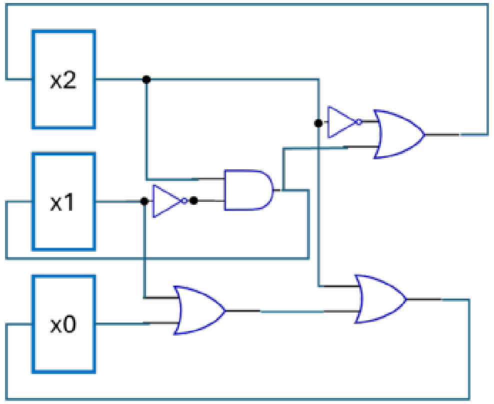
Homework

Deadline: **18 March 2025, 9:00 am**
Submit your solution through TeachCenter

Consider the synchronous circuit C from last week's exercise. (The initial value of the state variable x_0 is `true`. The initial values of x_1 and x_2 are unknown.)



Task 1. [**100 points**] We want to use BMC to check whether x_0 is always `true`.

3.1 Will BMC find a counterexample? If so, what is the smallest k such that BMC finds a counterexample. [**20 points**]

3.2 Write the BMC formula for $k = 2$. (You can use S_0 and R in your formula. [**40 points**]

3.3 Is the formula satisfiable? Explain. [**40 points**]

3.1 No, x_0 is always true

3.2 Let $V = \{x_0, x_1, x_2\}$ and let $\phi(V) = x_0$. The BMC formula is $\psi(V, V', V'') = S_0(V) \wedge R(V, V') \wedge R(V, V'') \wedge (\neg\phi(V) \vee \neg\phi(V') \vee \neg\phi(V''))$.

Note that I write $\phi(V')$ to mean $\phi(V)$, where every occurrence of x_i has been replaced by x'_i.

3.3 The formula is not satisfiable because there is no path of length 2 from an initial state to a state in which x_0 is false.

Verifying Reachability Properties with k-induction



Mary Sheeran, Koen Claessen, Per Bjesse, 2000

Make BMC Complete

Increase k until the following in unsatisfiable:

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

Drawback: k can be very large.

Make BMC Complete

Increase k until the following in unsatisfiable:

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

Drawback: k can be very large.

How do you prove $i < n + 1$ for the following program?

```
BigInt i;
i = 0;
while(true)
      if(i == n) i = 0;
      else i++;
```

Motivation

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

Base: Prove $Q(0)$

Induction: Prove $Q(\mathrm{t}-1) \Rightarrow Q(\mathrm{t})$

Conclusion: $\forall \mathrm{t}.\, Q(\mathrm{t})$

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

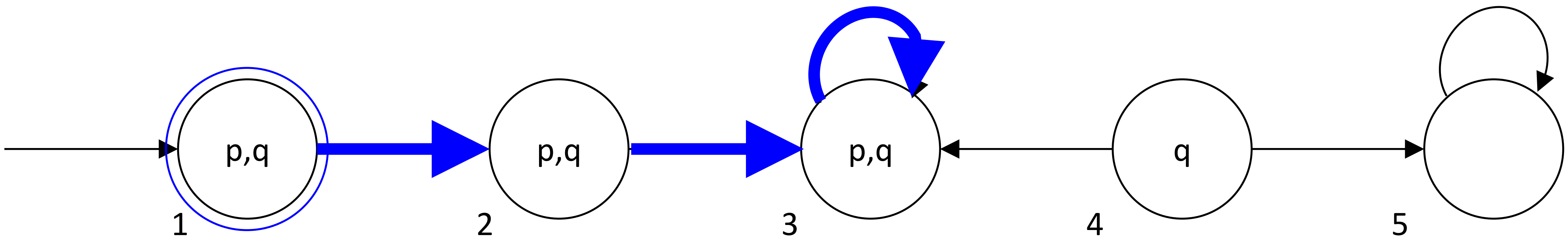
We will go through a series of algorithms until we find a nice one

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 $\forall n.\pi(n) \vDash p$

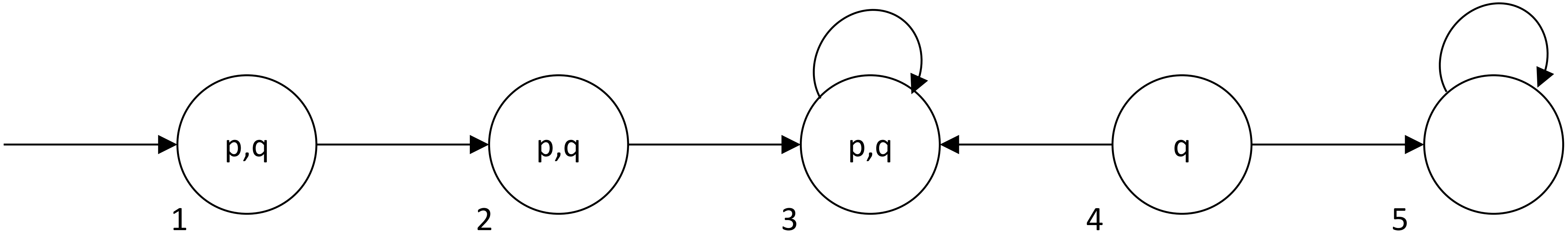


Satisfiability

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 $\forall n. \pi(n) \vDash p$

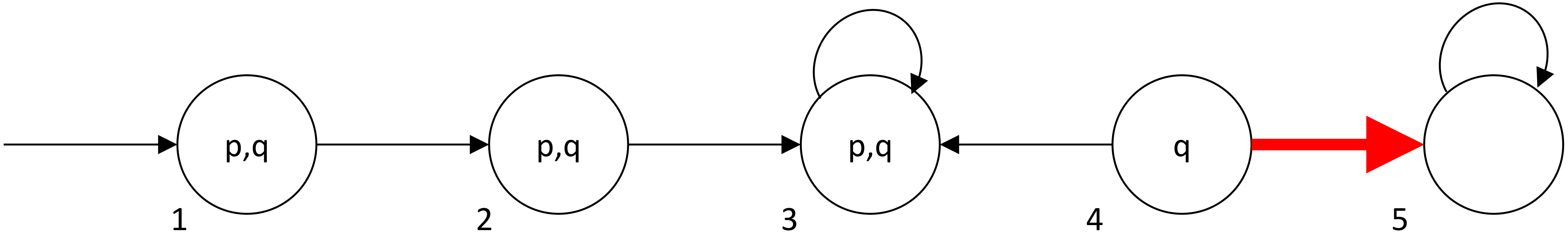


A Problem

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

Take arbitrary path π

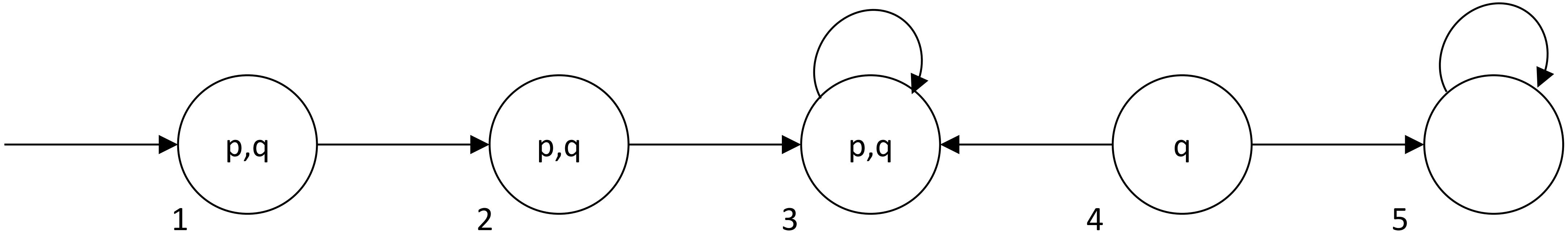
- **Base case:** $\pi(0) \vDash \mathrm{q}$
- **Induction:** if $\pi(n-1) \vDash q$ then $\pi(n) \vDash q$ **not true!**
- ~~**Conclusion**: for any path π we have $\forall n.\, \pi(n) \vDash q$~~ **not all true properties are inductive**



k-induction

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

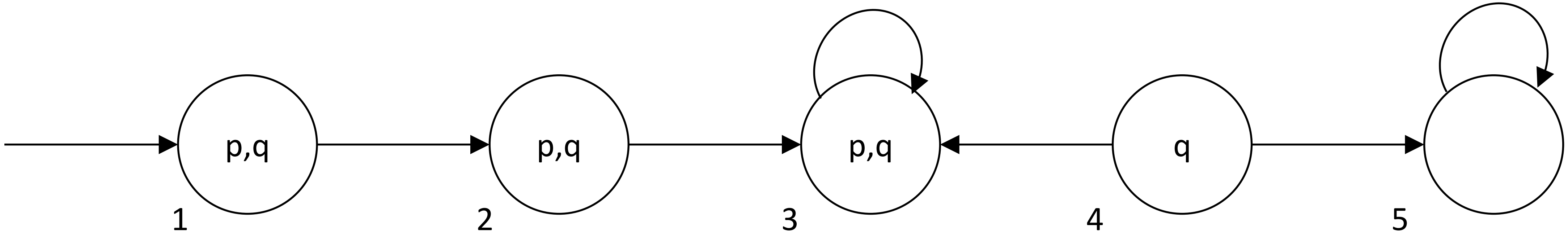
In our setting:
Base. all paths from S_0 with k or fewer edges are labeled *q*
Induction. all paths of length *k* labeled with all *q*s are followed by a *q*
Conclusion. All paths from S_0 are labeled *q*



***k*-induction**

Base: Prove $Q(1) \wedge \cdots \wedge \mathrm{Q}(k-1)$
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* labeled with all *q*s are followed by a *q*
Conclusion. All paths from S_0 are labeled *q*

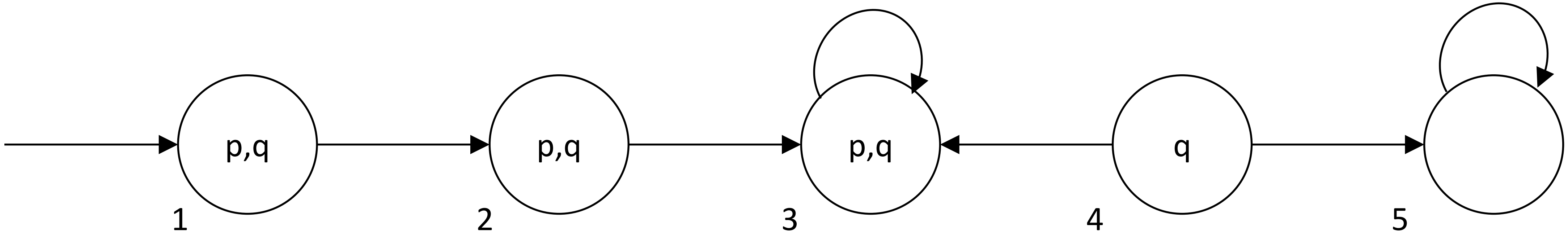


Prove AG q using 1-induction

Base: Consider all paths of length 1 from q_1: $q_1 \vDash q$ and $q_2 \vDash q$.
Induction: Do all successors of paths of length 1 labeled (q, q) fulfill q?

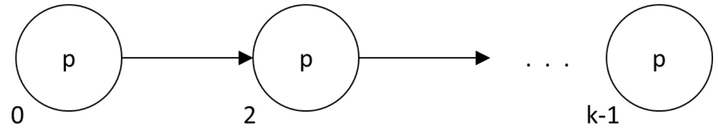
- (q_1, q_2)
- (q_2, q_3)
- (q_3, q_3)
- (q_4, q_3)

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



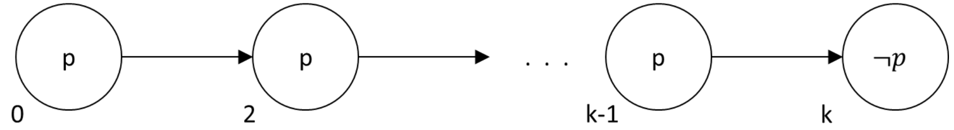
k-induction as Satisfiability

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

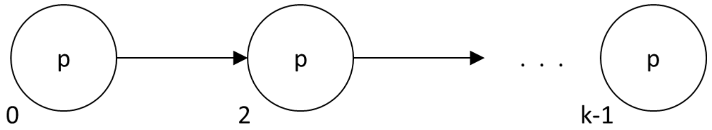


Induction. every path of length *k* labeled with all *p*s is followed by *p*

Formula satisfiable iff there is a counterexample



k-induction as Satisfiability



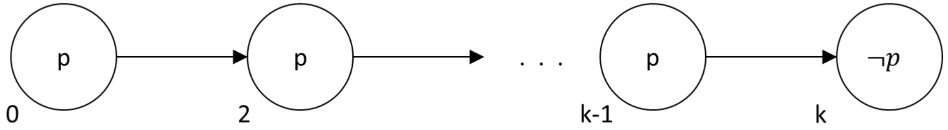
Base. all paths of length *k* from S_0 are labeled *p*

This is BMC! $S_0(s_1) \wedge \bigwedge_{i=1}^{k} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k+1} \neg p(s_i)$

Induction. every path of length *k* labeled with all *p*s is followed by *p*

$$\bigwedge_{i=1}^{k+1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{k+1} p(s_i) \wedge \neg p(s_{k+2})$$

Formula satisfiable iff there is a counterexample



k-induction

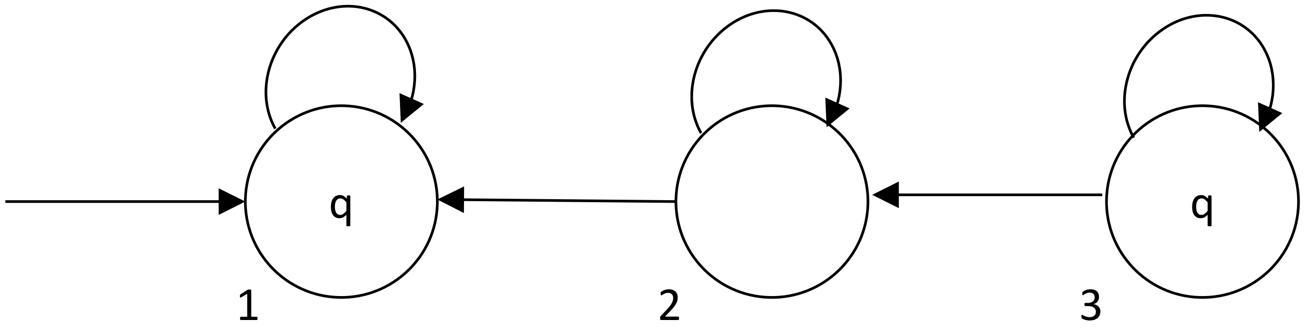
```
while(k=0; ; k++){
        build BMC formula ϕ_k
        if ϕ SAT return "bug!"

        build induction formula ψ_k
         if ϕ UNSAT return "correct!"
}
```

This Version of 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 all *q*s are followed by a *q*. *FALSE*

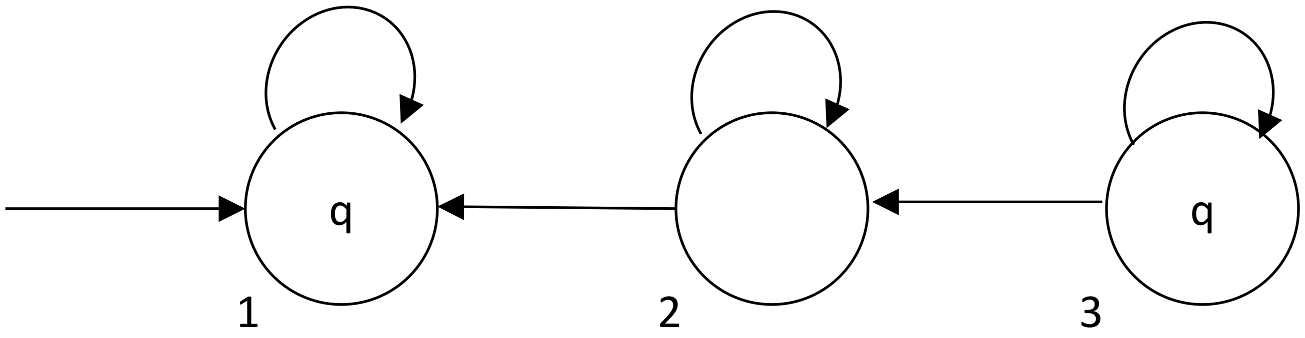


k-induction, the Final Version

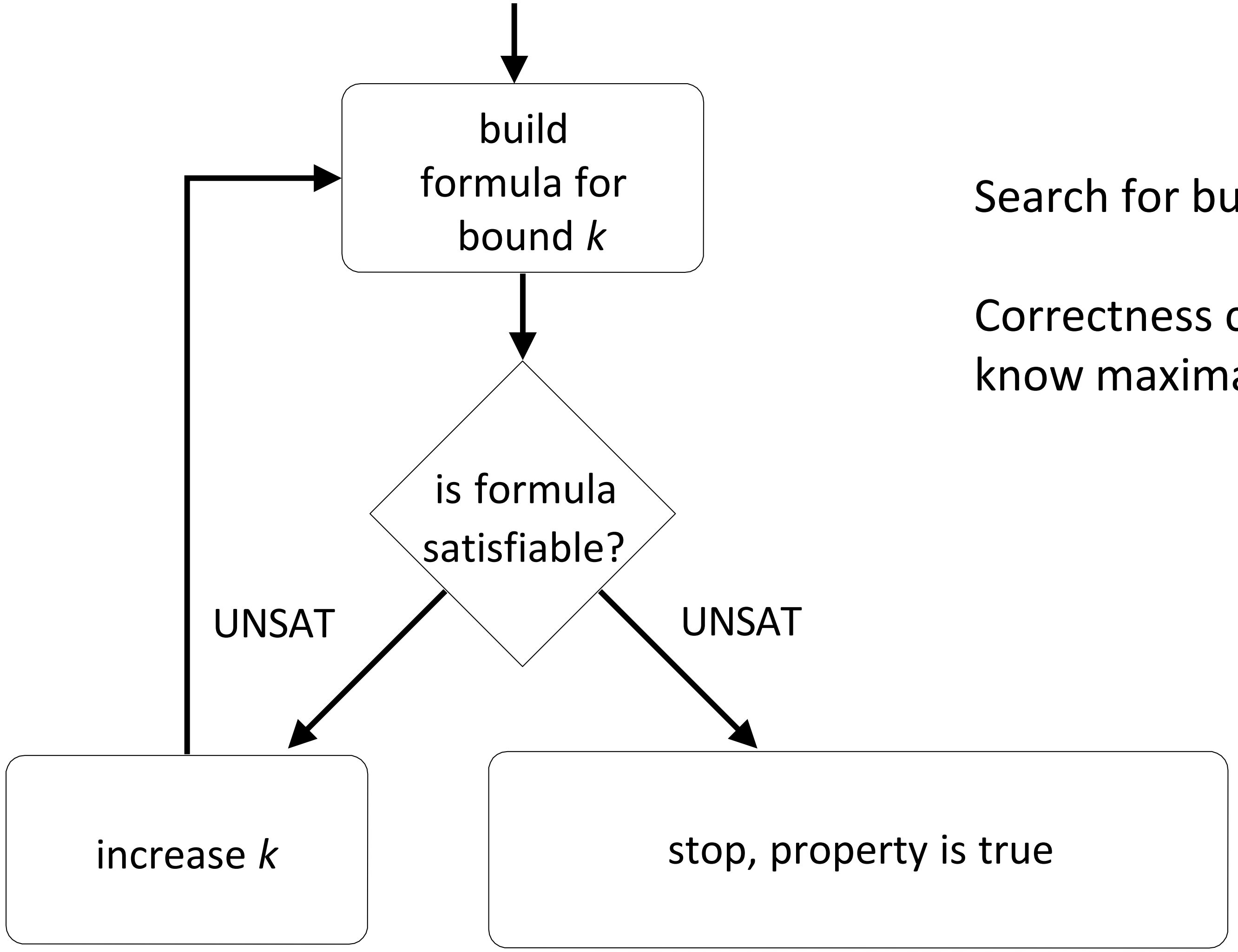
Induction. all noncyclic paths of length k labeled with all ps are followed by a p

$$\bigwedge_{i=1}^{k+1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{k+1} p(s_i) \wedge \neg p(s_{k+2}) \wedge$$

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



k-induction



Search for bugs within k steps.

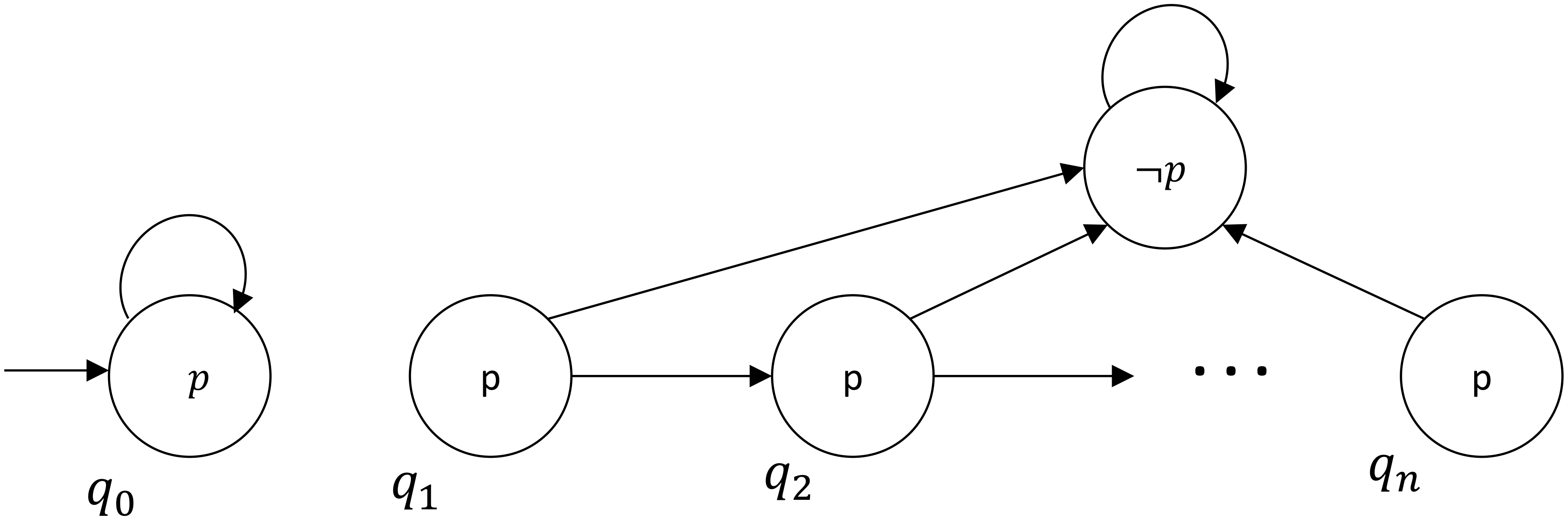
Correctness can only be proven if we know maximal value for *k*

Problems with *k*-induction

Problem: Sometimes *k* is very large

In the following machine, you need $k = n + 1$ to prove $\mathbf{AG}\, p$.

Idea: Automatically find better inductive invariants.



Note to myself

- There is a BASE formula and an INDUCTION formula. You need both to do k-induction. That was unclear in the home work.h