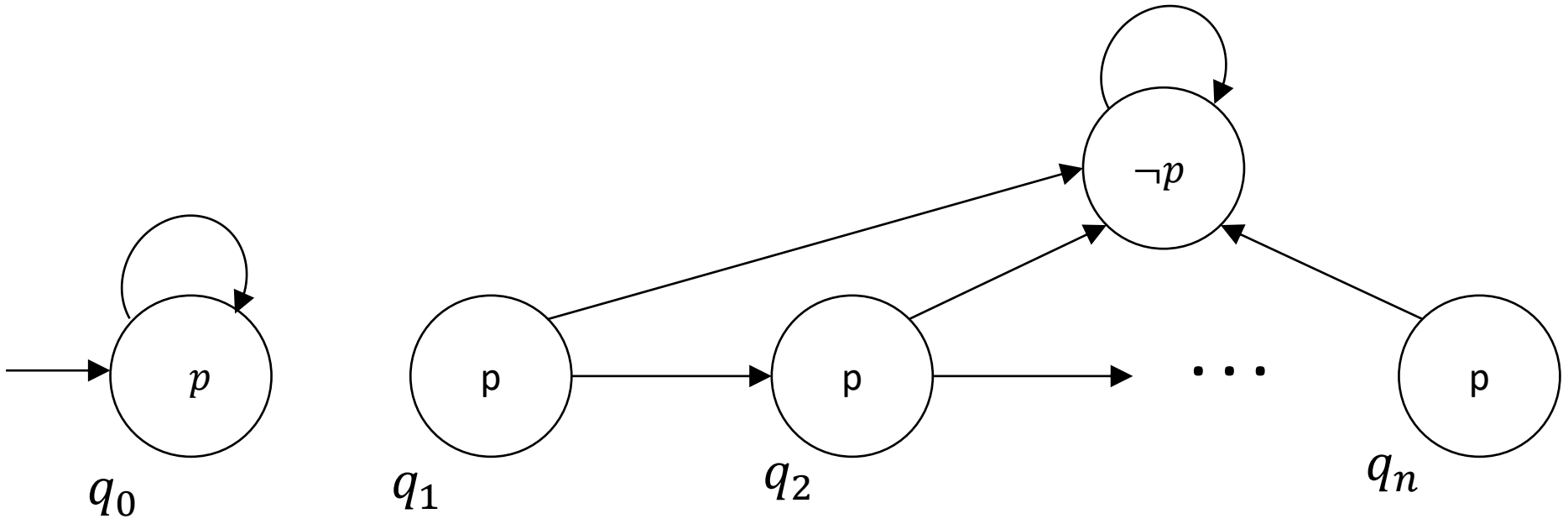
Model Checking with Inductive Invariants

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. Avoid many copies of R**!**



Inductive Invariant

Remember $img(Q) = \{s' \mid \exists s.\, R(s, s')\}$

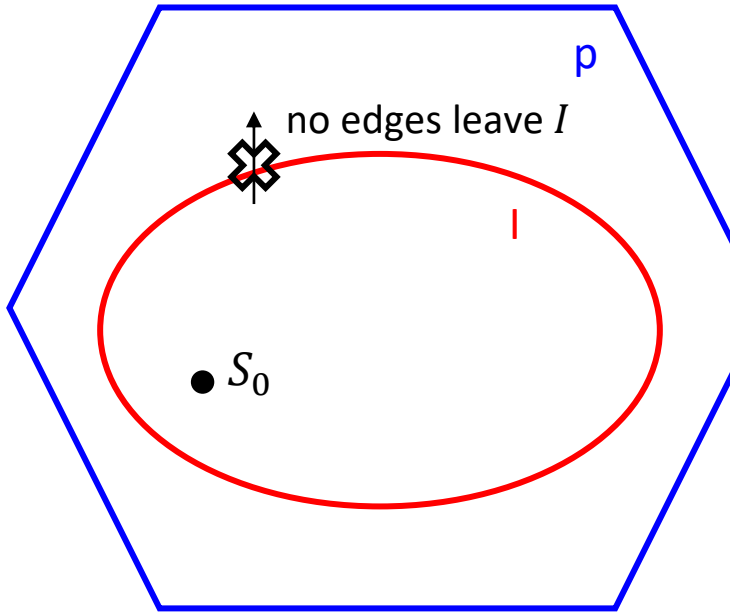
Definition. $I \subseteq S$ is an inductive invariant for AG p if

1. $S_0 \subseteq I$
2. $img(I) \subseteq I$
3. $\forall s \in I.\, s \vDash p$

In formulas:

1. $S_0 \rightarrow I$
2. $I \wedge R \rightarrow I'$
3. $\mathrm{I} \rightarrow p$

There is an inductive invariant for AG p iff AG p holds



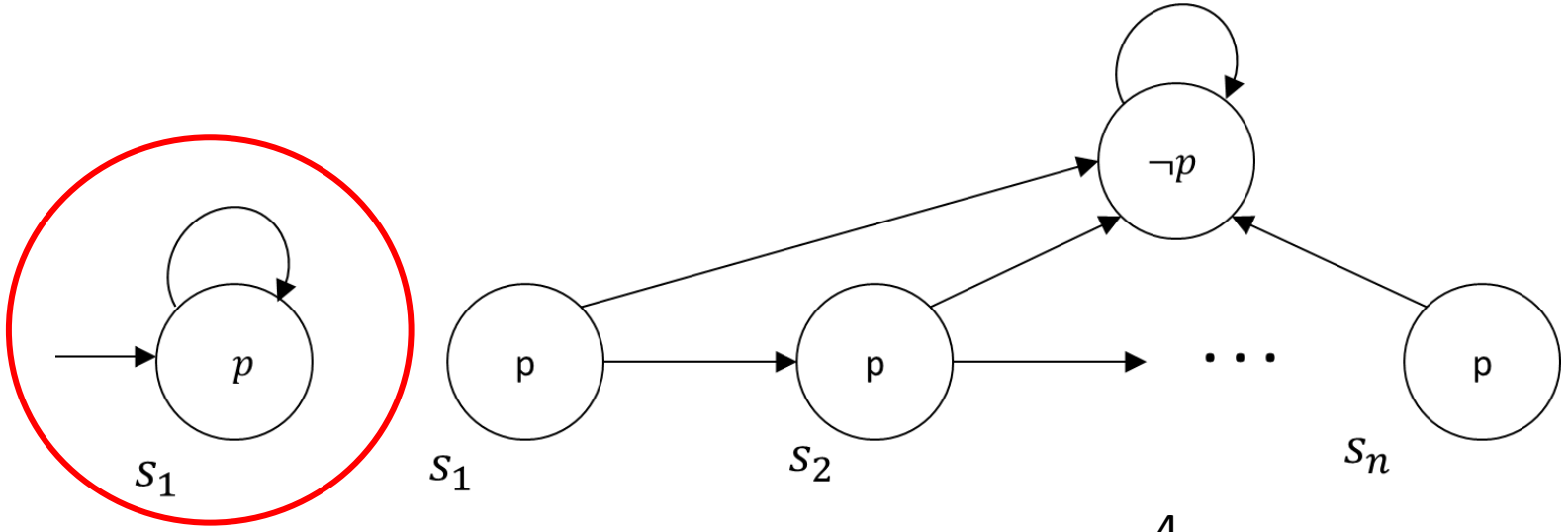
Inductive Invariant

Remember $img(Q) = \{ s' \mid \exists s. R(s, s') \}$

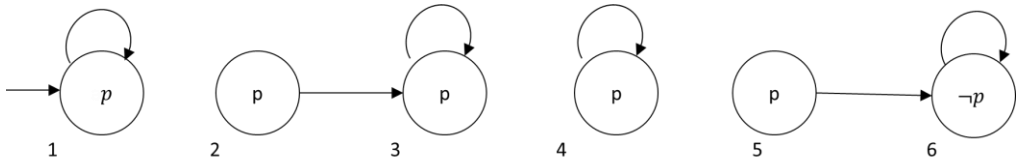
Definition. $I \subseteq S$ is an inductive invariant for AG p if

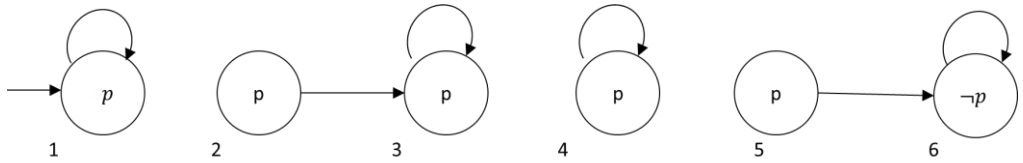
1. $S_0 \subseteq I$
2. $img(I) \subseteq I$
3. $\forall s \in I. s \vDash p$

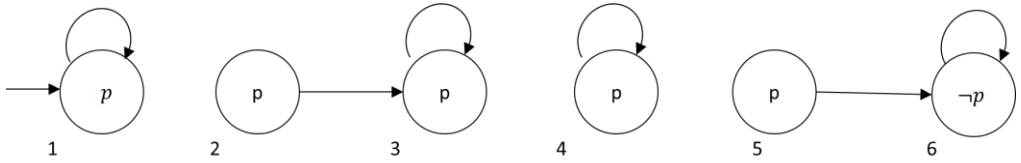
There is an inductive invariant for AG p iff AG p holds.

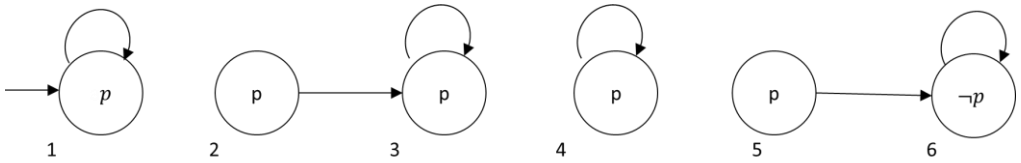


Multiple Invariants

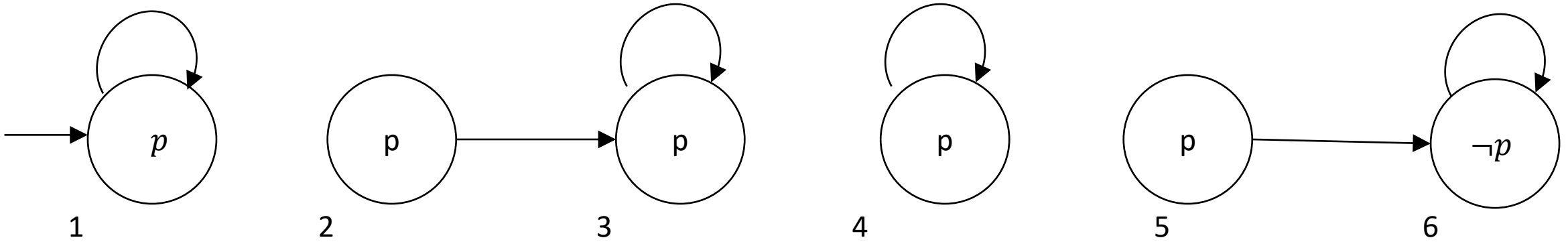








Strongest & Weakest Invariant



1. Smallest (strongest) invariant: all reachable states

2. Largest (weakest) invariant: all states that cannot reach $\neg p$

Two model checking algorithms:

1. Start with initial states, add successors until nothing changes or you reach $\neg p$
2. Start with $\neg p$, add predecessors until nothing changes or you reach an initial state

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.”

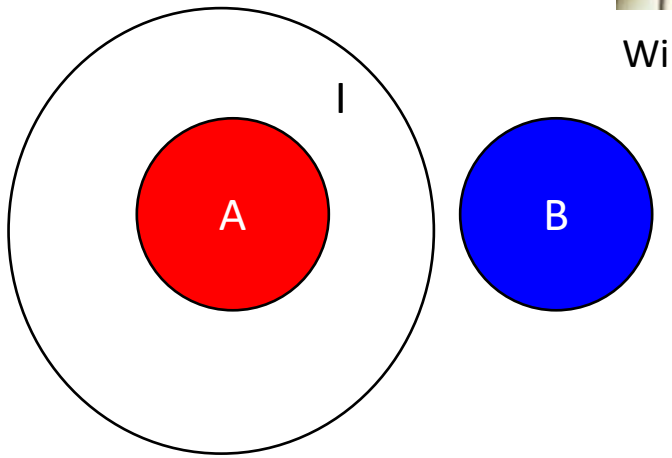
Kenneth McMillan

Interpolant

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

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

William Craig, 1957



Example. Let

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

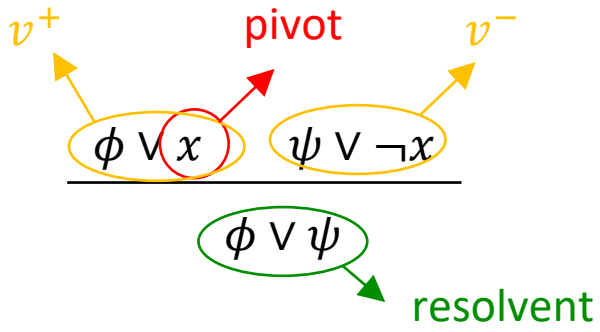
$B = (\neg a_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \neg a_4.$

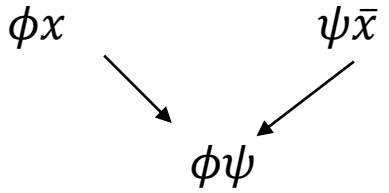
$A \wedge B$ is not satisfiable.

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

1. $\big((a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge a_2\big) \rightarrow (\neg a_3 \wedge a_2)$
2. $(\neg a_3 \wedge a_2) \wedge \big((\neg a_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \neg a_4\big) \equiv \bot$
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

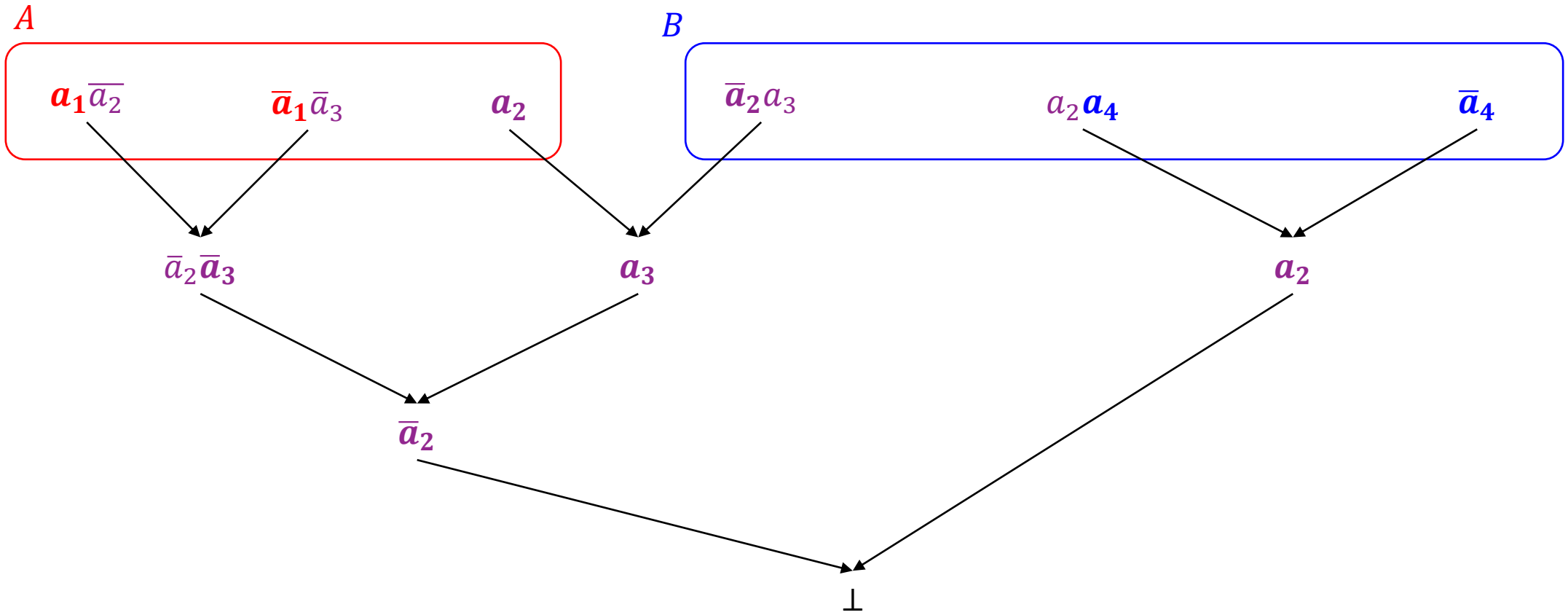
Algorithm. Go through resolution proof top-down.

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.

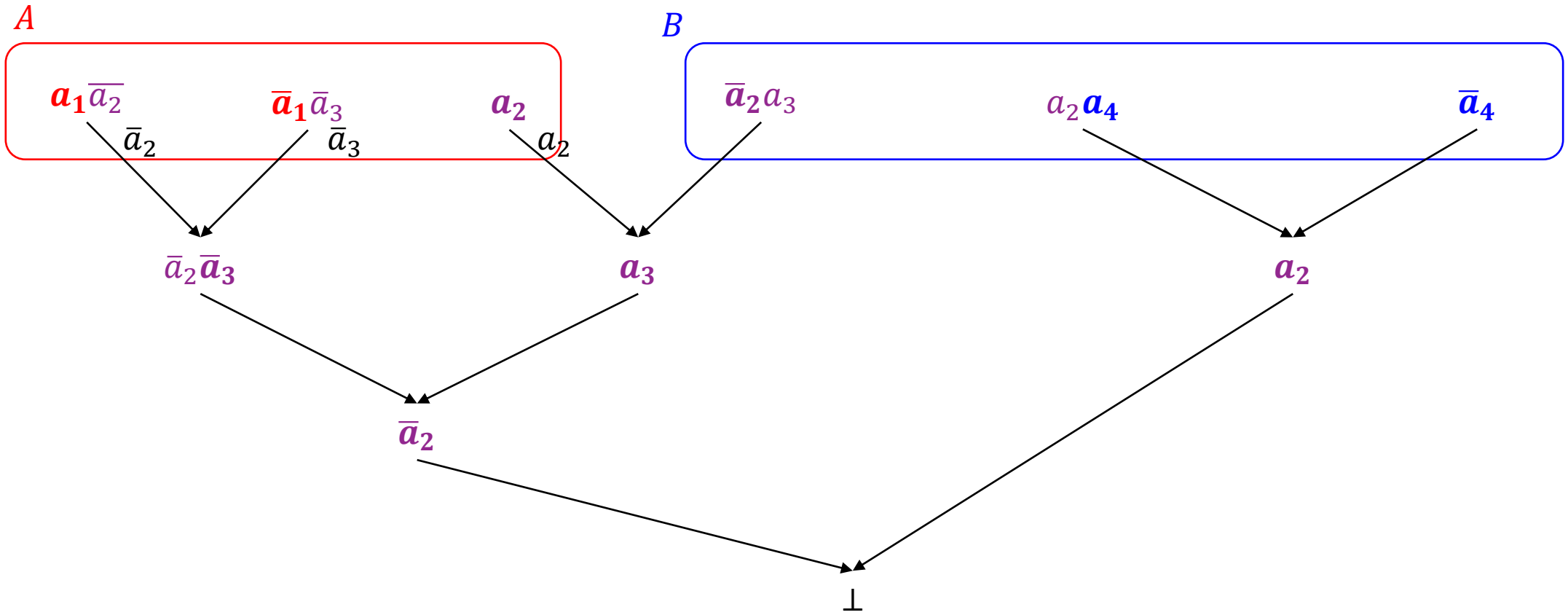
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

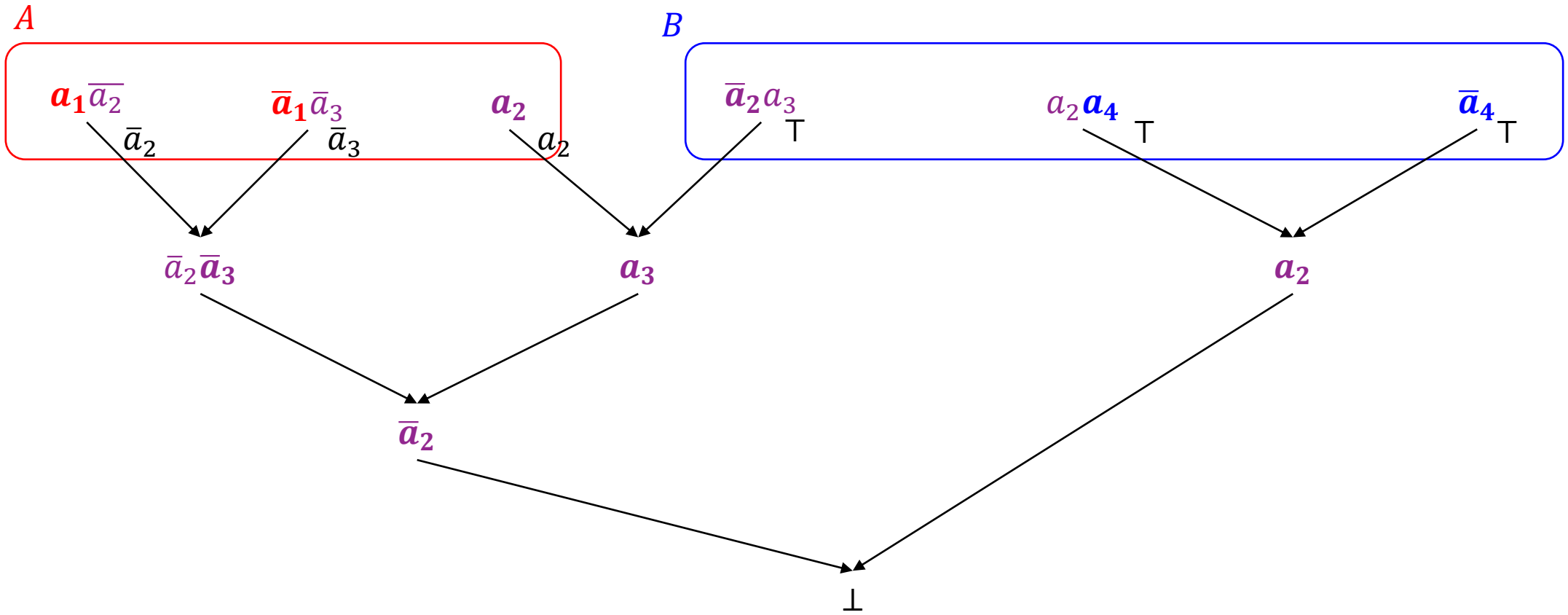
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

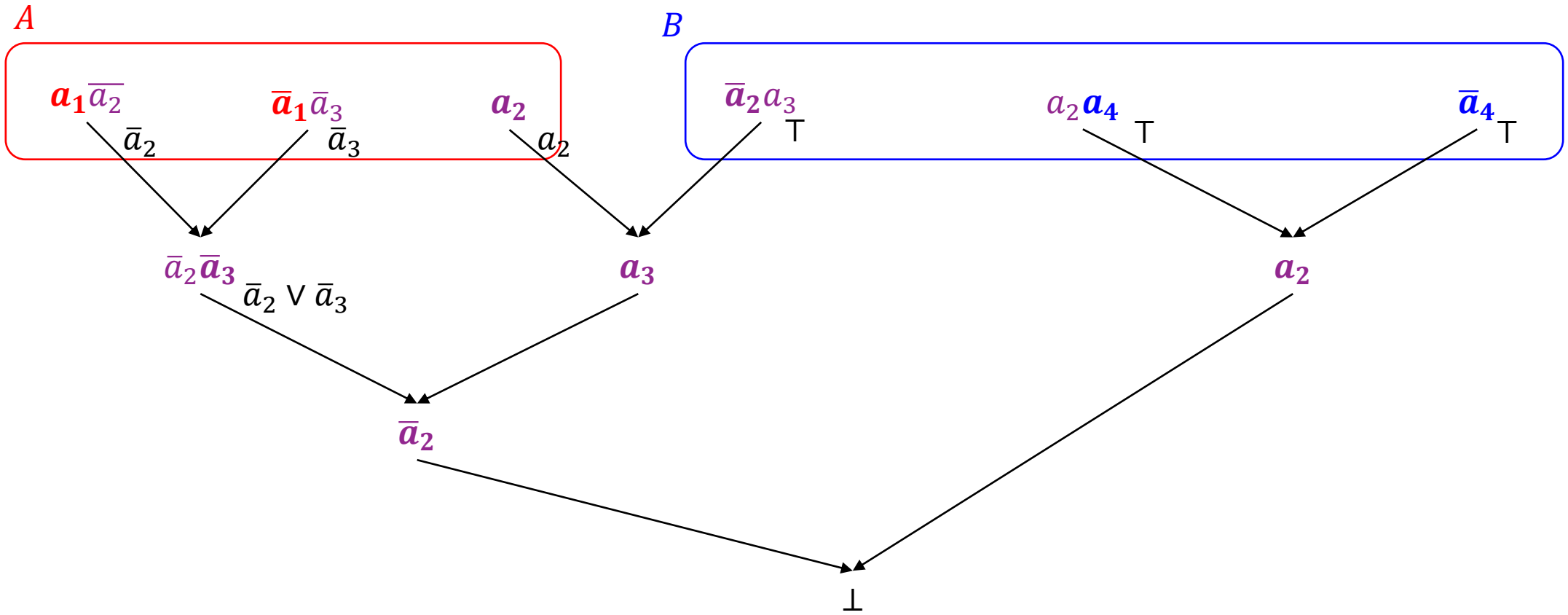
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

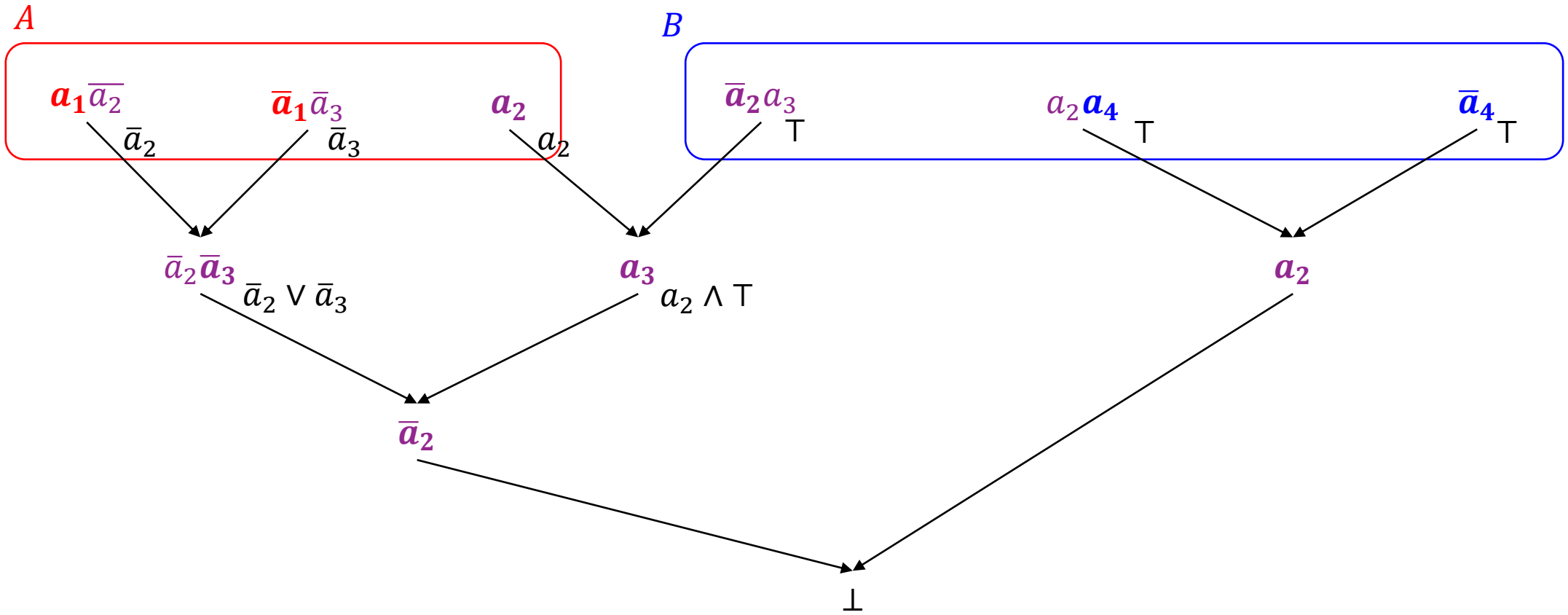
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

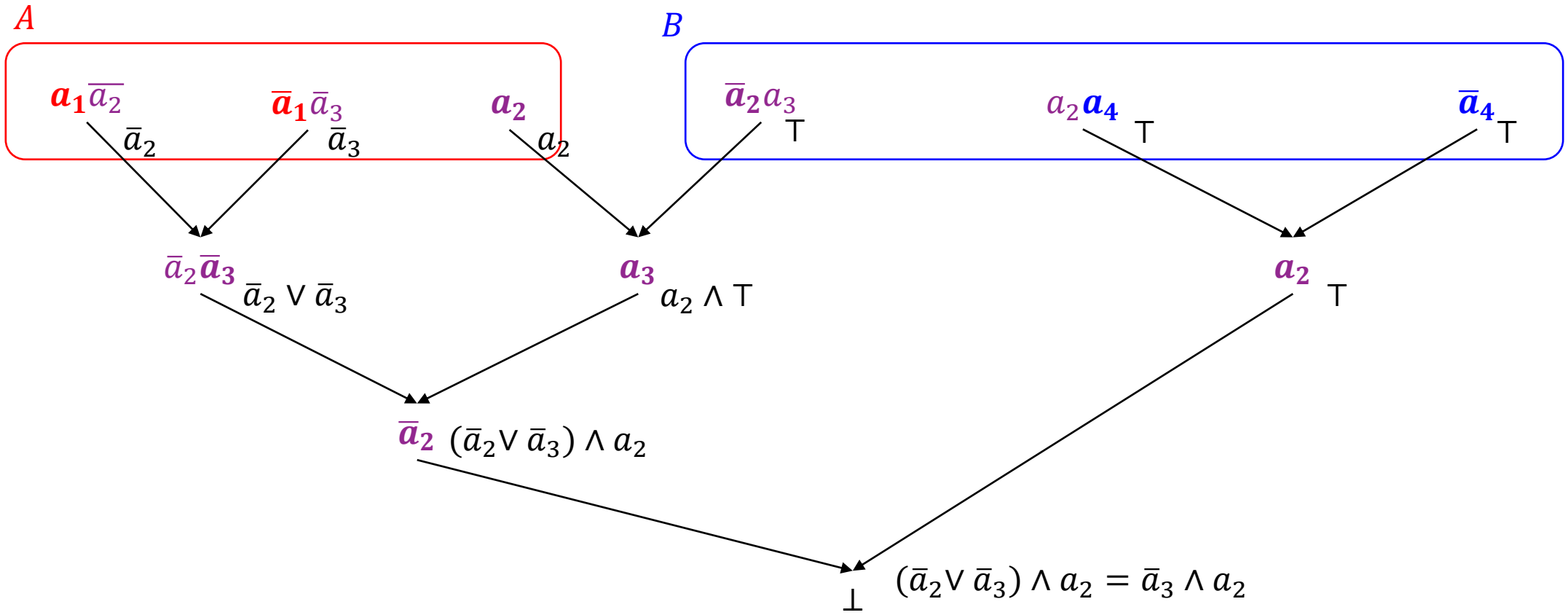
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



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 $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Model Checking with Interpolations

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
  - Until you find an inductive invariant
- Interpolants are (good) overapproximation of post-image computation

Overapproximation

BMC to prove p:

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

Suppose we have another transition relation R' with more edges: $R \rightarrow R'$

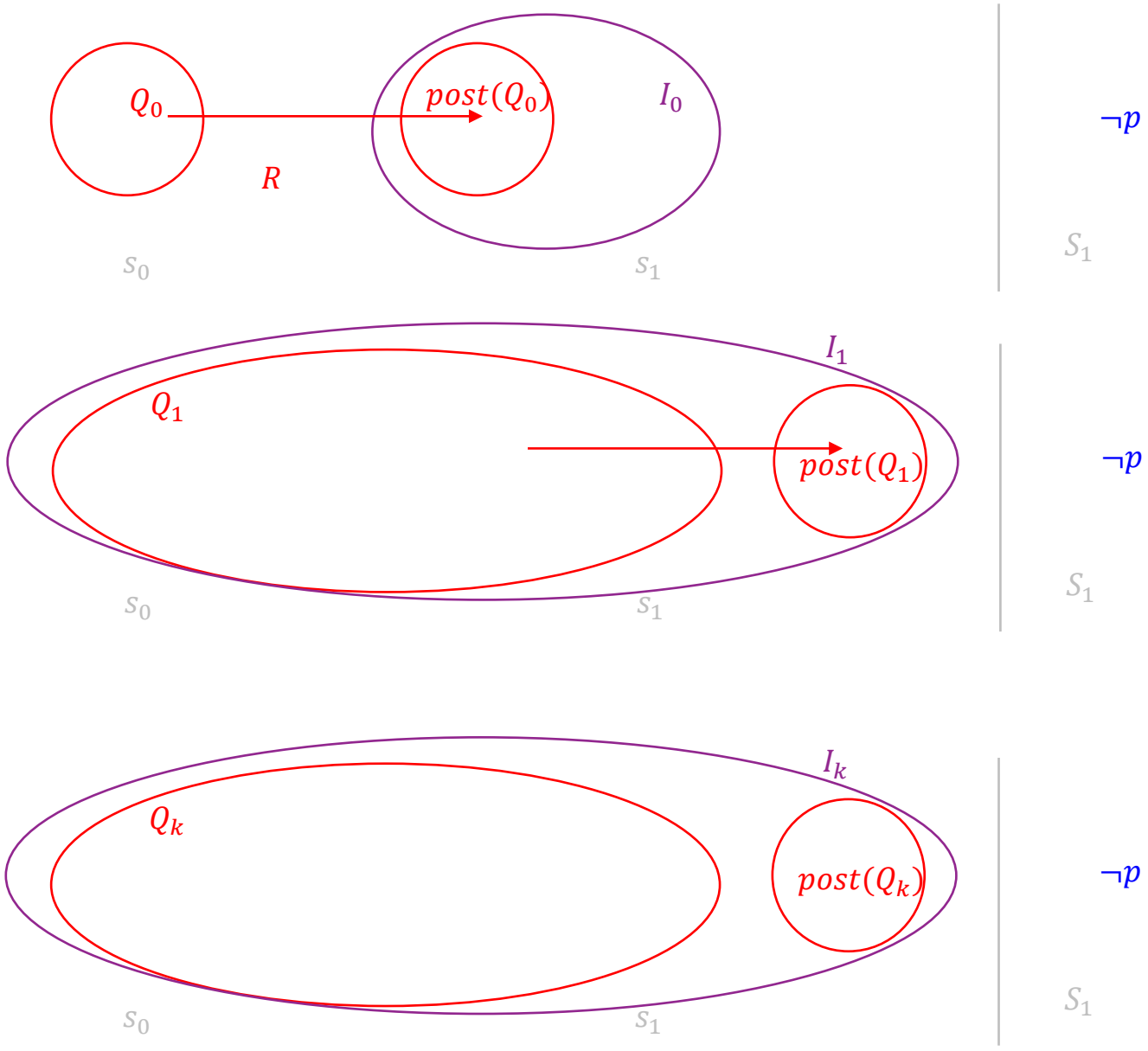
What does this do?

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

Idea 1:

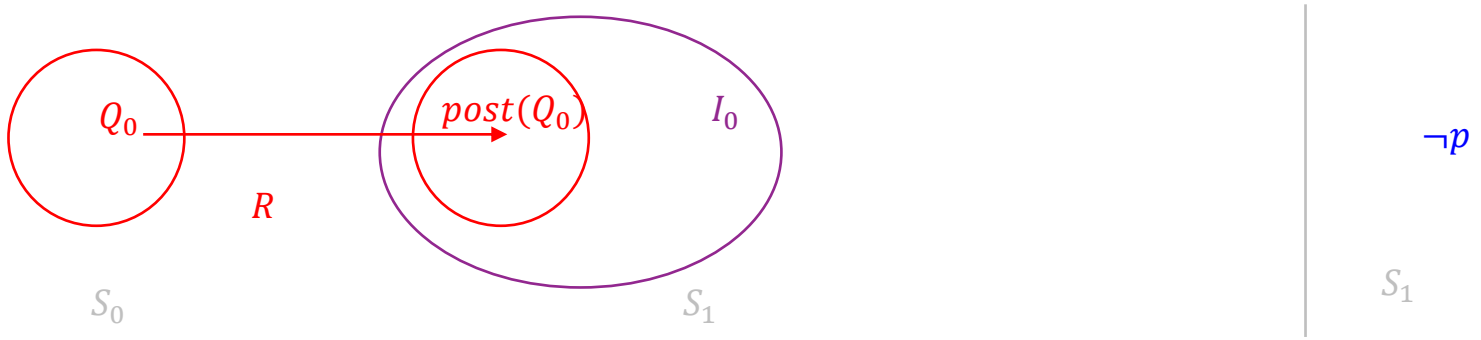
01. if $S_0 \wedge \neg p$ is SAT return "$M \not\models \mathrm{AG}\, p$ "
02. $Q := S_0(s_0)$
03. **while** true **do**
04. $A := Q(s_0) \wedge R(s_0, s_1)$
05. $B := \neg p(s_1)$
06. **if** $A \wedge B$ is SAT **then**
…
10. **else**
11. compute interpolant $I(S_1)$ for A and B;
12. if $I(s_0) == Q(s_0)$ then return "$M \models \mathrm{AG}\, p$";
13. $Q := Q \vee I(s_0)$;
14. **end if**
15. **end while**

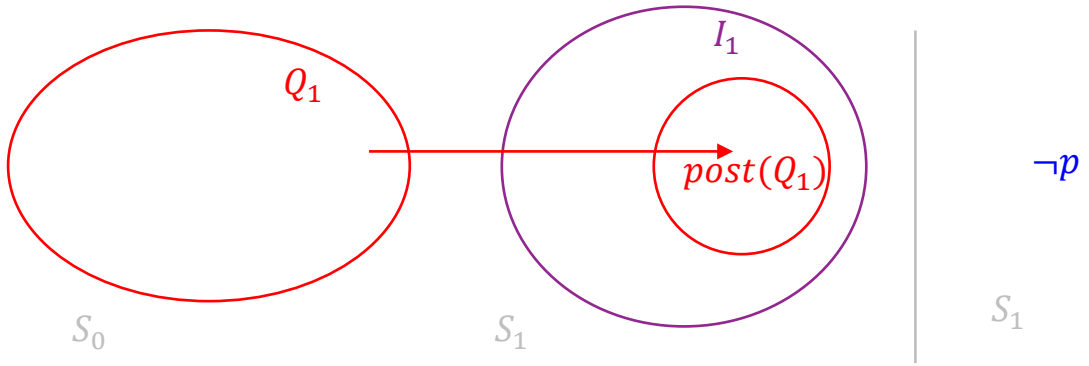
$$Q_k \supseteq postimg^k(S_0)$$



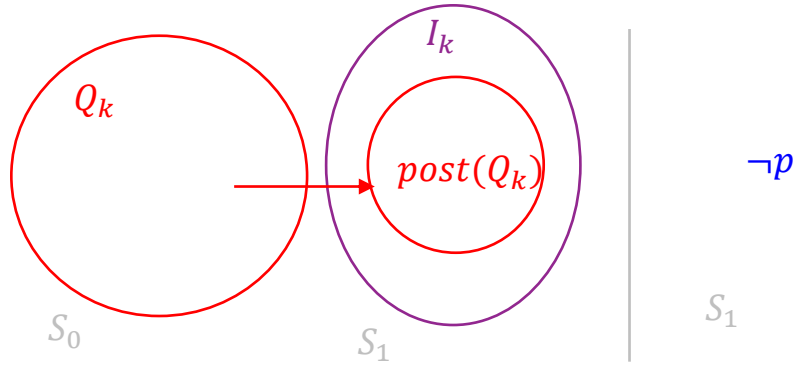
Idea 1:

01. if $S_0 \wedge \neg p$ is SAT return "$M \not\models \mathrm{AG}\, p$ "
02. $Q := S_0(s_0)$
03. **while** true **do**
04. $A := Q(s_0) \wedge R(s_0, s_1)$
05. $B := \neg p(s_1)$
06. **if** $A \wedge B$ is SAT **then**
…
10. **else**
11. compute interpolant $I(S_1)$ for A and B;
12. if $I(s_0) == Q(s_0)$ then return "$M \models \mathrm{AG}\, p$";
13. $Q := Q \vee I(s_0)$;
14. **end if**
15. **end while**





$$Q_k \supseteq postimg^k(S_0)$$

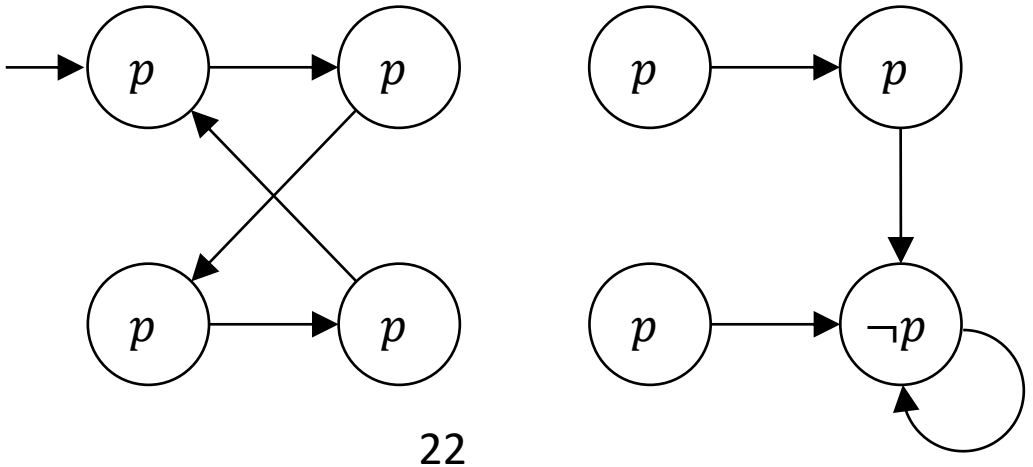


Idea 1:

01. if $S_0 \wedge \neg p$ is SAT return "$M \not\models \mathrm{AG}\, p$ "
02. $Q := S_0(s_0)$
03. **while** true **do**
04. $A := Q(s_0) \wedge R(s_0, s_1)$
05. $B := \neg p(s_1)$
06. **if** $A \wedge B$ is SAT **then**
    …
10. **else**
11. compute interpolant $I(s_1)$ for A and B;
12. if $I(s_0) == Q(s_0)$ then return "$M \models \mathrm{AG}\, p$";
13. $Q := Q \vee I(s_0)$;
14. **end if**
15. **end while**

What could go wrong?

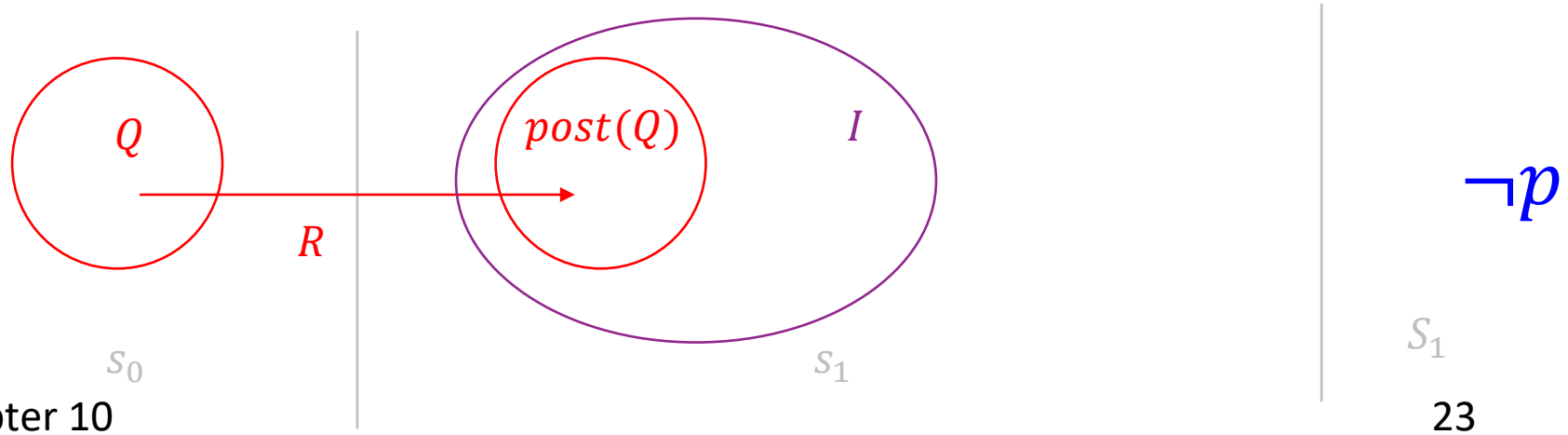
And what do we do against it?



Completeness

Recall BMC check for $\neg \mathbf{AG} p$:
$S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(s_i)$.

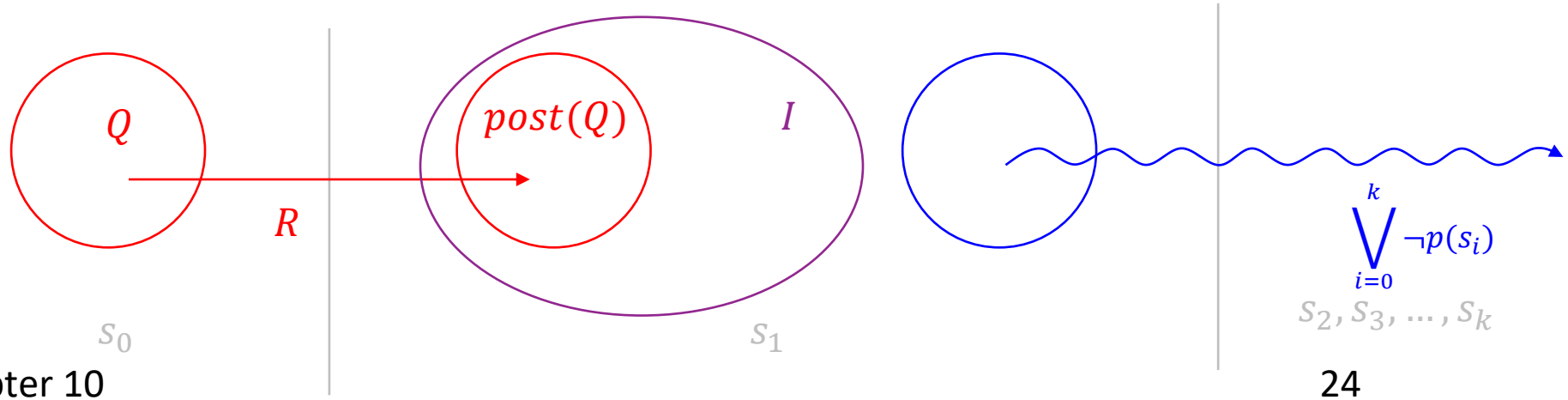
Instead, start from Q such that $Q \vDash p$
$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \neg p\ (s_1)$.
Suppose ϕ unsatisfiable, $I(s_1)$ is an interpolant



Reachability Checking with Interpolation

Recall BMC check for $\neg \mathbf{AG} p$:
$S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(s_i)$.

Instead, start from Q such that $Q \vDash p$
$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k} \neg p(s_i)$.
Suppose ϕ unsatisfiable, $I(s_1)$ is an interpolant



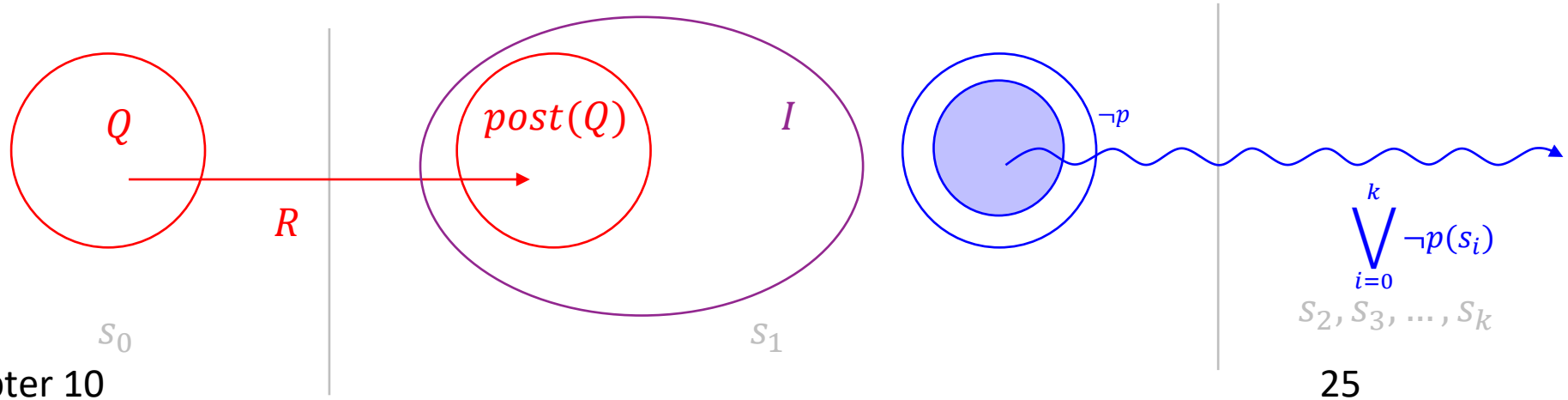
Reachability Checking with Interpolation

Recall BMC check for $\neg \mathbf{AG} p$:
$S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} \neg p(s_i)$.

Instead, start from Q such that $Q \vDash p$
$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k} \neg p(s_i)$.
Suppose ϕ unsatisfiable, $I(s_1)$ is an interpolant

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

Note 2: $I \supseteq post(Q)$



Algorithm

01. if $S_0 \wedge \neg p$ is SAT return "$M \not\models \mathrm{AG}\, p$ "
02. $Q := S_0(s_0)$
03. **while** true **do**
04. $A := Q(s_0) \wedge R(s_0, s_1)$
05. $B := \bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k} \neg p(s_i)$;
06. **if** $A \wedge B$ is SAT **then**
07. **if** $Q = S_0$ then return "$M \not\models \mathrm{AG}\, p$"; // $\neg p$ can be reached from S_0
08. Increase k // Not sure if path to $\neg p$ is real. Increase precision
09. $Q := S_0(s_0)$;
10. **else**
11. compute interpolant $I(s_1)$ for A and B;
12. if $I(s_0) == Q(s_0)$ then return "$M \models \mathrm{AG}\, p$";
13. $Q := Q \vee I(s_0)$;
14. **end if**
15. **end while**

10.4.4 Correctness

If CraigReachability returns "$\boldsymbol{M \models AG\ p}$" then $\boldsymbol{M \models AG\ p}$

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 fixed point $Q^* = Q_i$ so $Q^* \leftarrow postimage^*(Q_0)$. Now because $Q_i \wedge \neg p = \bot$, we have $postimage^*(Q_0) \wedge \neg p = \bot$.

If CraigReachability returns "$\boldsymbol{M \not\models AG\ p}$" then $\boldsymbol{M \not\models AG\ p}$

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

CraigReachability terminates

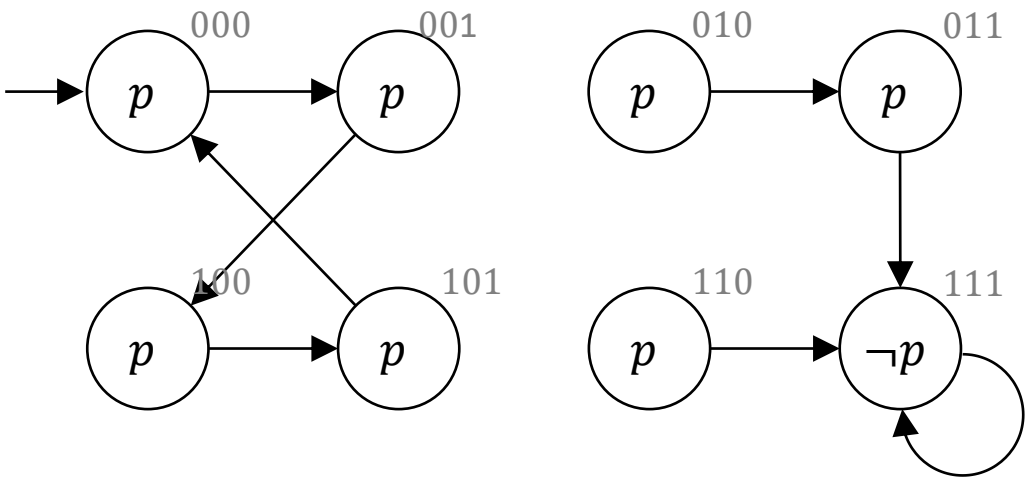
Note that k increases.

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.

Example $\mathbf{AG}\ p$

$x_1 x_2 x_3$



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

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

  if $Q = S_0$ then return "$M \not\models \mathrm{AG}\, p$";

  increase k

$Q := S_0(s_0)$;

else

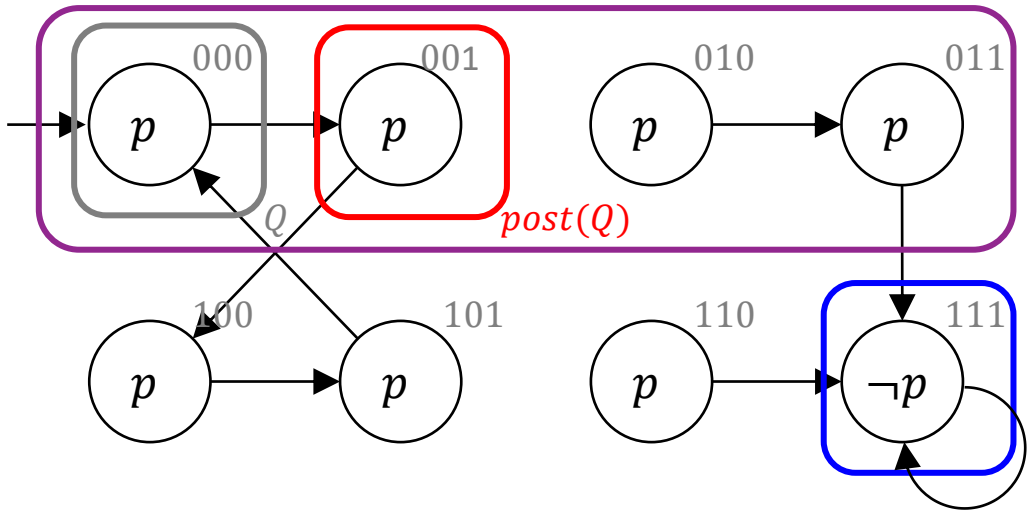
  I = interpolate I A and B;

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

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

Example $\mathbf{AG}\ p$

$x_1x_2x_3$



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

$\boldsymbol{k = 1.}$

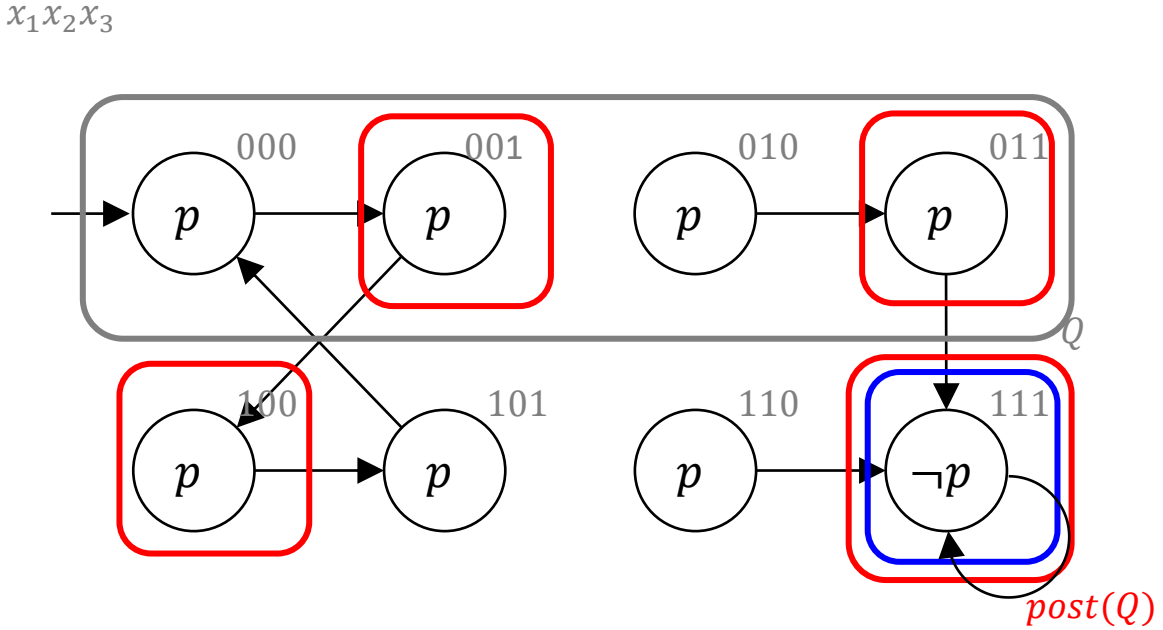
$Q = \neg x_1 \wedge \neg x_2 \wedge \neg x_3 = \{000\}.$

ϕ is UNSAT

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

if $A \wedge B$ is SAT **then**
  if $Q = S_0$ then return "$M \not\models \mathrm{AG}\ p$";
  increase k
 $Q := S_0(s_0)$;
else
  compute interpolant I for A and B;
  if $I(s_0) \rightarrow Q$ then return "$M \models \mathrm{AG}\ p$";
 $Q := Q \vee I(s_0)$;

Example $\mathbf{AG}\ p$



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

$\boldsymbol{k = 1.}$

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

ϕ is SAT

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

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

  increase k

 $Q := S_0(s_0)$;

else

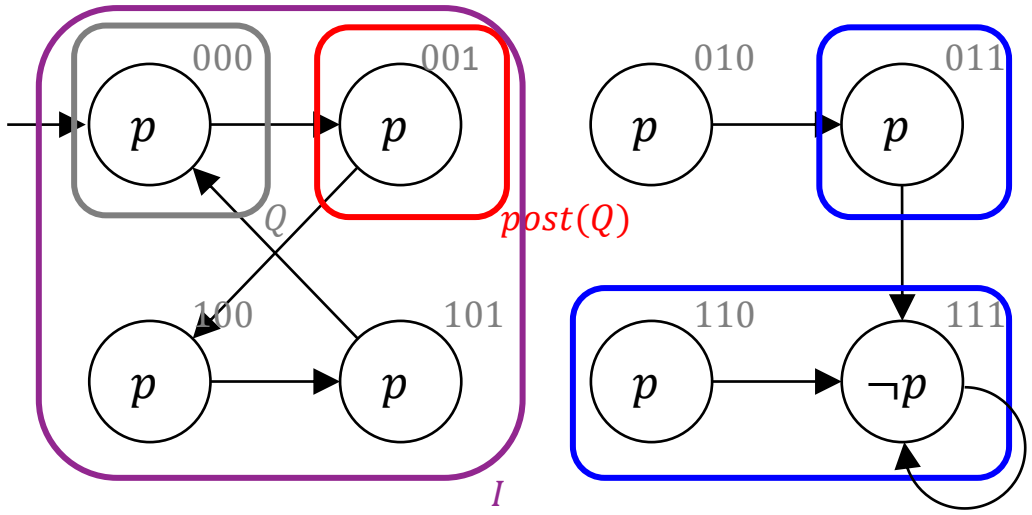
  I = interpolate I A and B;

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

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

Example $\mathbf{AG}\ p$

$x_1 x_2 x_3$



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

$\boldsymbol{k = 2.}$

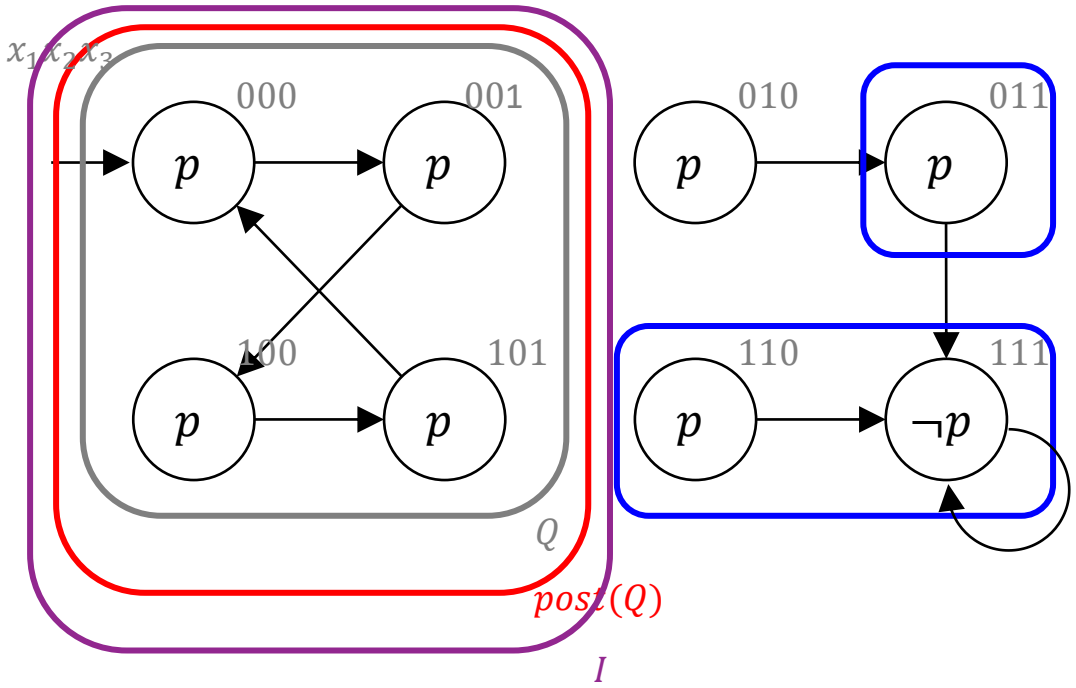
$Q = \neg x_1 \wedge \neg x_2 \wedge \neg x_3 = \{000\}.$

ϕ is UNSAT

Invariant checks 2nd bit: $I = \neg x_2$

if $A \wedge B$ is SAT **then**
  if $Q = S_0$ then return "$M \not\models \mathrm{AG}\ p$";
  increase k
 $Q := S_0(s_0)$;
else
  I = interpolate I A and B;
  if $I(s_0) \rightarrow Q$ then return "$M \models \mathrm{AG}\ p$";
 $Q := Q \vee I(s_0)$;

Example $\mathbf{AG}\ p$



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

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

  increase k

  $Q := S_0(s_0)$;

else

  I = interpolate I A and B;

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

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

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

$\boldsymbol{k = 2.}$

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

ϕ is UNSAT

$\boldsymbol{I = \neg x_2 = Q.}$

Algorithm terminates.

How Did I Pick the Interpolants?

What I did

- Start with $\mathrm{A} = \ postimg(Q)$
- Perform each of the following steps
  1. Can I throw away x_3? (Is $(\exists x_3.A) \cap B = \emptyset$?) If yes, $A \ := \exists x_3.A$
  2. Can I throw away x_2? If yes, $A \ := \exists x_2.A$
  3. Can I throw away x_1? If yes, $A \ := \exists x_1.A$

This hack only works because the postimg(Q) is a state or a cube!

In the homework, you will get CNFs. Try getting rid of clauses.

Note that A is always a valid interpolant.