COL:750

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage

https://priyanka-golia.github.io/teaching/COL-750/index.html

Bounded Model Checking with SAT (BMC)

General idea:

Fix a K

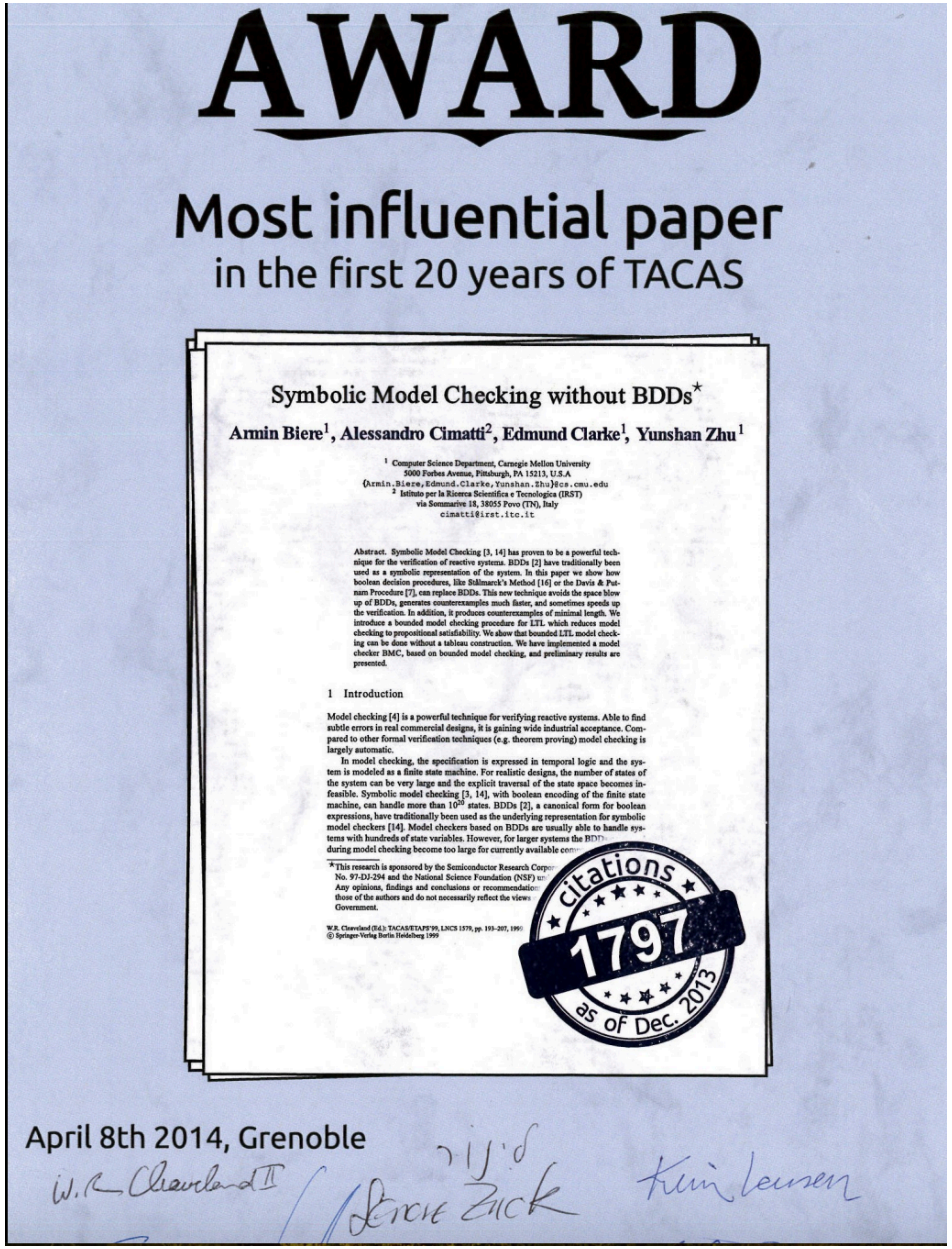
1. Convert transition system to propositional encoding — unroll for path length k
2. Convert temporal formula along the states to propositional encoding for k length
3. Using SAT Solvers look for counterexamples
4. Found a counterexample :

    Return counterexample

5. Else:

    K= K+1

6. At some point, check if $K \overset{?}{\geq} rd$ Return True, Else: K = K+1 For safety property.



extensions to completeness

diameter checking,

k-induction,

interpolation –

SAT based model checking without unrolling: IC3

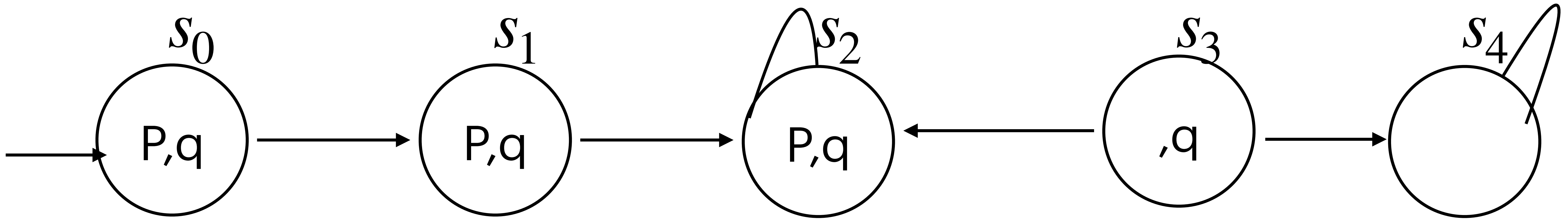
Prove or Disprove that the recurrence diameter is a completeness threshold for properties of the form $\forall\Diamond P$.

Induction

For verifying safety property/ verifying reachability properties.

Often the completeness threshold is very large.

Exploring techniques that requires fewer unwinding.



$M \models \forall \Box p \qquad M \models \forall \Box q$

Induction principles —

To prove the claim $Q(n)$ for all values of some parameter n.

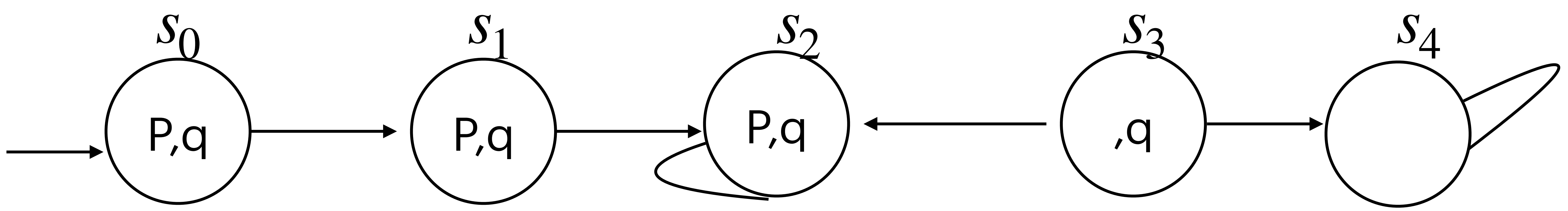
We need to show the validity of these cases

Base case — Q(0)

Inductive step case — Q(n-1) -> Q(n)

Induction

For verifying safety property/ verifying reachability properties.



$M \models \forall \Box p$

Induction principles

To prove $\forall \Box p$, we prove that $p(\pi(n))$ holds for $\forall n$

Given M, π denotes a path in M.

i^{th} state in π is $\pi(i)$.

$p(\pi(i))$ is to denote that property p holds in state $\pi(i)$

Idea — base case (initial states). $p(s_0)$ holds.

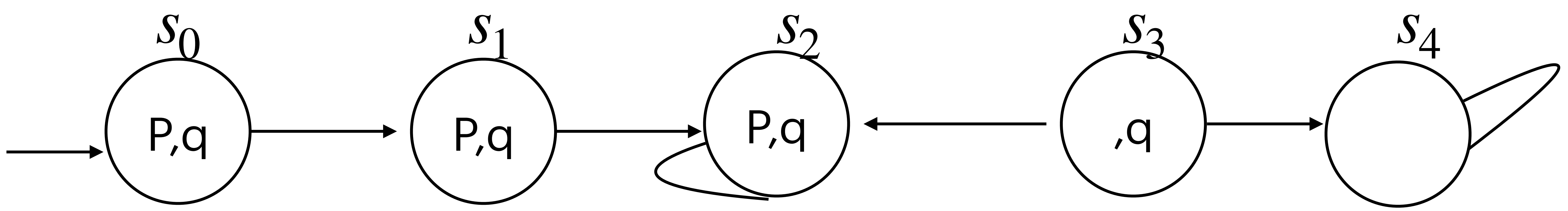
Inductive step. Assuming $p(\pi(n-1))$ holds, $p(\pi(n))$ must hold.

all the states labelled with p, that is, $\{0,1,2,\}$

All the states where $p(\pi(n-1))$ holds!

$p(\pi(n))$ must hold true, which will be successor of $\{0,1,2\}$ — $\{1,2\}$

Induction For verifying safety property/ verifying reachability properties.



$M \models \forall \Box p$

To prove $\forall \Box p$, we prove that $p(\pi(n))$ holds for $\forall n$

Idea — base case (initial states). $p(s_0)$ holds.

Inductive step. $p(\pi(n-1))$ holds, states labelled with p, that is, $\{0,1,2\}$

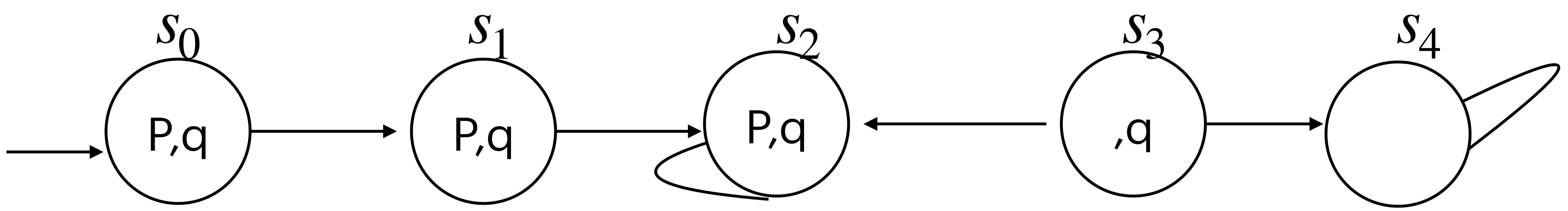
$p(\pi(n))$ must hold true, which will be successor of $\{0,1,2\}$ — $\{1,2\}$

Validity of the base case and inductive step?

Validity of F $\equiv \neg F$ being UNSAT.

Induction

For verifying safety property/ verifying reachability properties.



$M \vDash \forall \square p$

To prove $\forall \square p$, we prove that $p(\pi(n))$ holds for $\forall n$

For base case, check satisfiability of

$$s_o \wedge \neg p(s_o) \; \forall s_o \in I \qquad (p_o \wedge q_o) \wedge \neg p_o$$

If this is UNSAT, then all initial state satisfy p.

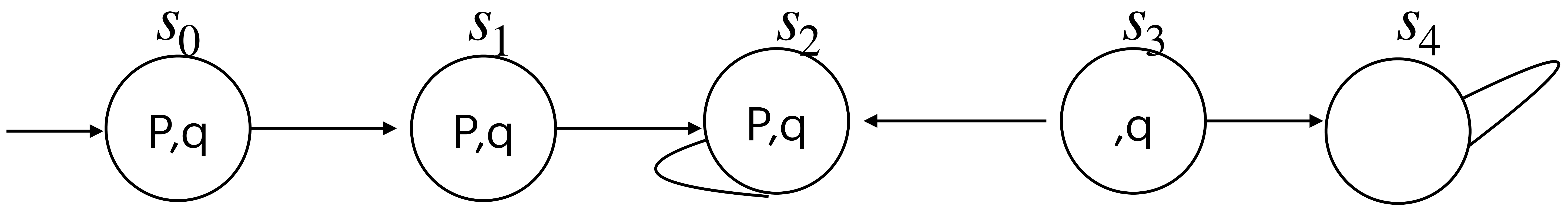
Inductive case – observation $T(\pi(n-1), \pi(n))$ holds.

Let s be the states in $\pi(n-1)$ Let s' be the states in $\pi(n)$

Validity of $p(s) \wedge T(s, s') \to p(s')$ CheckSAT($p(s) \wedge T(s, s') \wedge \neg p(s')$)

Induction

For verifying safety property/ verifying reachability properties.



$M \models \forall \Box p$

To prove $\forall \Box p$, we prove that $p(\pi(n))$ holds for $\forall n$

Inductive case — observation $T(\pi(n-1), \pi(n))$ holds.

Let s be the states in $\pi(n-1)$ Let s' be the states in $\pi(n)$

Validity of $p(s) \wedge T(s, s') \rightarrow p(s')$ CheckSAT($p(s) \wedge T(s, s') \wedge \neg p(s')$) $\forall s \in S$, s.t. $P(s)$

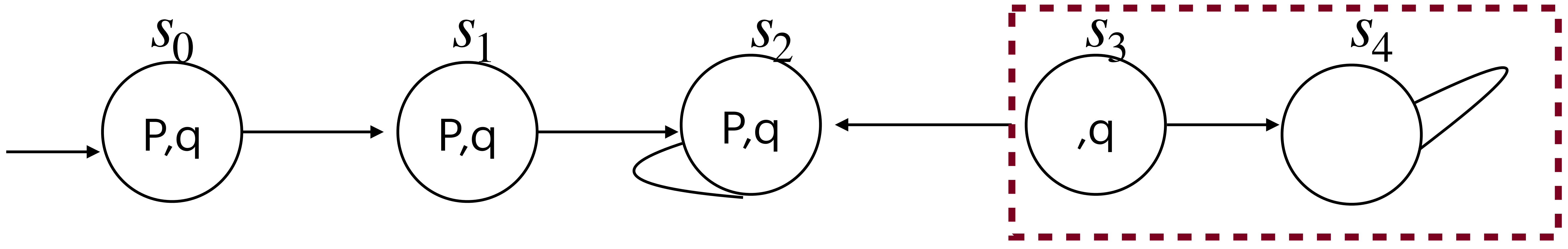
Inductive steps — any reachable state in model M

$$\text{CheckSAT}((p_o \wedge \neg p_1') \vee (p_1 \wedge \neg p_2') \vee (p_2 \vee \neg p_3') \wedge T)$$

This requires only a single copy of T.

Induction

For verifying safety property/ verifying reachability properties.



$M \models \forall \Box q$

To prove $\forall \Box q$, we prove that $q(\pi(n))$ holds for $\forall n$

Base case: $q_o \wedge \neg q_o$ UNSAT

Inductive case: $((q_o \wedge \neg q_1') \vee (q_1 \wedge \neg q_2') \vee (q_2 \wedge \neg q_3') \vee (q_3 \wedge \neg q_4')) \wedge T$ SAT

$M \overset{?}{\models} \forall \Box q$

Inductive step case $-q(\pi(n-1)) \rightarrow q(\pi(n))$

Inductive case also deals with unreachable states

K-Induction For verifying safety property/ verifying reachability properties.

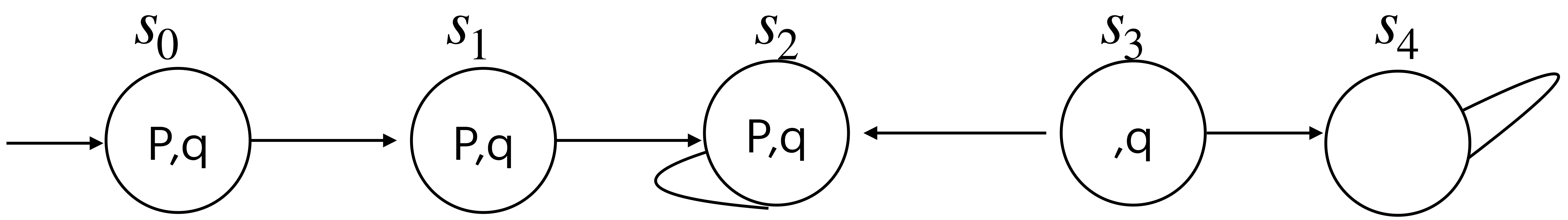
we strengthen the criterion for base case, and weaken the criterion for step case.

Input K, M, F/P

Base case — $p(0) \wedge \ldots \wedge p(K-1)$

Step case — $p(n-K) \wedge \ldots \wedge p(n-1) \rightarrow p(n)$

Any path with k states labelled with p,
is followed by a state labelled with p.



$M \models \forall \Box q$ K = 2

K-Induction

For verifying safety property/ verifying reachability properties.

Base case — $p(0) \wedge \ldots \wedge p(K-1)$ Step case — $p(n-K) \wedge \ldots \wedge p(n-1) \rightarrow p(n)$

Base case — property holds in K states starting from initial state   Same as BMC

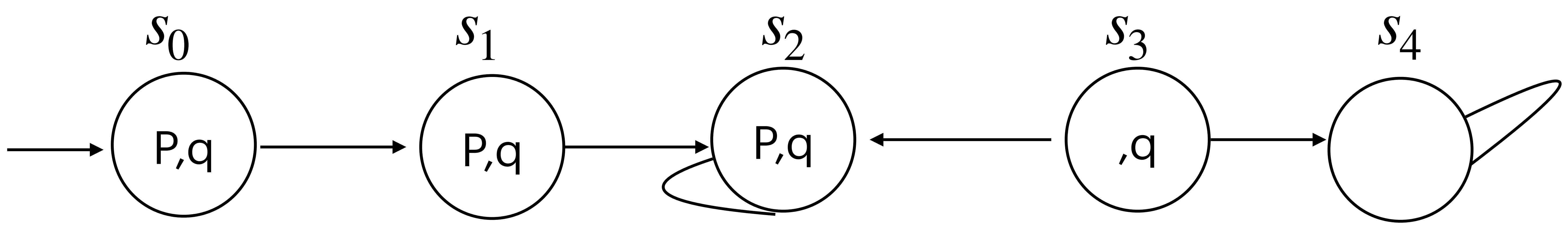
Property q holds true in {0,1}

Inductive step — we need to consider all paths with two states.

Property q holds true in {0,1}, {1,2},{2,2},{3,2}

For each of these paths, q holds in their successor

Property q holds true in {2}



$M \models \forall \Box q$ K = 2

K-Induction

For verifying safety property/ verifying reachability properties.

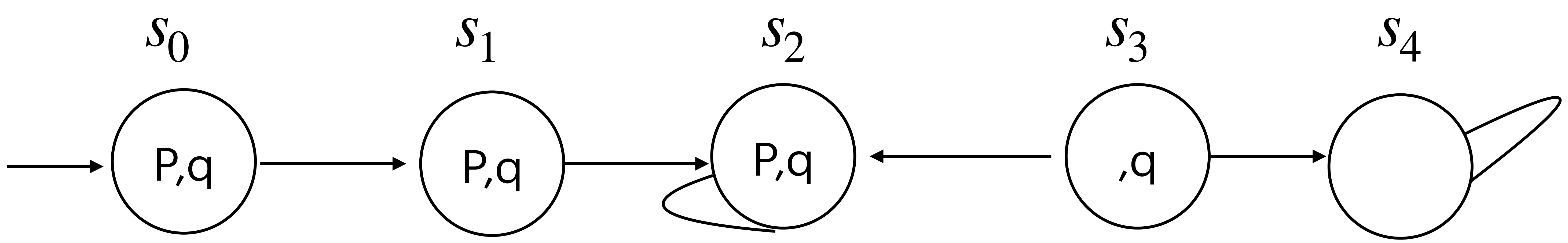
Base case — $p(0) \wedge \ldots \wedge p(K-1)$ Step case — $p(n-K) \wedge \ldots \wedge p(n-1) \rightarrow p(n)$

Base case — property holds in K states starting from initial state Same as BMC

$$M_k \wedge \neg p_k$$

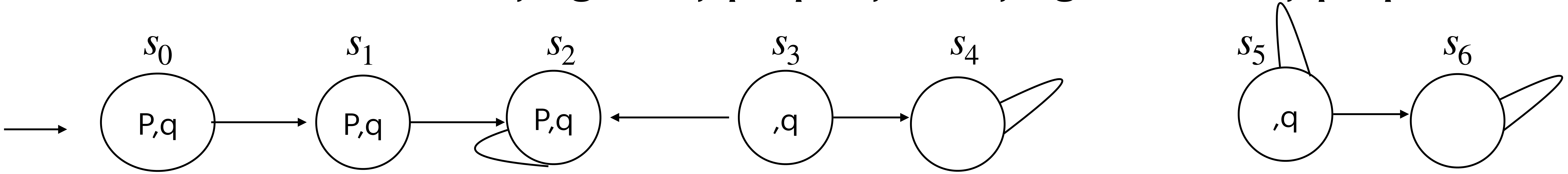
Inductive step — we need to consider all paths with K states.

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



$M \vDash \forall \Box q$ K = 2

K-Induction For verifying safety property/ verifying reachability properties.



$$M \overset{?}{\models} \forall \Box q$$

Neither of the two states are initial states
Nor are they reachable from an initial state.

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

This should be UNSAT!

No, irrespective of K, this is SAT

$$s_o \mapsto 5, \ldots, s_{k+1} \mapsto 5, s_k \mapsto 6$$

Therefore, to obtain completeness — we need to add $\bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^{k} s_i \neq s_j$

Ensuring simple path

K-Induction

For verifying safety property/ verifying reachability properties.

Base case — property holds in K states starting from initial state

$$M_k \wedge \neg p_k \equiv I(s_o) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \neg p_k$$

Same as BMC

Inductive step — we need to consider all paths with K states.

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

- If Base case is SAT, return counterexample.
- If Inductive case is UNSAT, return True.
- Otherwise, increase K and continue.

K-Induction For verifying safety property/ verifying reachability properties.

k-induction extends the capabilities of BMC by not only detecting counterexamples within a bounded number of steps but also proving the absence of such counterexamples, thereby establishing the validity of properties over unbounded executions

Observations

1. We do not need to know the exact reachable states, as long as we can guarantee they meet the property .
2. Beginning of "Property directed" techniques — which is associated with a family of techniques that build inductive invariants automatically

Property Directed Reachability (PDR) is another name for IC3 (Incremental Construction of Inductive Clauses for Indubitable Correctness).
The phrase "property directed" refers to how IC3 works — it constructs inductive invariants incrementally, guided by the property being verified.

Interpolants based Model Checking

Interpolants: Introduced by Craig in 1957

Let A and B be two formulas such that : $A \wedge B \models \bot$

then, there exists a formula I called Interpolant such that:

1. $A \rightarrow I$
2. $I \wedge B \models \bot$
3. $Vars(I) \subseteq Vars(A) \cap Vars(B)$

It acts as a kind of summary or abstraction of A relevant to the contradiction with B.

$A = (p \vee q) \wedge (\neg p \vee r) \qquad B = \neg q \wedge \neg r \qquad I = (q \vee r)$

How to Compute Interpolants!

1. $A \wedge B$ are unsatisfiable.

SAT solver can return resolution proof!

All the initial nodes have in-degree 0. All internal nodes have in-degree 2.
Sink nodes has out-degree 0.

Internal node v, with edges $(v_1, v), (v_2, v)$ implies that v is a resolvent of v_1, v_2

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

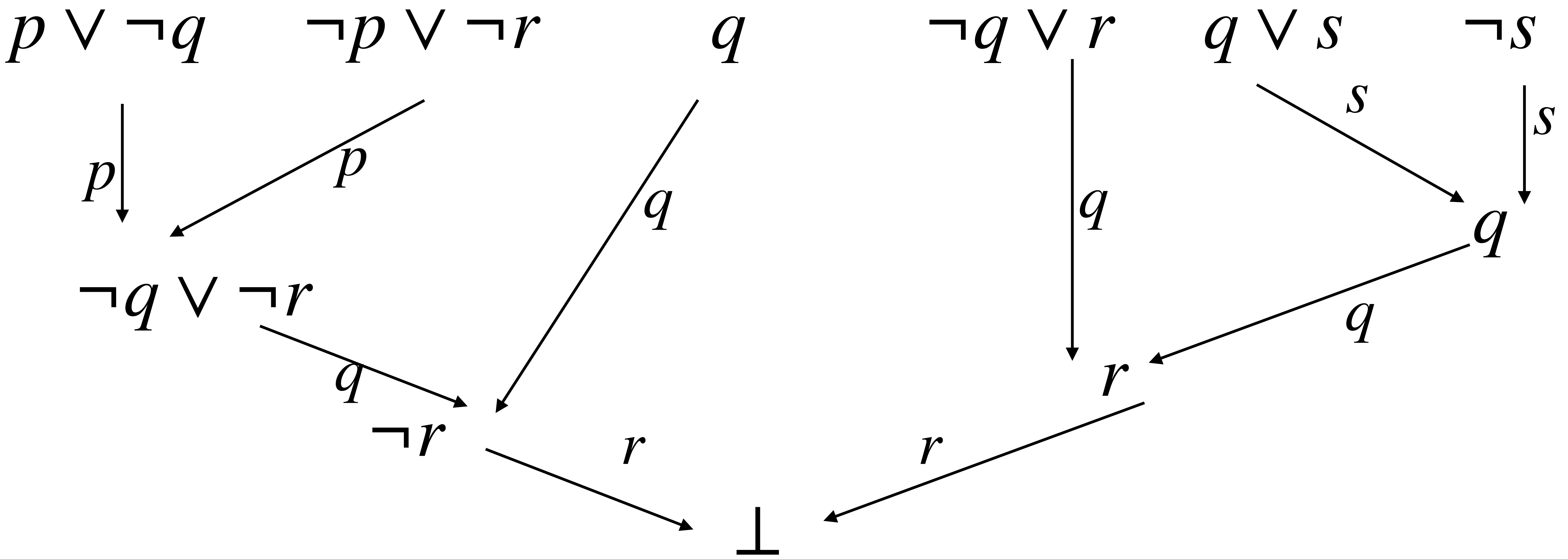
How to Compute Interpolants!

1. $A \wedge B$ are unsatisfiable. SAT solver can return resolution proof!

All the initial nodes have in-degree 0. All internal nodes have in-degree 2. Sink nodes has out-degree 0. Internal node v, with edges $(v_1, v), (v_2, v)$, v is a resolvent of v_1, v_2

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \quad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

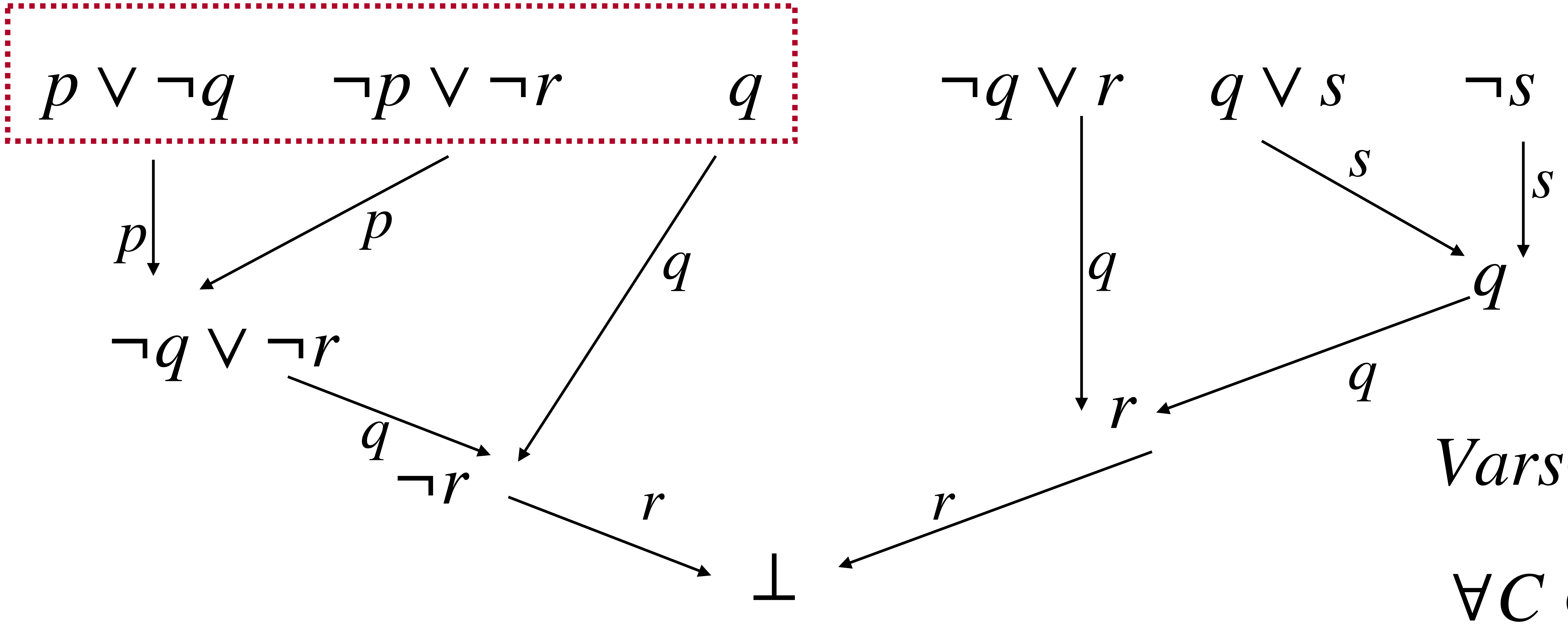
$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



How to Compute Interpolants!

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



$$Vars(I) \subseteq Vars(A) \wedge Vars(B)$$

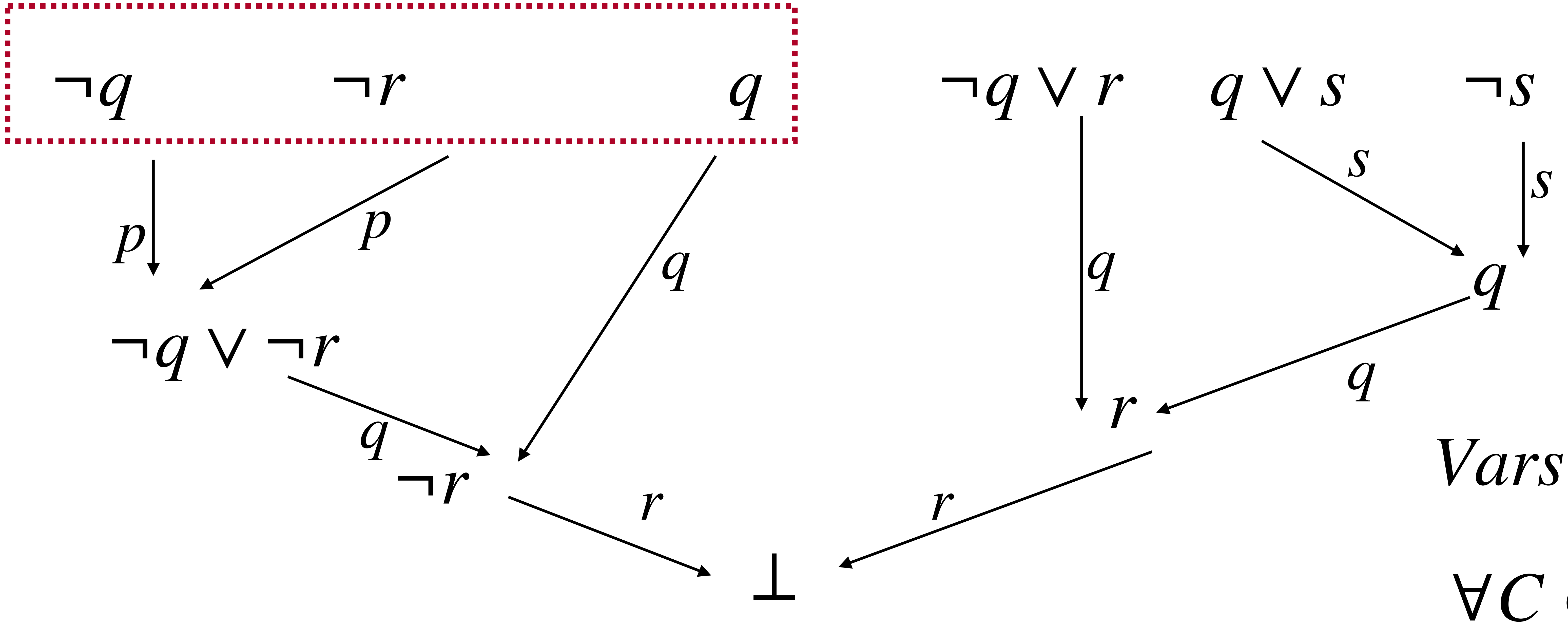
$$\forall C \in Clauses(A), C_{\downarrow(Vars(B))}$$

$$A \rightarrow I$$

How to Compute Interpolants!

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



$$Vars(I) \subseteq Vars(A) \wedge Vars(B)$$

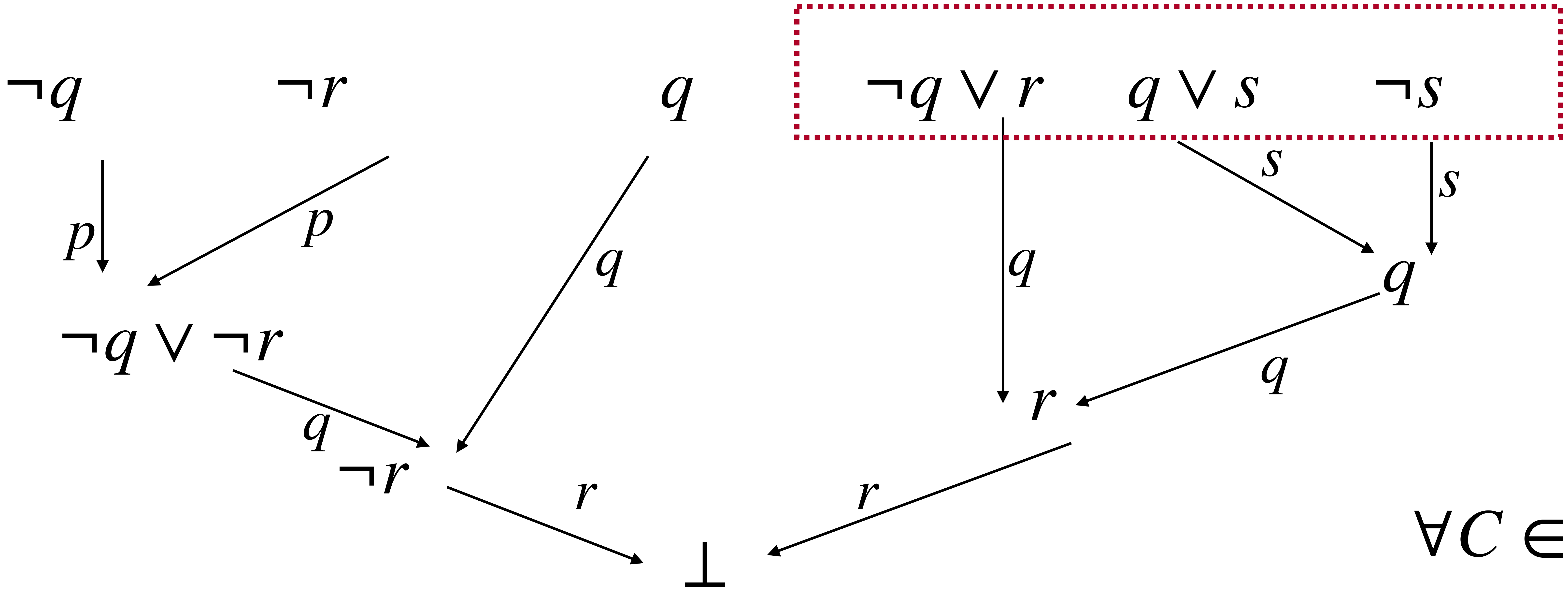
$$\forall C \in Clauses(A), C_{\downarrow(Vars(B))}$$

$$A \rightarrow I$$

How to Compute Interpolants!

$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$

$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$



$\forall C \in Clauses(B), True$

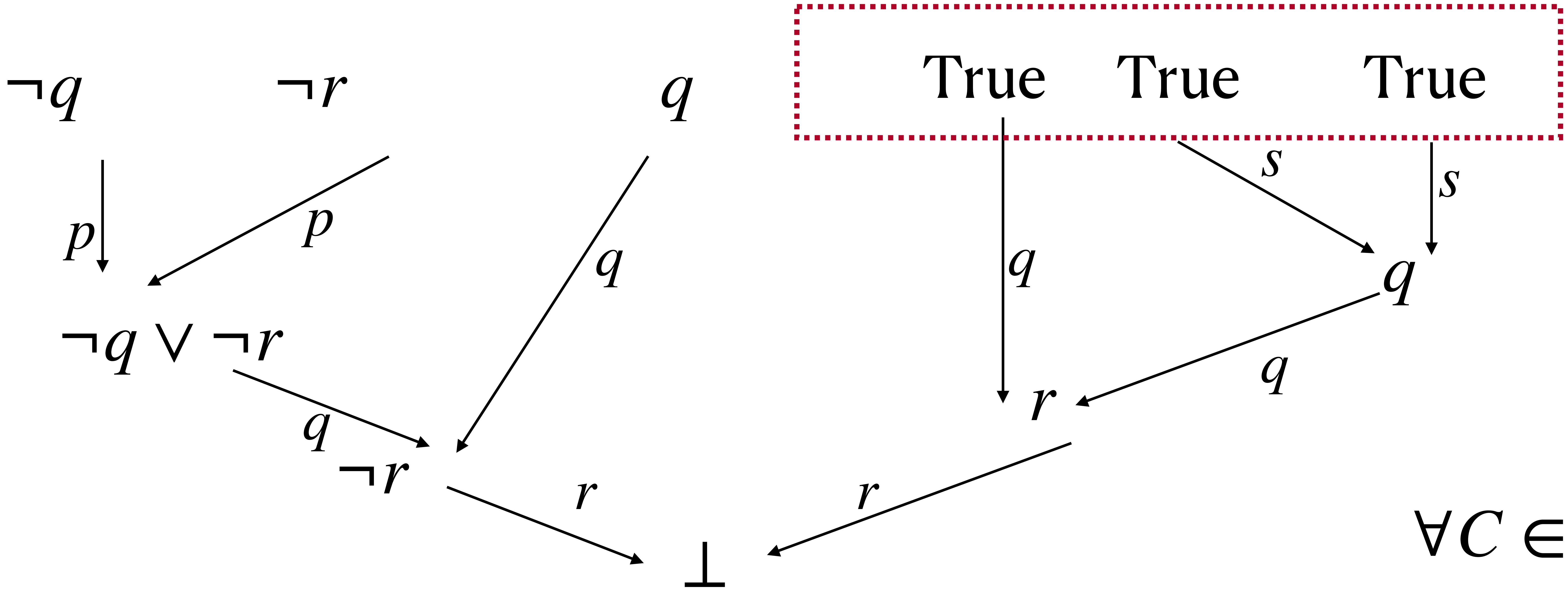
$Clauses(B)$ doesn't contribute to I

$I \wedge B \models \perp$ will be taken care by internal nodes.

How to Compute Interpolants!

$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$

$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$



$\forall C \in Clauses(B), True$

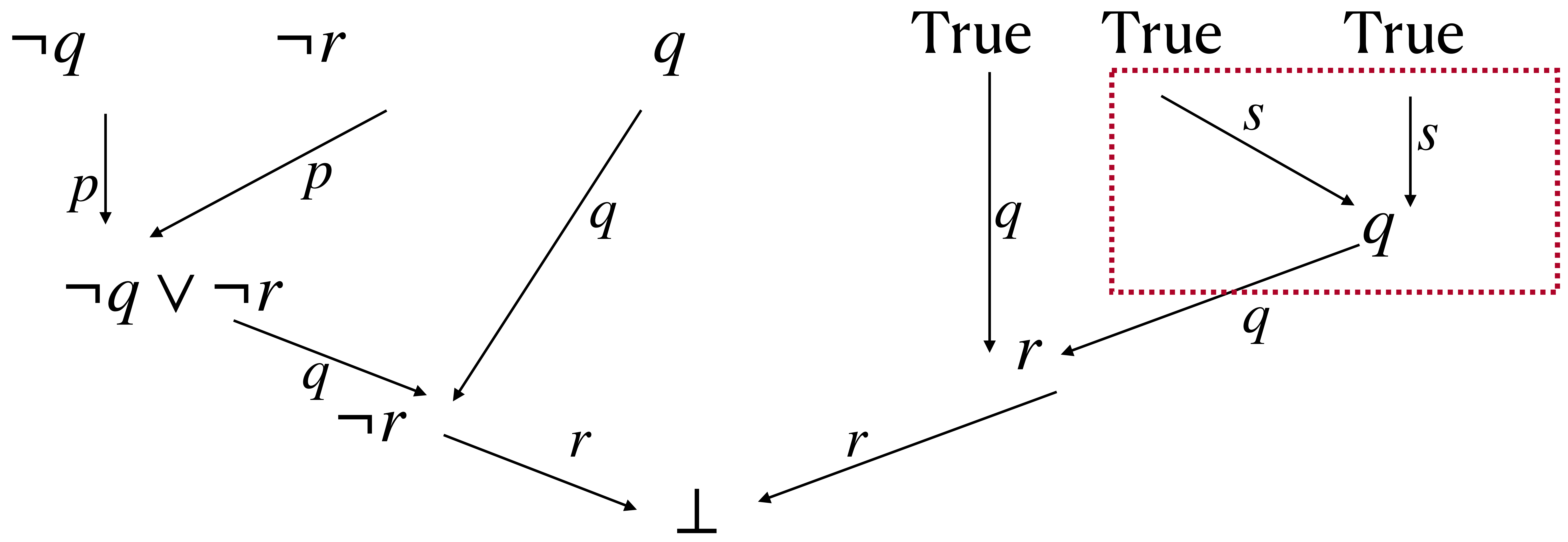
$Clauses(B)$ doesn't contribute to I

$I \wedge B \models \perp$ will be taken care by internal nodes.

How to Compute Interpolants!

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



When pivot variable is B

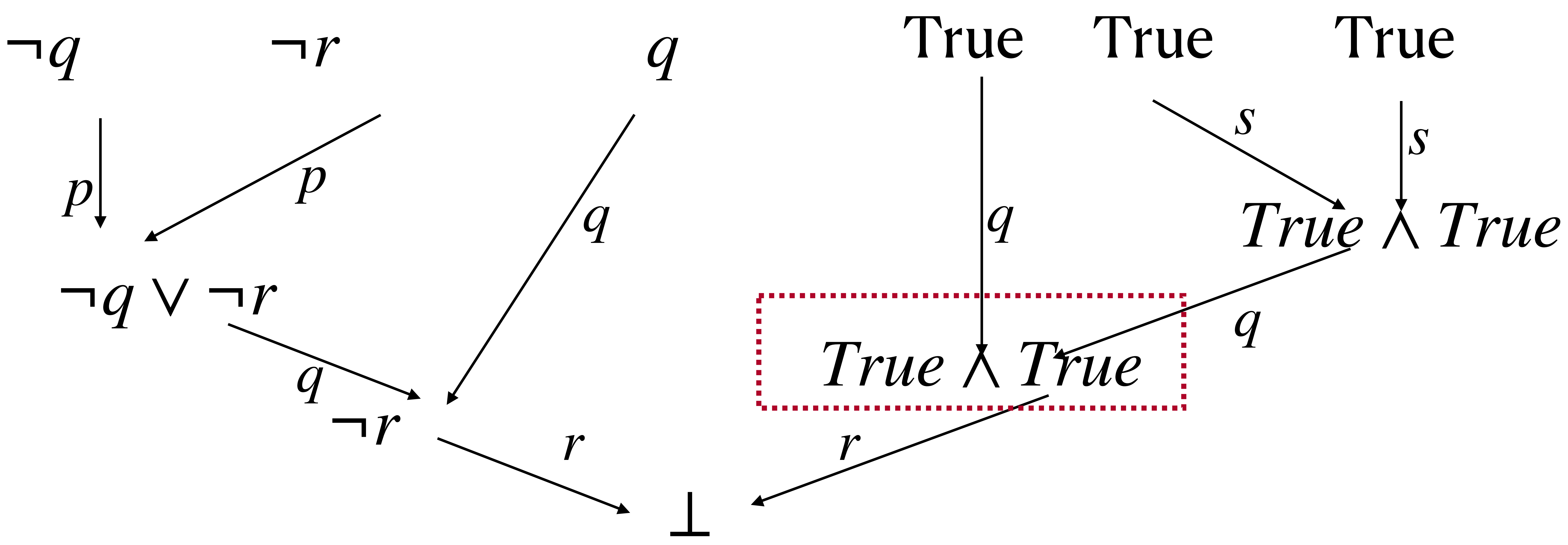
Internal nodes will be “AND” of its both source nodes

To preserve the contradiction with B.
“both source should be considered.

How to Compute Interpolants!

$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$

$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$



When pivot variable is B

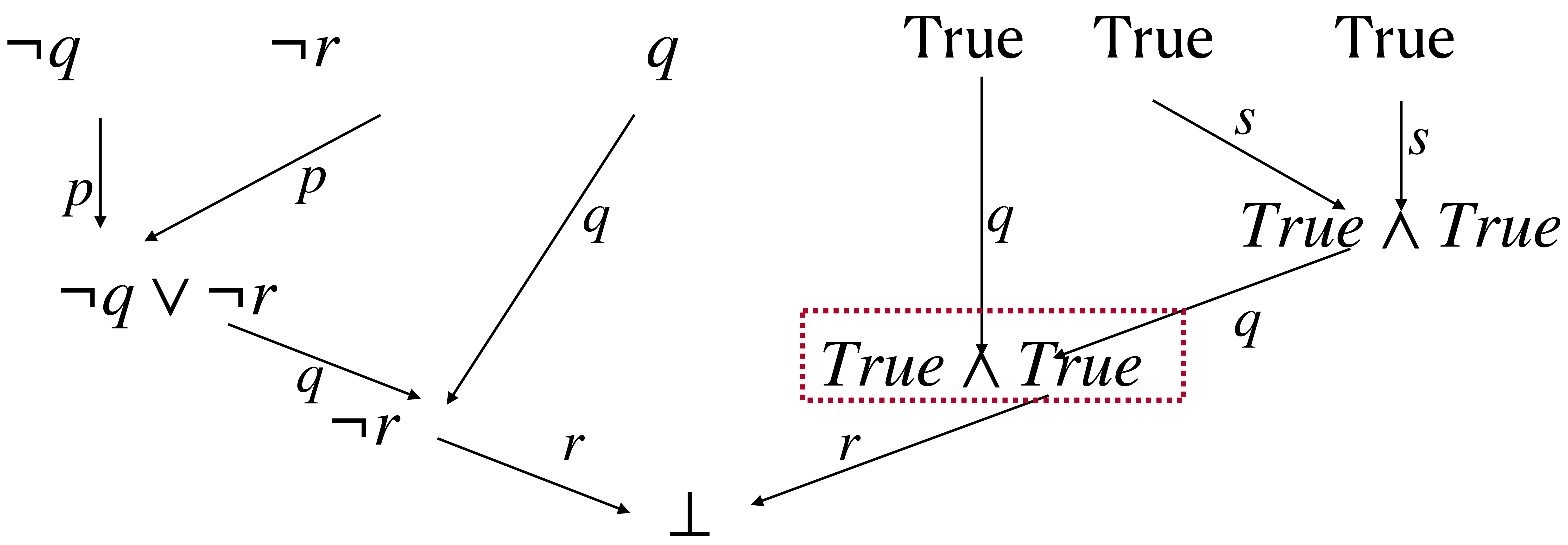
Internal nodes will be "AND" of its both source nodes

To preserve the contradiction with B.
"both source should be considered.

How to Compute Interpolants!

$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$

$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$



When pivot variable is B

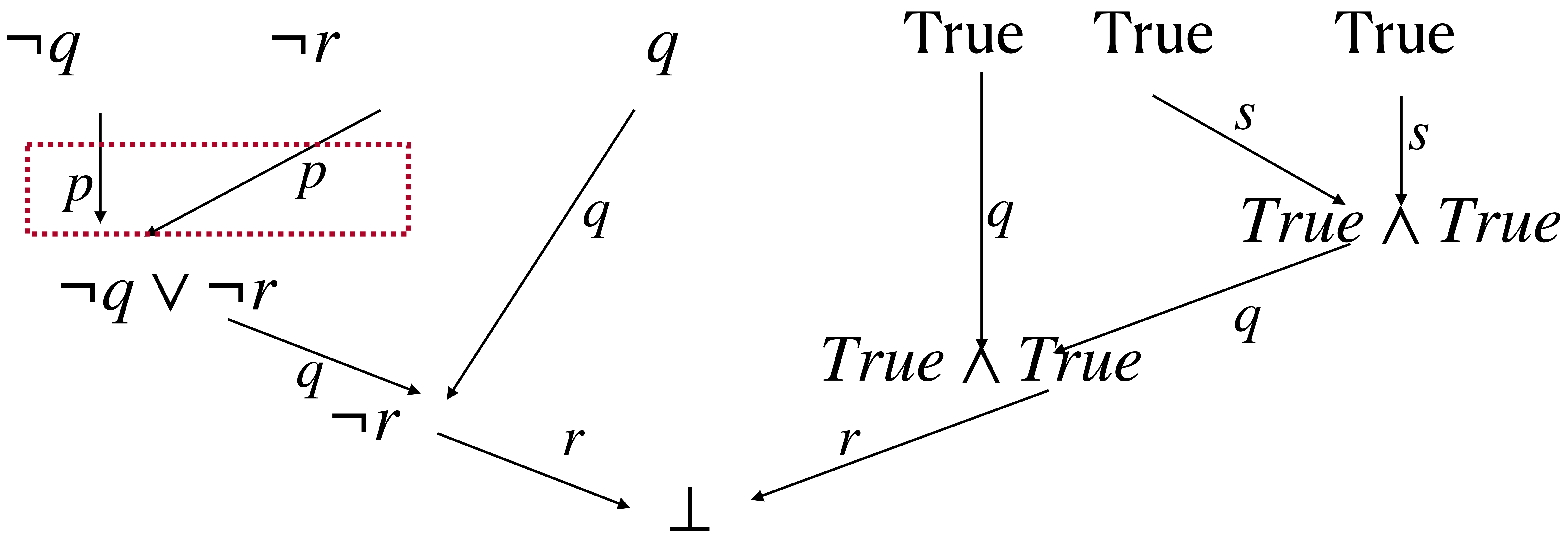
Internal nodes will be "AND" of its both source nodes

To preserve the contradiction with B. "both" source should be considered.

How to Compute Interpolants!

$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$

$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$



When pivot variable is A

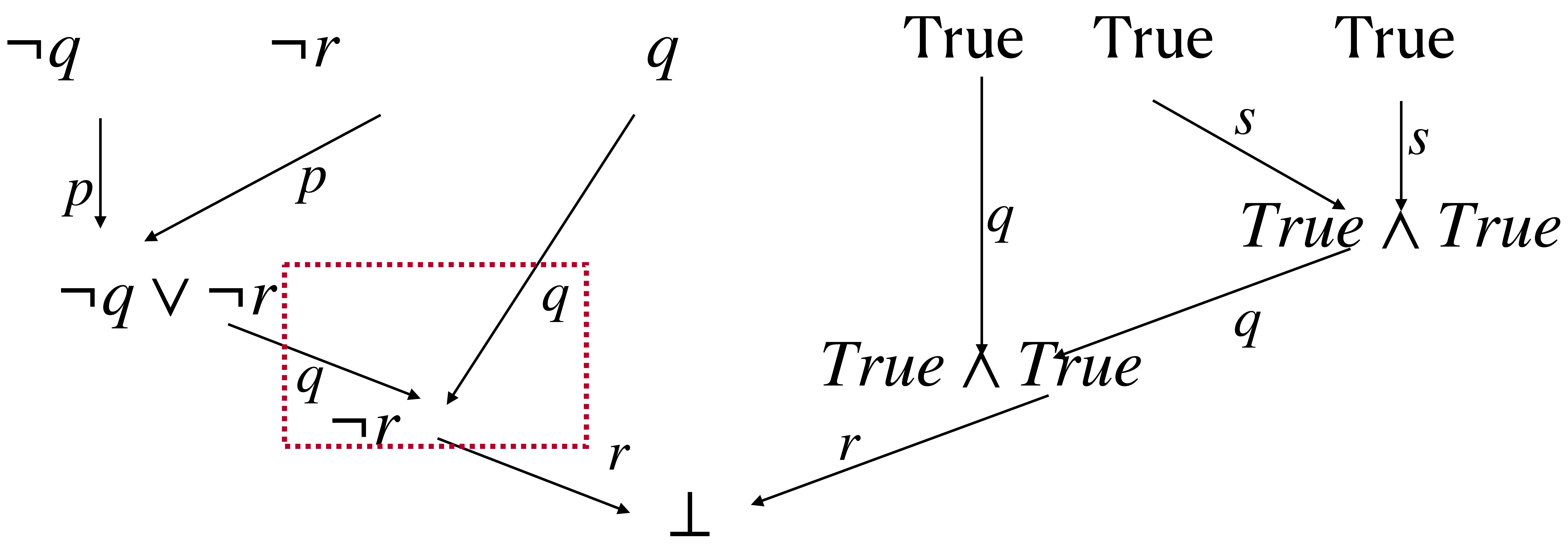
Internal nodes will be "OR" of its both source nodes

To preserve the implication.
Node implies either left clause or right clause

How to Compute Interpolants!

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



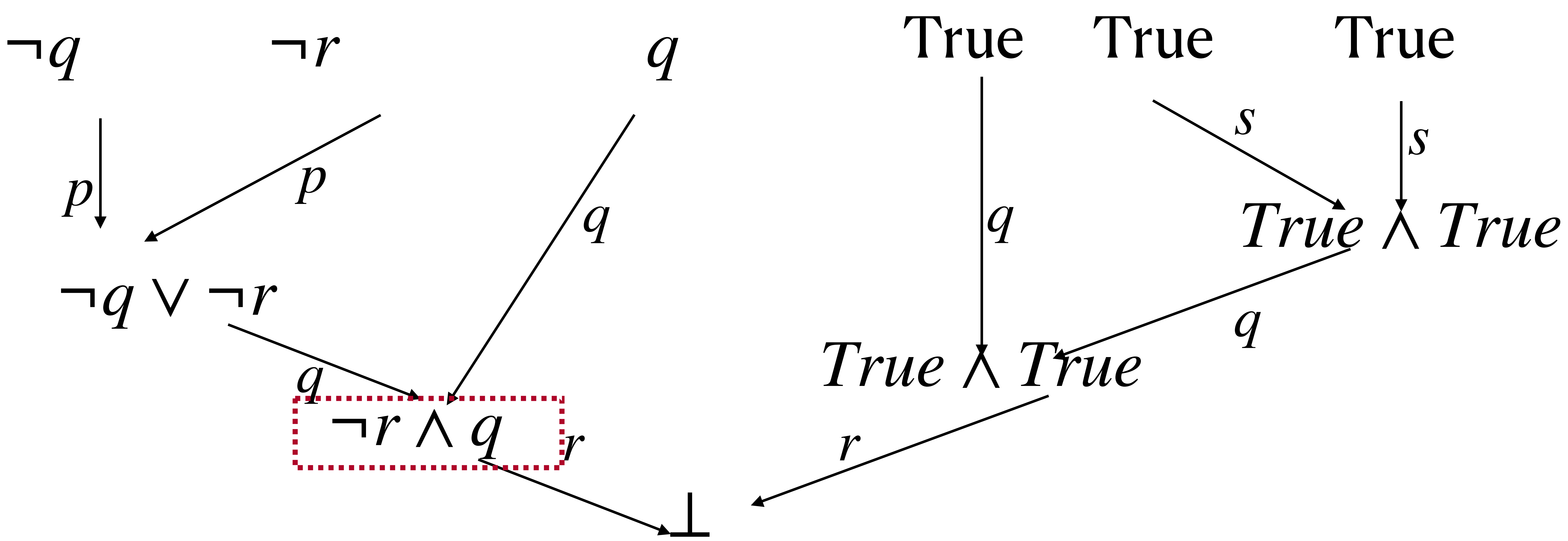
$q \in Vars(B)$

To preserve the contradiction with B.
“both” source should be considered.

How to Compute Interpolants!

$A = (p \lor \neg q) \land (\neg p \lor \neg r) \land q \qquad B = (\neg q \lor r) \land (q \lor s) \land \neg s$

$A \land B = (p \lor \neg q) \land (\neg p \lor \neg r) \land q \land (\neg q \lor r) \land (q \lor s) \land \neg s$



$q \in Vars(B)$

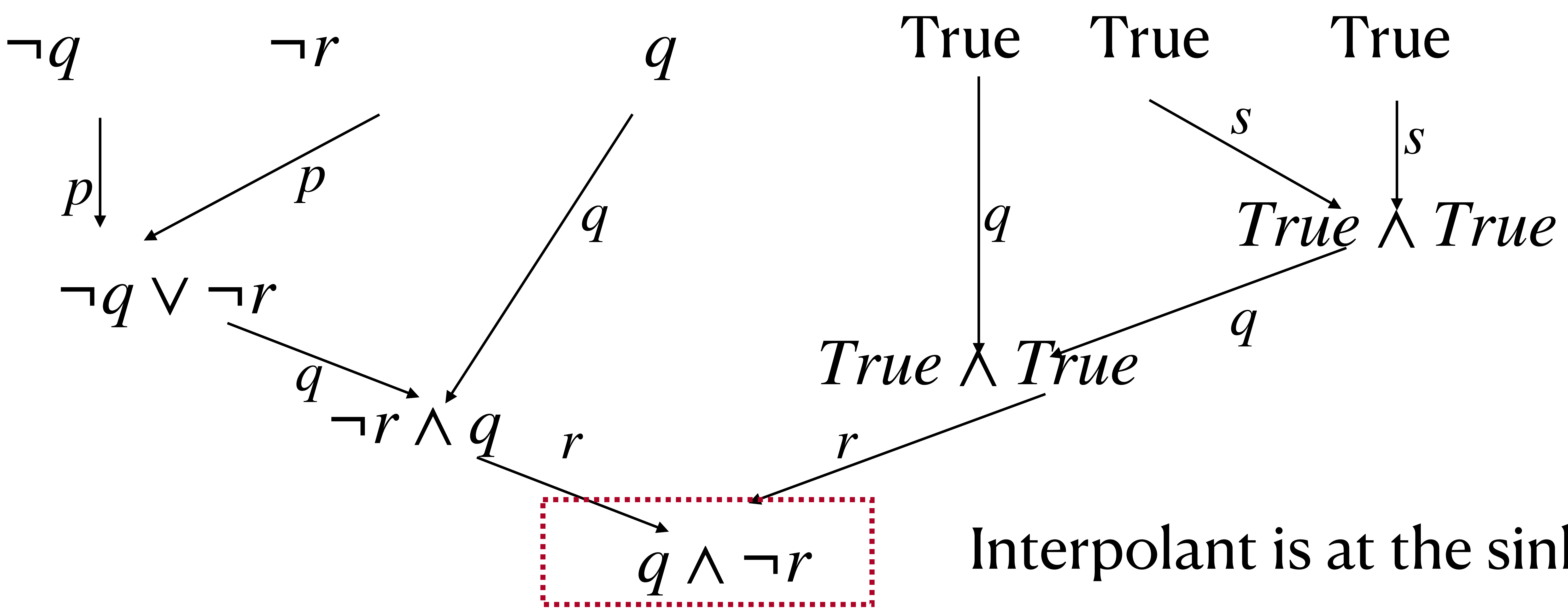
To preserve the contradiction with B.
“both” source should be considered.

How to Compute Interpolants!

$$A = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \qquad B = (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$

$I = q \wedge \neg r$

$$A \wedge B = (p \vee \neg q) \wedge (\neg p \vee \neg r) \wedge q \wedge (\neg q \vee r) \wedge (q \vee s) \wedge \neg s$$



Interpolant is at the sink node

$r \in Vars(B)$

To preserve the contradiction with B.
“both” source should be considered.

How to Compute Interpolants!

McMillan interpolation algorithm (2003)

All the initial nodes have in-degree 0. All internal nodes have in-degree 2. Sink nodes has out-degree 0.

Internal node v, with edges (v_1, v), (v_2, v) implies that v is a resolvent of v_1, v_2

1. Compute Resolution Proof of A and B

2. Base case (input clauses):

If $C \in Clauses(A)$:

$$I_c = C_{\downarrow(Vars(B))})$$

If $C \in Clauses(B)$:

$$I_c = True$$

3. Resolution step: C is Derived by C_1 and C_2 over pivot x.

If $x \in Vars(B)$:

$$I_C = I_{C_1} \wedge I_{C_2}$$

If $x \in Vars(A)$:

$$I_c = I_{c_1} \vee I_{c_2}$$

Interpolant of A,B is $I_\perp$

Compute Interpolants

$$A = (p \vee q) \wedge (\neg p \vee r) \qquad B = \neg q \wedge \neg r$$

Compute Interpolants

$$A = (p \vee q) \wedge (\neg p \vee r) \qquad B = \neg q \wedge \neg r$$