mathrm{k}-1) \wedge \cdots \wedge Q(n-1) \Rightarrow Q(n)$
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,* where each state is labeled with q, are followed by a state with *q*
Conclusion. All paths from S_0 are labeled *q*



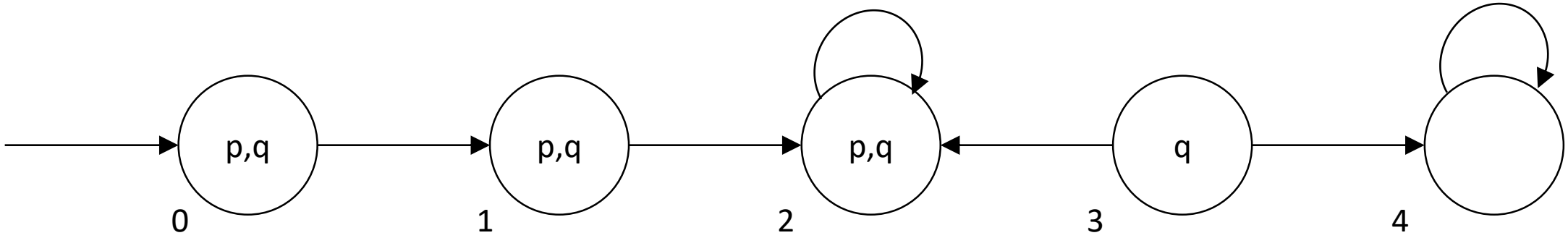
Prove AG q using 1-induction

Base: Consider paths of length 2 from initial states. $s_0 \vDash q$ and $s_1 \vDash q$.

Induction: Do all successors of arbitrary path $(s_i, s_j) \vDash q$ fulfill q?

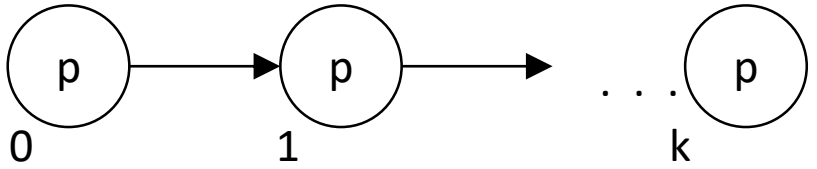
- (s_0, s_1)
- (s_1, s_2)
- (s_2, s_2)
- (s_3, s_2)

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



k-induction as Satisfiability

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

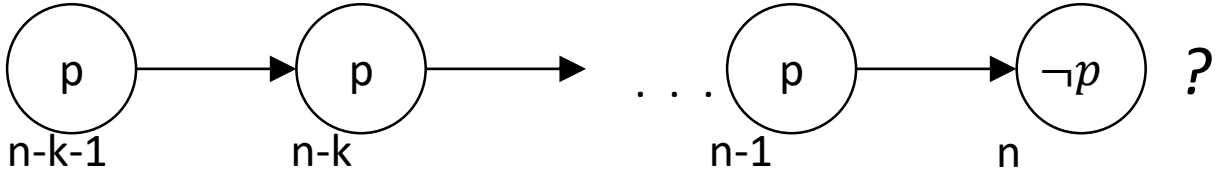


This is BMC! $S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(s_i)$

Induction. all paths of length *k* labeled with *p* are followed by a *p state*

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

Is there a path of the form



?

k-induction

while(k=0; ; k++){

build BMC formula ϕ

if ϕ SAT return “bug!”

build induction formula ψ

if ϕ UNSAT return “correct!”

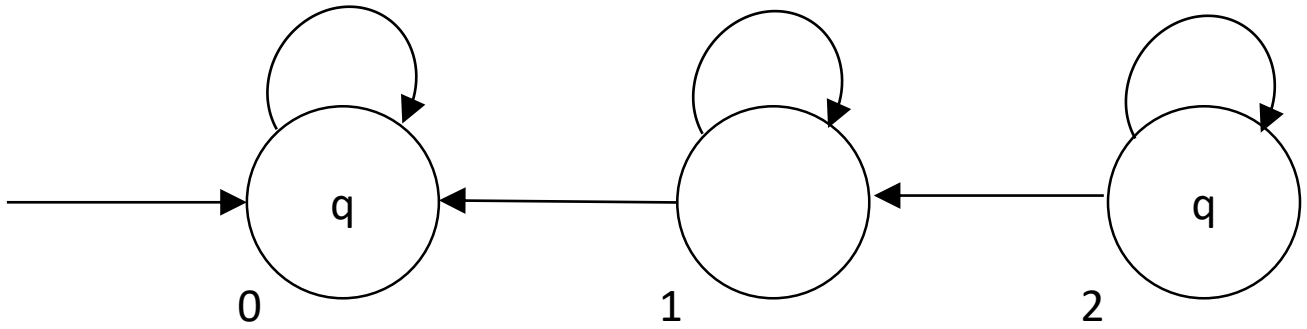
}

k-induction is not Complete

System satisfies $\mathbf{AG}\ q$, 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 *q* are followed by a *q state.* *FALSE*



Making k-induction Complete

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

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