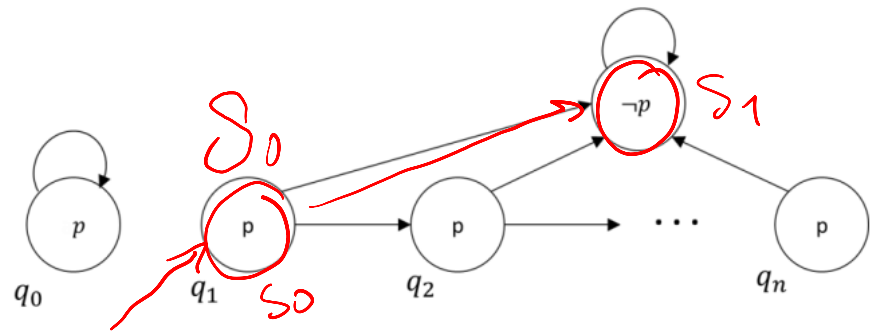
Model Checking with Inductive Invariants

Consider the following synchronous Kripke structure K.



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.

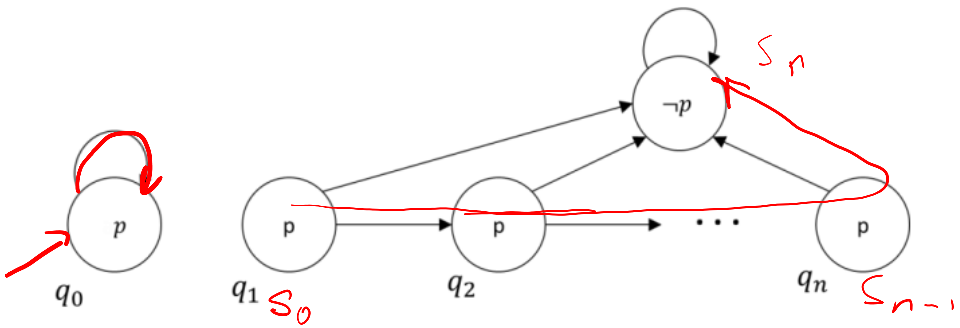
$$path_k(s_0, \dots, s_k) = S_0(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$$

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

$k = 1.$

$S_0(s_0) \wedge R(s_0, s_1) \wedge \neg p(s_0) \vee \neg p(s_1)$

Consider the following synchronous Kripke structure K.



We wish to prove that p is always true.

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.

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

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

UNSAT

SAT

UNSAT for k = 3 and n = 2

$k = n+1$

$k=3$

Homework Results

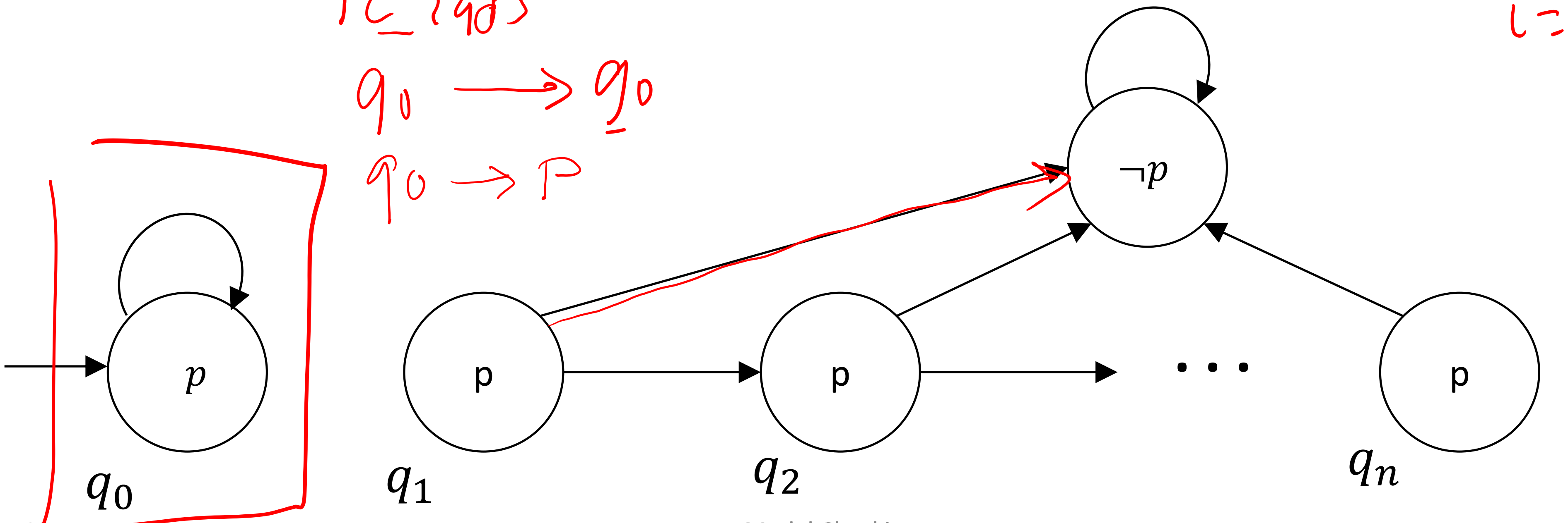
- Find the results here: https://cloud.tugraz.at/index.php/s/zeEgt8ptcRQCXEW
- Confused? Write an email to modelchecking@iaik.

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.



Inductive Invariant

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

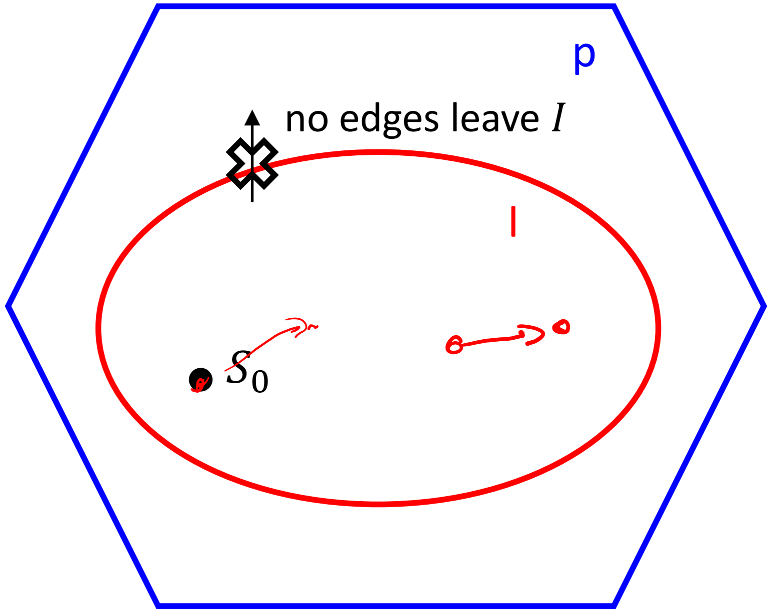
Definition. $I \subseteq S$ is an inductive invariant for $\mathrm{AG}\, p$ if

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

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

In formulas:

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



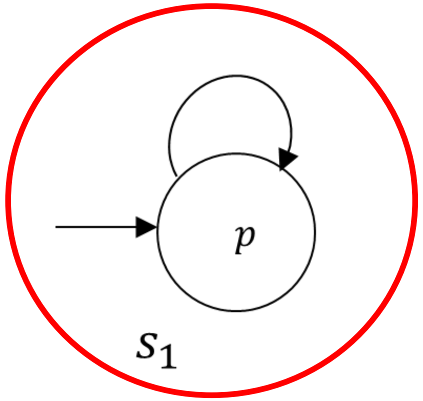
Inductive Invariant

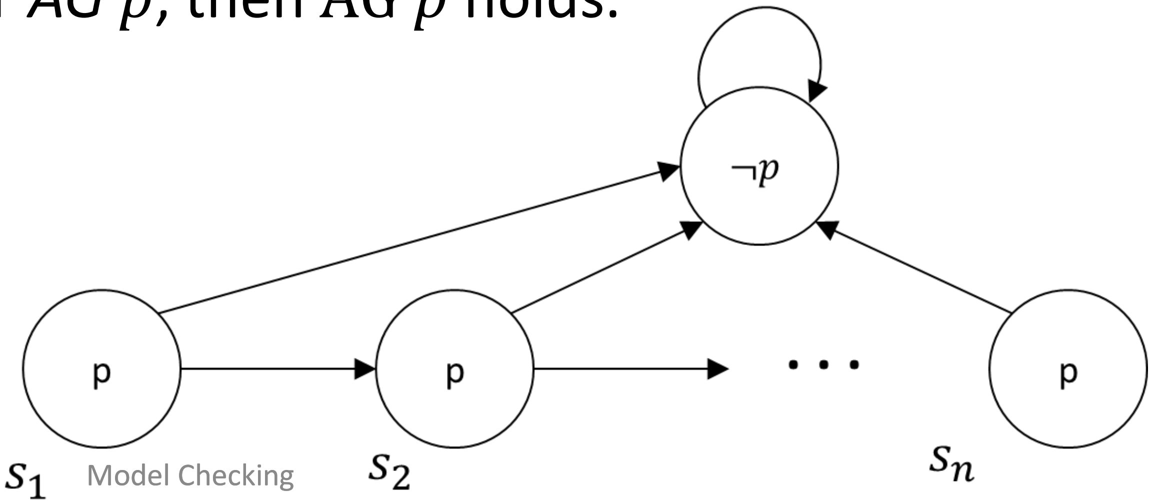
Remember $postimage(Q) = \{ s' \mid \exists s. R(s, s')\}$ (see Chapter 5).

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

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

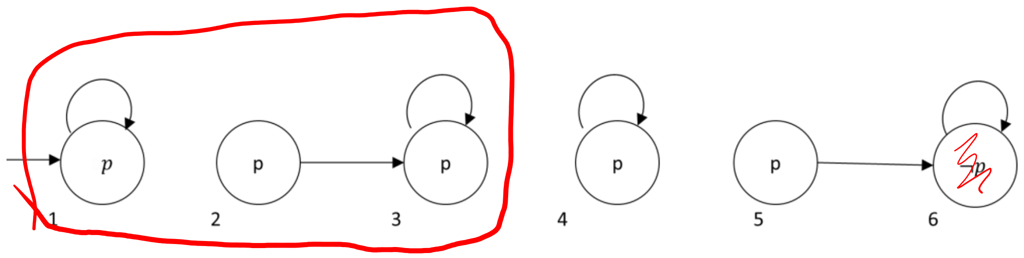
If there is an inductive invariant for *AG* p, then $\mathrm{AG}\ p$ holds.

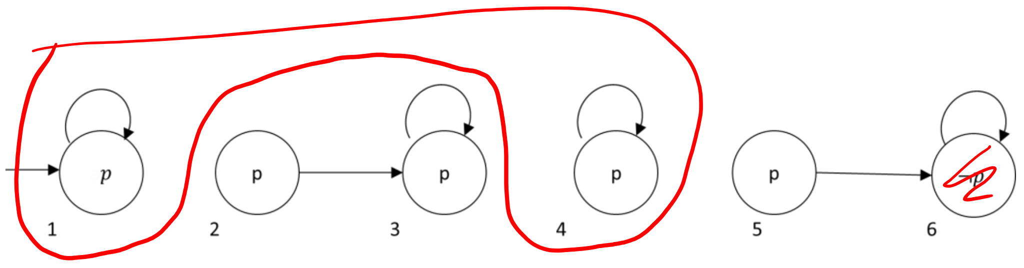




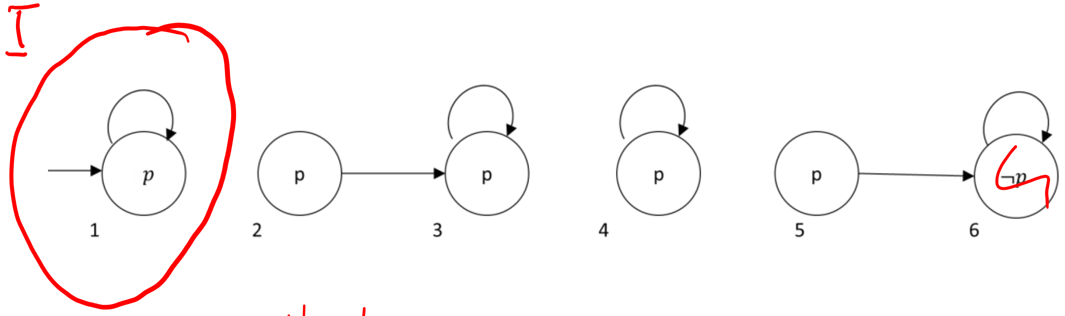
Inductive

Multiple Invariants



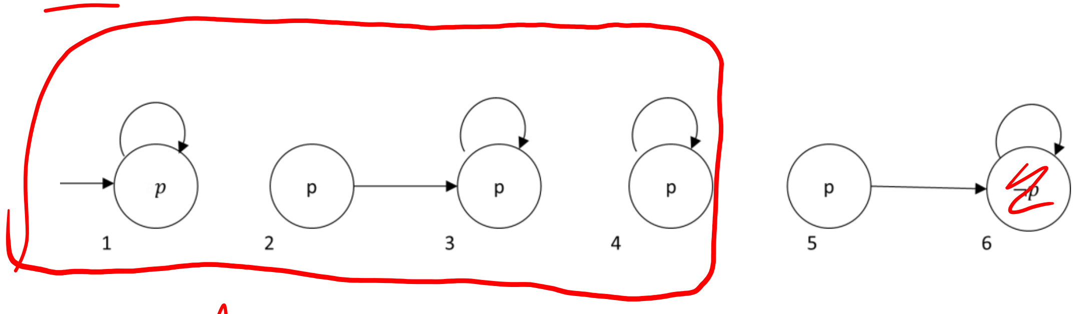


I



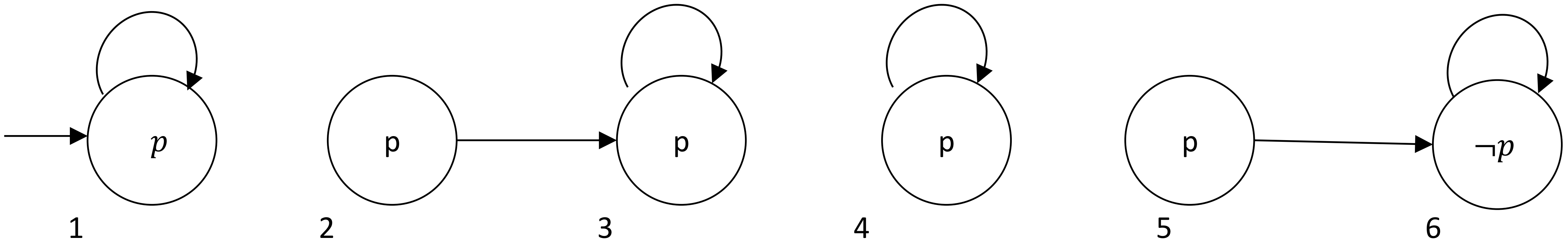
Smallest
strongest
= Reachable states

I



Largest
weakest.
states that cannot reach ¬p

Strongest & Weakest Invariant



Smallest (strongest) invariant is reachable state
Largest (weakest) invariant is states that cannot reach $\neg p$

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

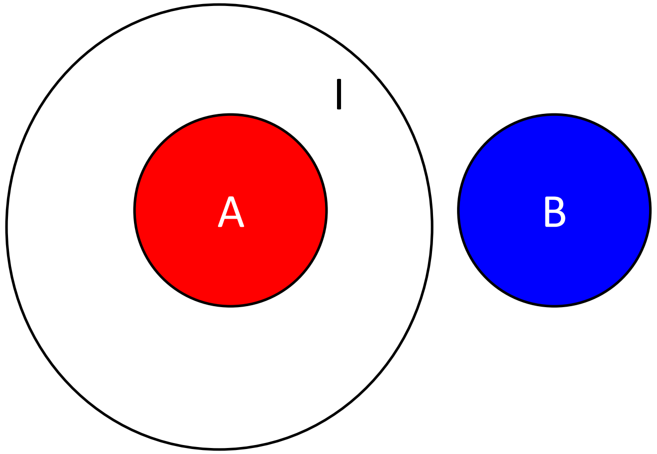
- Find Interpolants I such that
  - States reachable in k steps are in I
  - no bad states are in I
- Interpolants are (good) overapproximation of post-image computation

Interpolant

William Craig, 1957

Definition. Given formulas A, B such that $A \wedge B = \bot$, 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



Example. Let

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

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

$I = a_2 \wedge \neg a_3$

1. $A \rightarrow I$ ✓

2. $I \wedge B =$ FALSE ✓

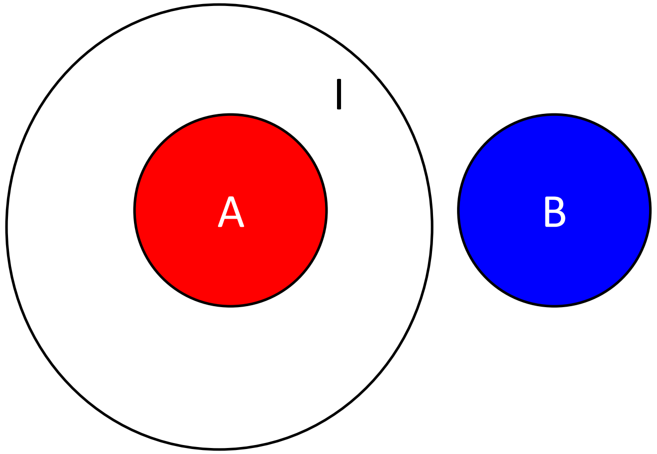
3. ✓

Interpolant

William Craig, 1957

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



Example. Let

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

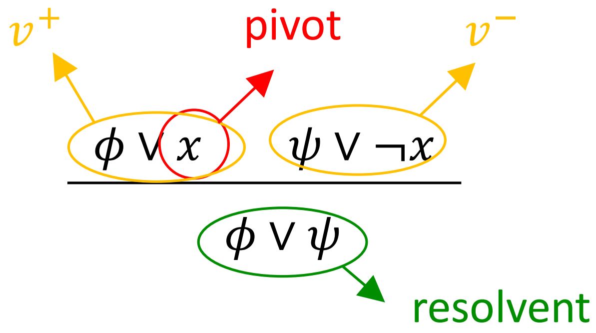
$\mathrm{B} = (\neg a_2 \vee a_3) \wedge (a_2 \vee a_4) \wedge \neg a_4$.

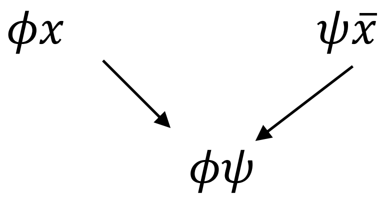
$A \wedge B$ is not satisfiable.

$\neg a_3 \wedge a_2$ 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

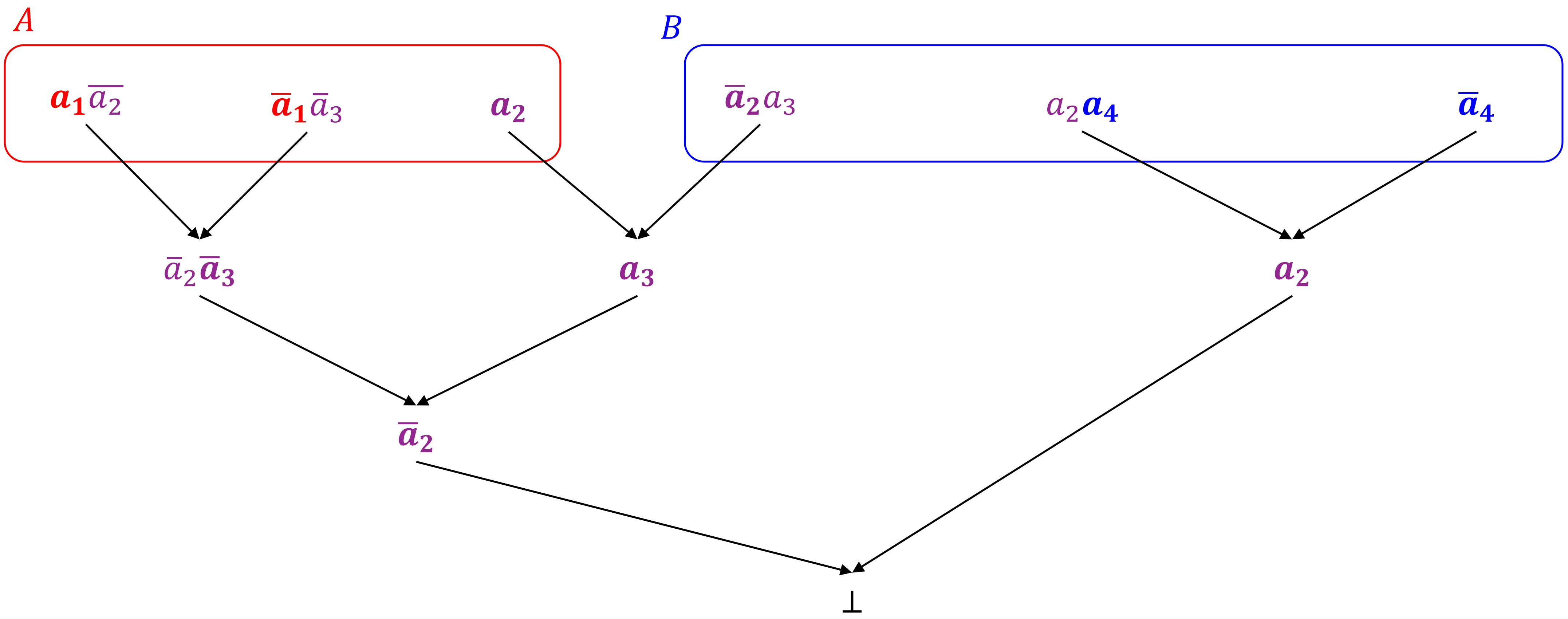
Algorithm. Go through resolution proof top-down.

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.

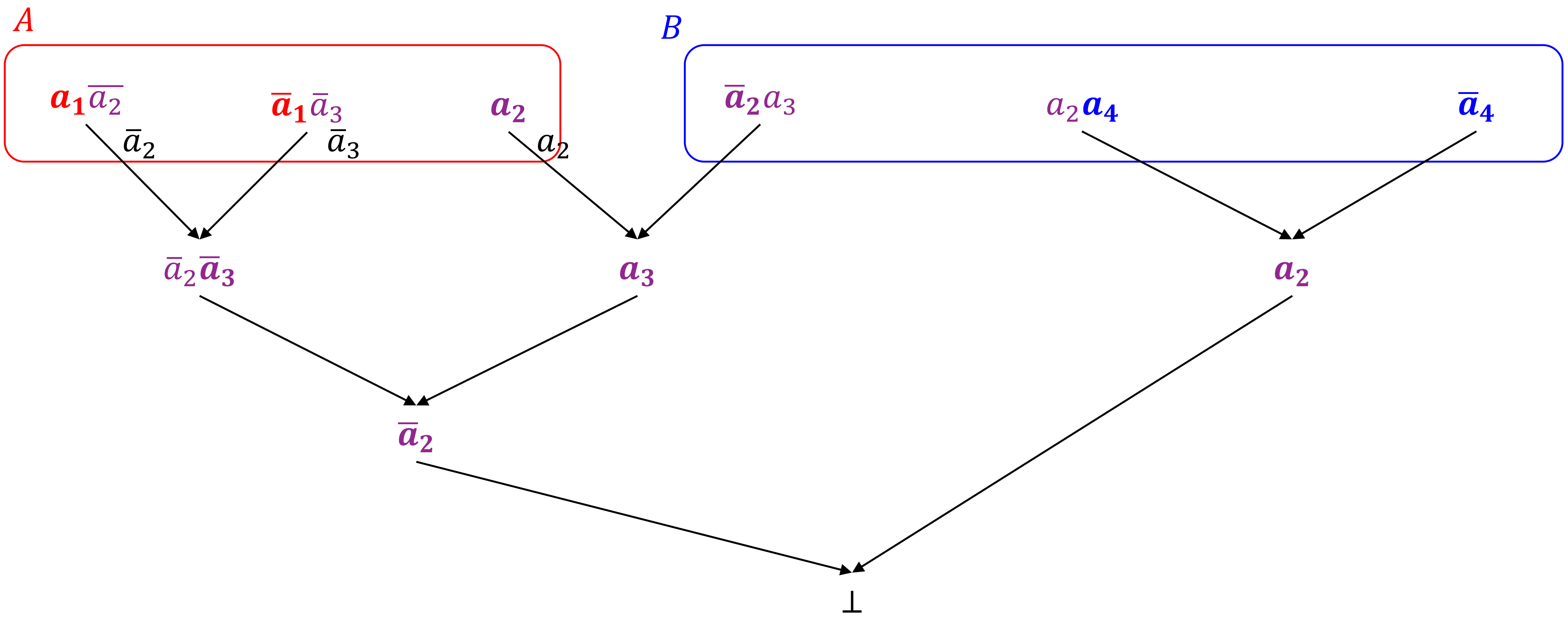
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

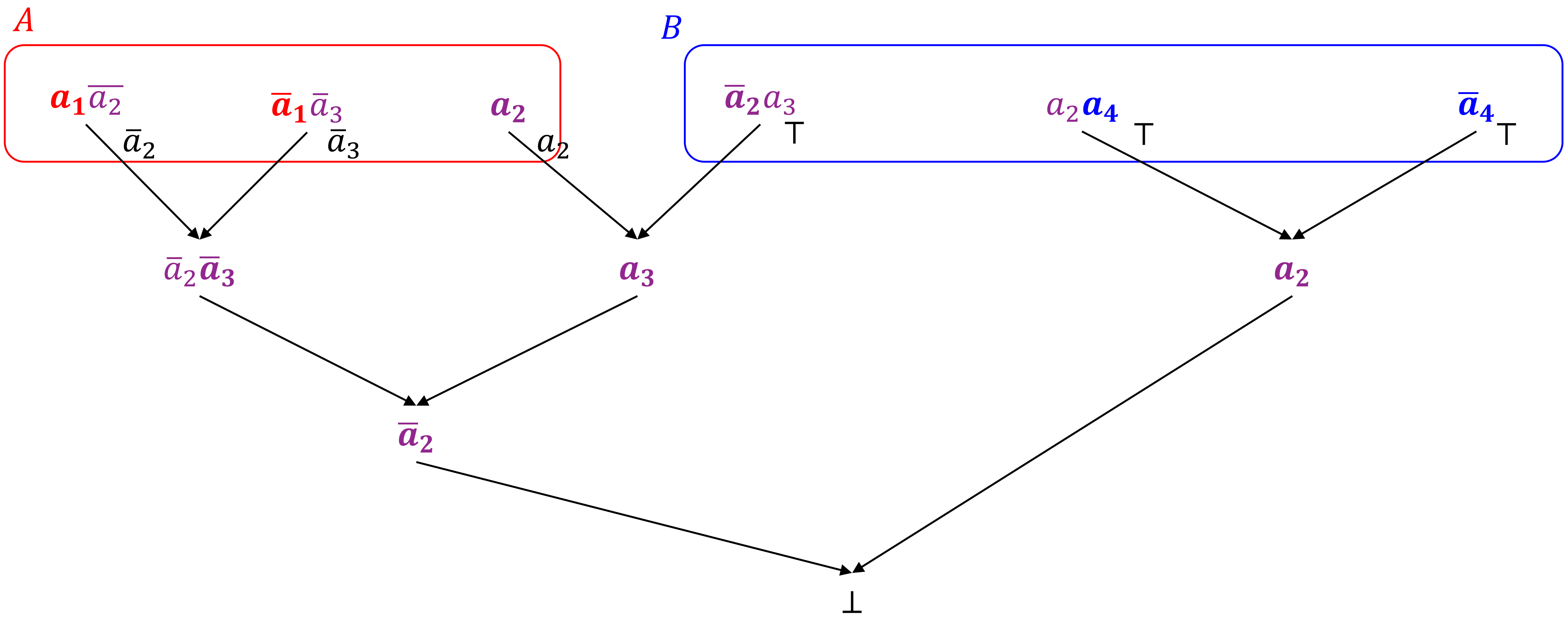
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

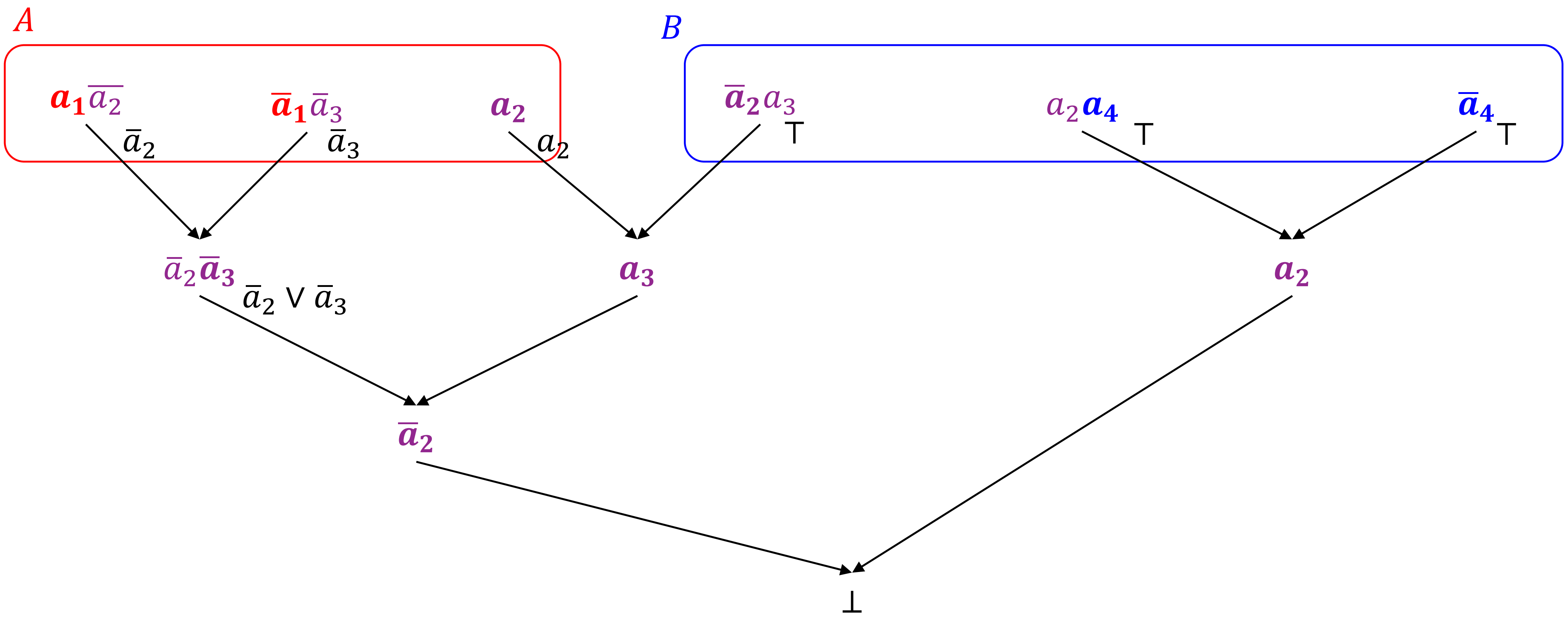
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^{+}) \wedge Itp(v^{-})$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^{+}) \vee Itp(v^{-})$



Interpolation Example

Algorithm. Go through resolution proof top-down.

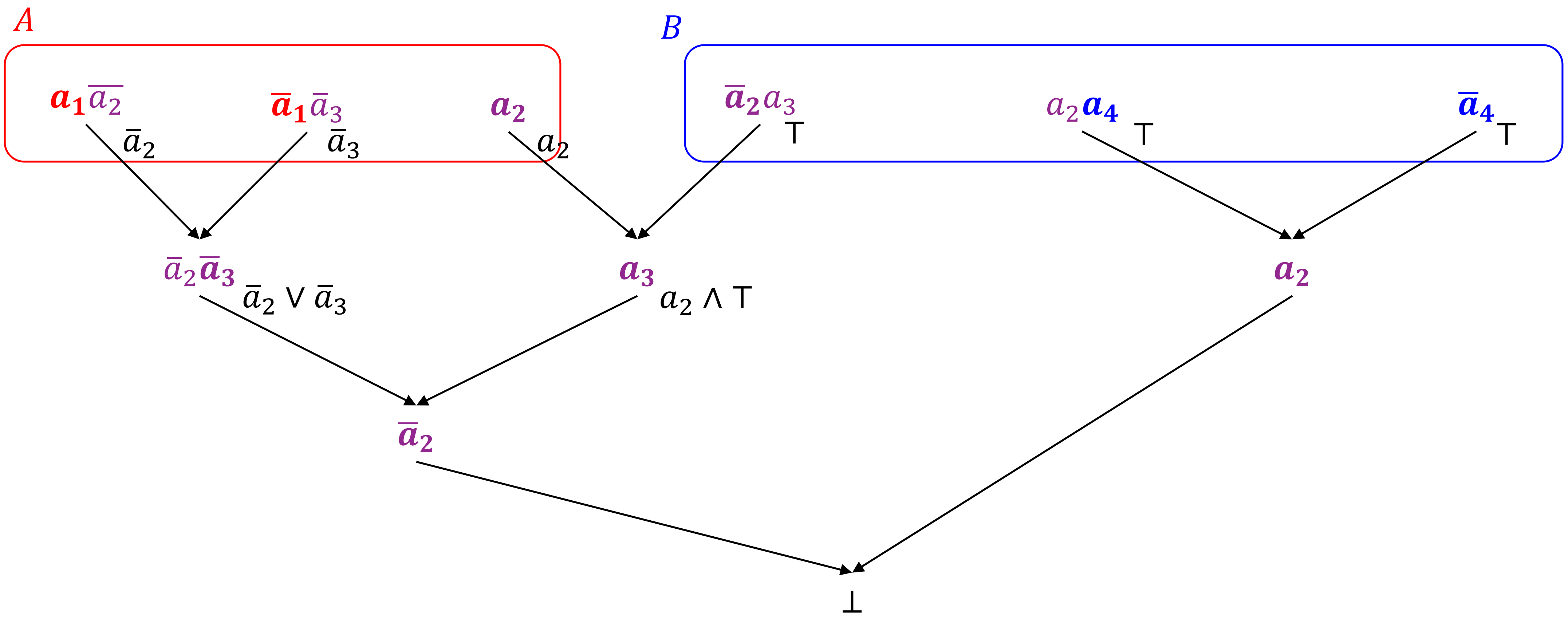
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



Interpolation Example

Algorithm. Go through resolution proof top-down.

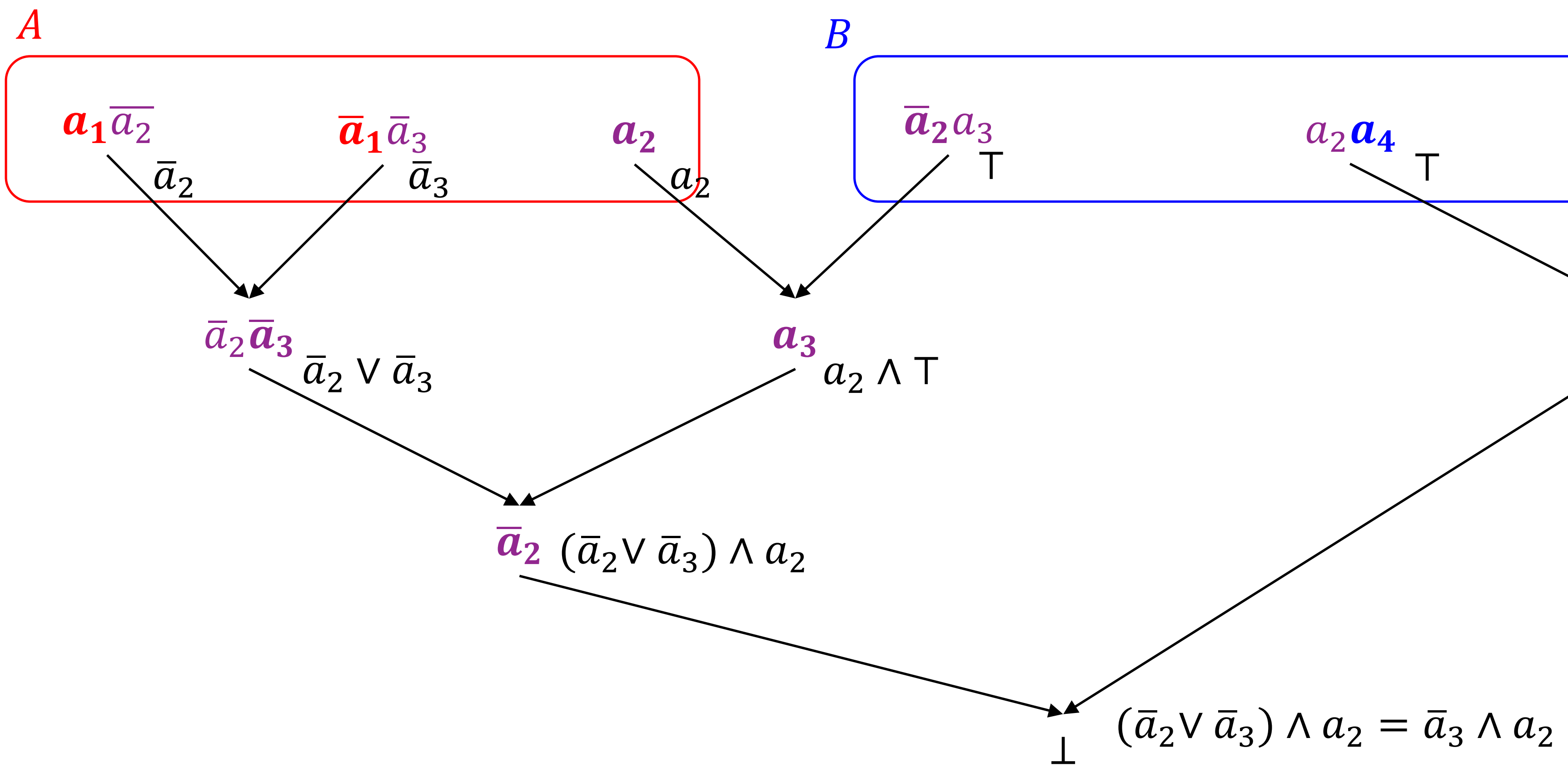
1. If leaf v is labeled $C \in A$, then $Itp(v) = C|B$
2. If leaf v is labeled $C \in B$, then $Itp(v) = \top$
3. If node v has pivot variable $x \in B$ then $Itp(v) = Itp(v^+) \wedge Itp(v^-)$
4. If node v has pivot variable $x \notin B$ then $Itp(v) = Itp(v^+) \vee Itp(v^-)$



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



Reachability Checking with Interpolation

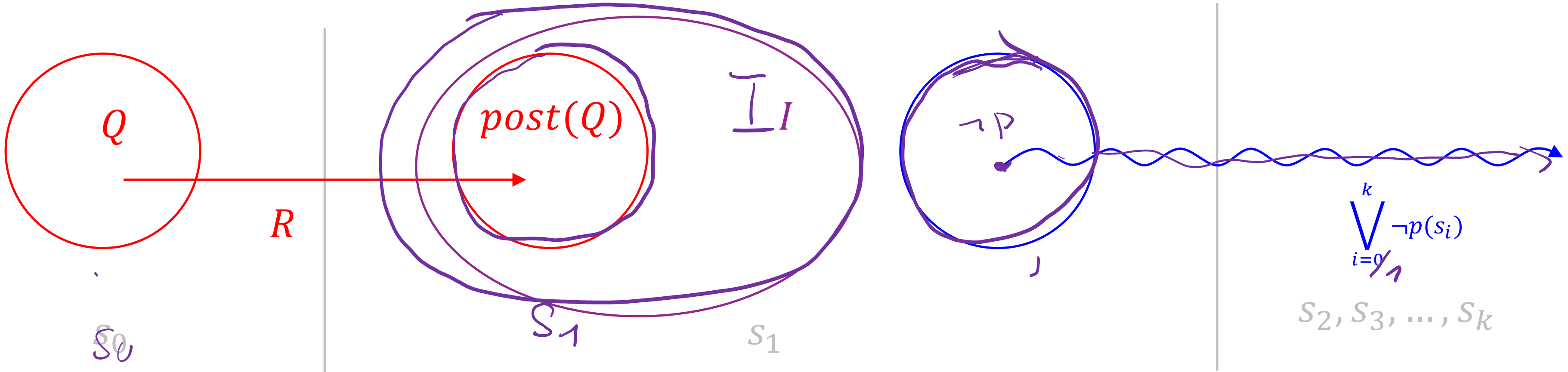
Recall BMC check for $\neg\mathbf{AG}p$:

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

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 $\neg \mathbf{AG}p$:

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

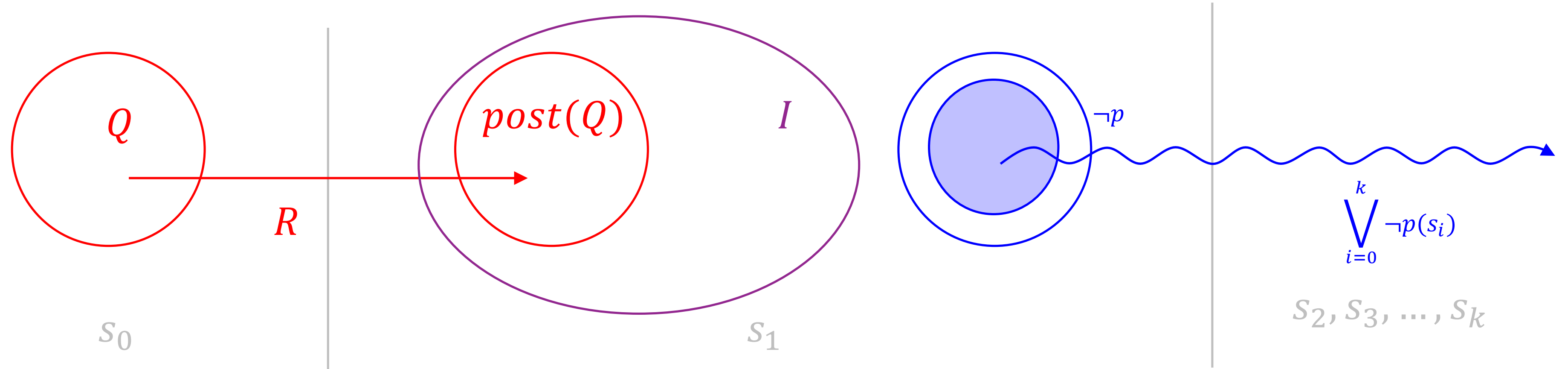
Start from Q such that $Q \vDash p$

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

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: $I \supseteq post(Q)$



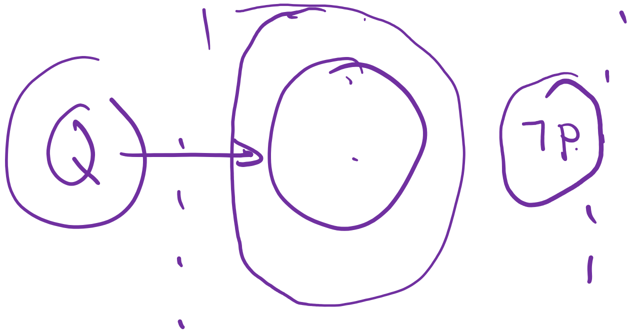
Interpolant Reachability Idea

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

1. Start with $Q = S_0$
2. If ϕ is satisfiable, $\neg p$ is **reachable**
3. If not, set Q to I
4. If I remains unchanged, p **cannot be reached** (Interpolants are approximation to post-image)
5. If ϕ is eventually satisfiable, 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 "$M \nvDash \text{AG}\, p$ ";

  k := 1;

  Q := $S_0(s_0)$;

  while true **do**

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

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

  if $A \wedge B$ is SAT **then** // $\neg p$ can be reached from Q

    if $Q = S_0$ then return "$M \nvDash \text{AG}\, p$"; // $\neg p$ can be reached from S_0

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

    Q := $S_0(s_0)$;

  else

    compute interpolant I for A and B;

    If $I(s_0) == Q$ then return "$M \vDash \text{AG}\, p$"; // Reached the fixpoint of overapproximated reachability?

    Q := $Q \vee I(s_0)$; // Another step of overapproximated reachability?

  end if

  end while

end procedure

10.4.4 Correctness

If CraigReachability returns "$M \vDash AG\ p$" then $M \vDash AG\ p$

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 "$M \nvDash AG\ p$" then $M \nvDash AG\ p$

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

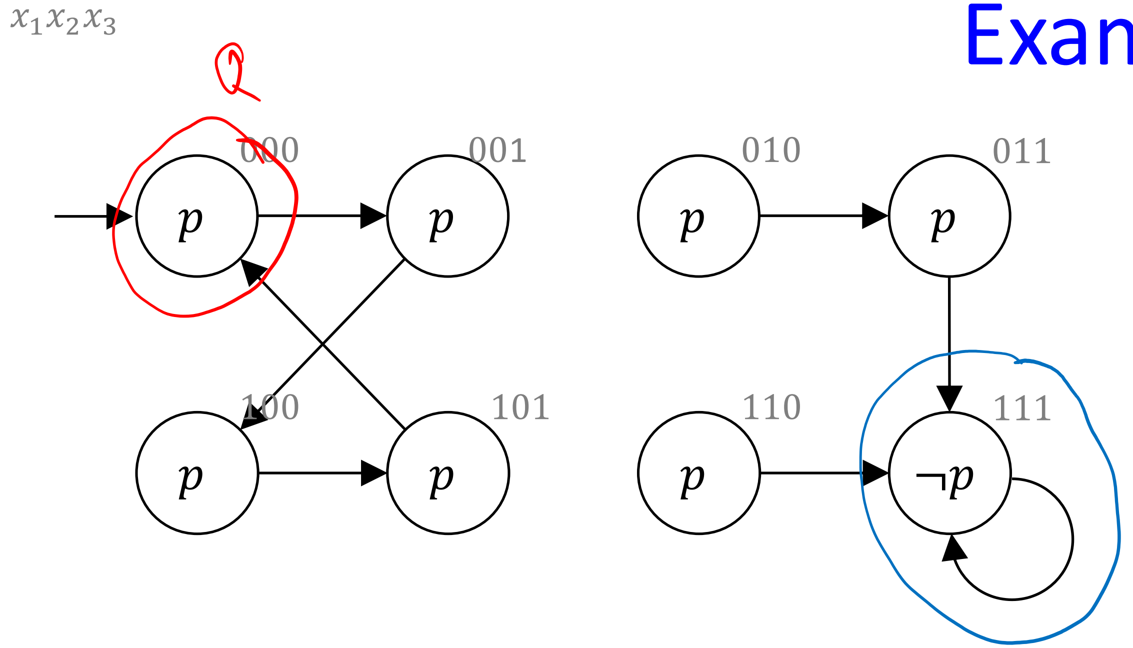
CraigReachability terminates

Note that k increases.

If $M \nvDash AG\ p$, there is a path of length l to $\neg p$ and we will find it when $l = k$.

Suppose $M \vDash AG\ p$. 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 AG p



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

$Q = S_0$ $k = 1$

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

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

increase k

$Q := S_0(s_0)$;

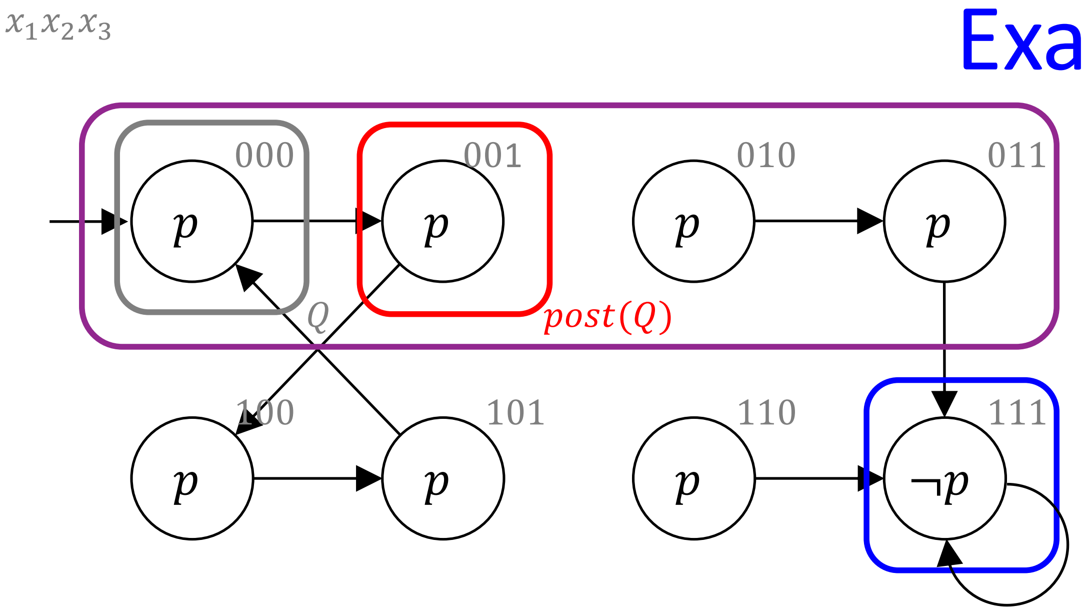
else

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***



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

$\boldsymbol{k}$ **=1.**

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

ϕ is UNSAT

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

$I = \neg x_1$

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

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

increase k

$Q := S_0(s_0)$;

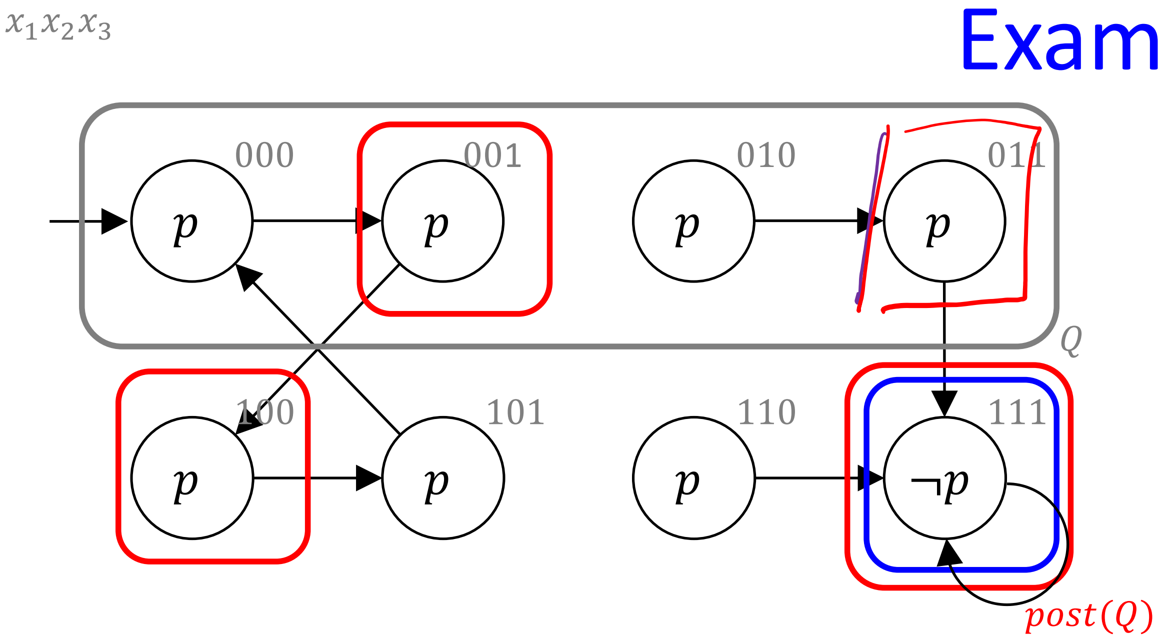
else

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***



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

  if $Q = S_0$ then return "$M \nvDash$ 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 "$M \vDash$ AG p";

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

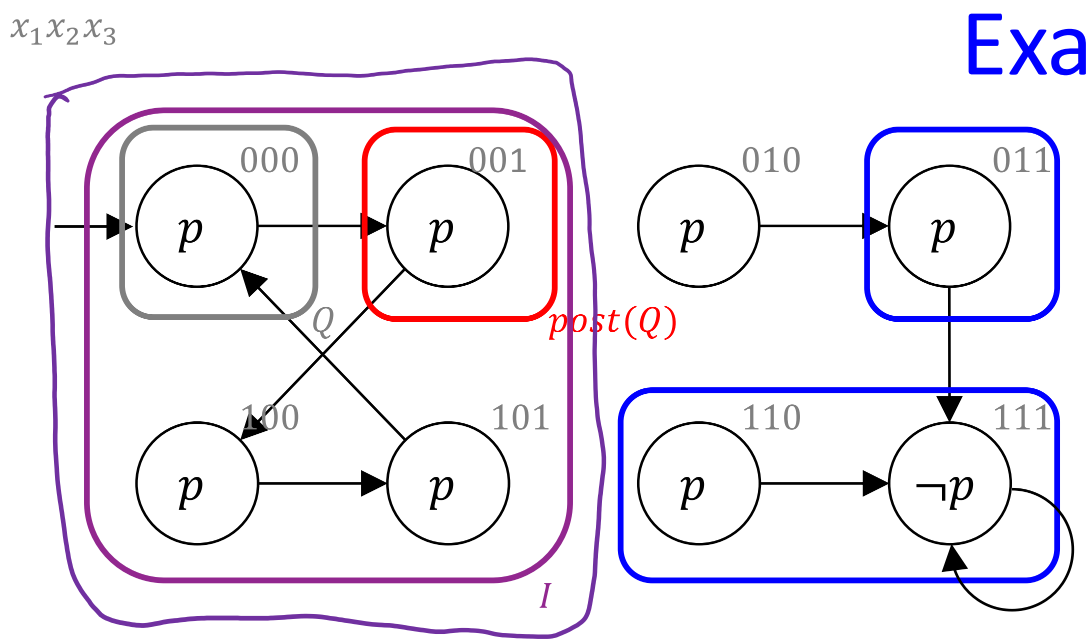
$\phi = Q(s_0) \wedge R(s_0, s_1) \wedge$
$\bigwedge_{i=1}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=1}^{k} \neg p(s_i)$.

$\boldsymbol{k}$ **=1.**

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

ϕ is SAT

Example **AG *p***



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

$\boldsymbol{k = 2.}$

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

ϕ is UNSAT

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

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

if $Q = S_0$ then return "$M \nvDash \text{AG}\, p$";

increase k

$Q := S_0(s_0)$;

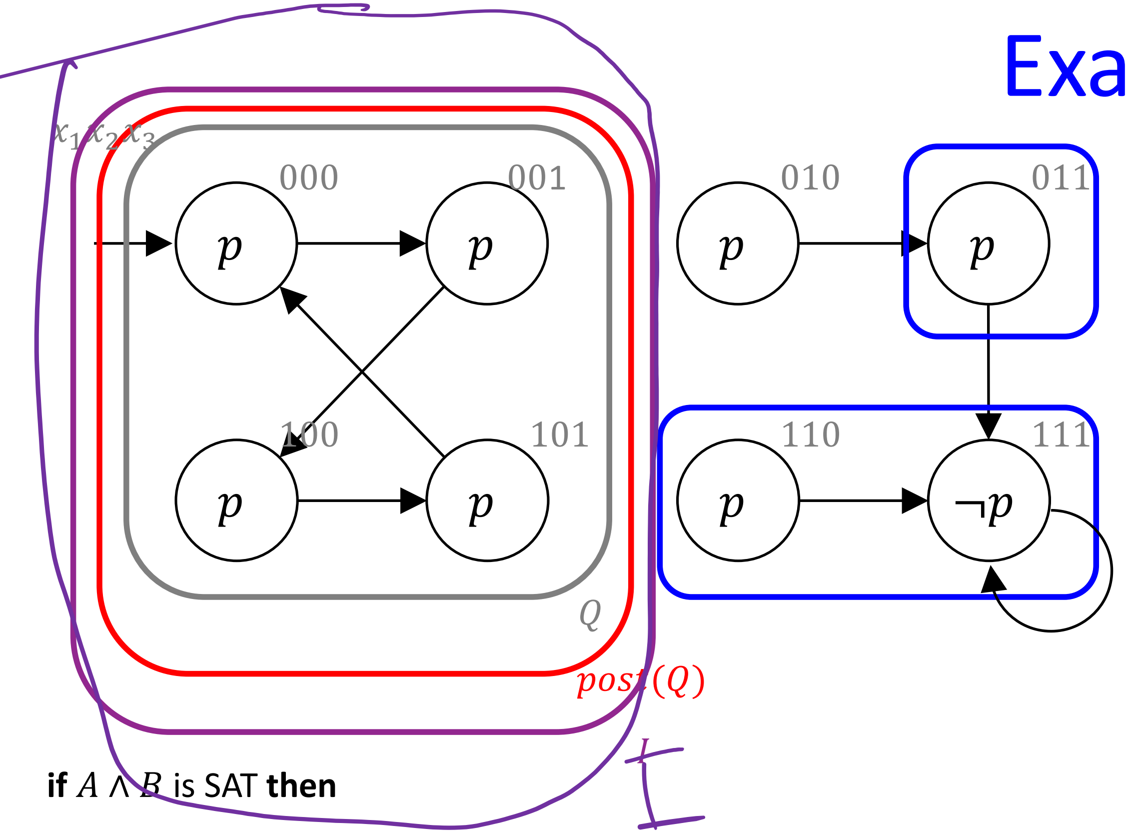
else

compute interpolant I for A and B;

if $I(s_0) \to Q$ then return "$M \vDash \text{AG}\, p$";

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

Example AG p



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";

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

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

$\boldsymbol{k = 2}$.

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

ϕ is UNSAT

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

Algorithm terminates.

How Did I Pick the Interpolants?

What I did

- Start with $\mathrm{A} = \; postimg(Q) \; \setminus Q$
- Can I throw away x_3? (Does $(\exists x_3 . A) \cap B$ hold?) If yes, $A \; \coloneqq \exists x_3 . A$
- Can I throw away x_2? If yes, $A \; \coloneqq \exists x_2 . A$
- Can I throw away x_1? If yes, $A \; \coloneqq \exists x_1 . A$

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