IN3070/4070 – Logic – Autumn 2020

Lecture 7: Resolution

Martin Giese

1st October 2019

Department of Informatics

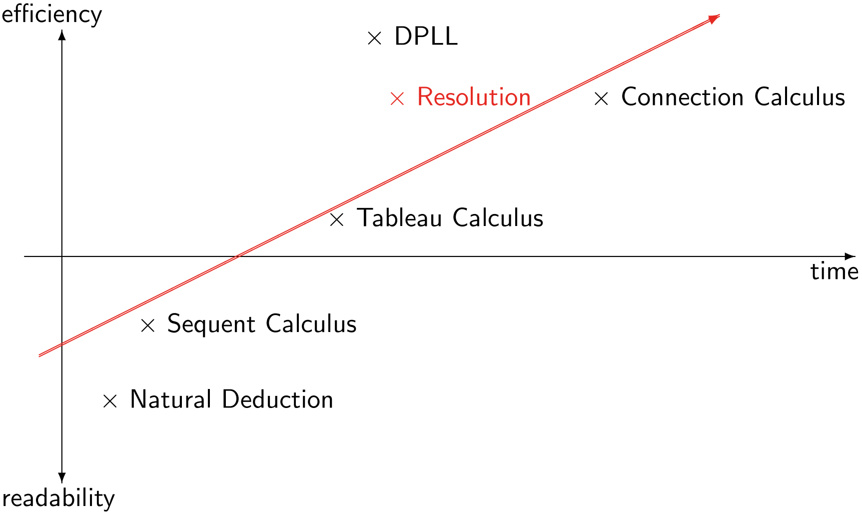
University of Oslo

Today's Plan

- Introduction
- Repetition: Negation Normal Form
- Conjunctive Normal Form
- Clausal Form
- Resolution
- Soundness of Resolution
- Completeness of Resolution

Outline

Proof Search Calculi



Robinson's Resolution Calculus

"A formulation of first-order logic which is specifically designed for use as the basis theoretical instrument of a computer theorem-proving program."

- the resolution calculus was published by Alan Robinson in 1965
- works for first-order formulae in clausal form (e.g. conjunctive or disjunctive normal form)
- consists of one (two for first-order) inference rules and one axiom
- is one of the most popular proof search calculi
- has been implemented in many automated theorem provers

Outline

Negation Normal Form

Definition 2.1 (Negation Normal Form).

A formula is in negation normal form (NNF) if it contains no implications, and all negations are in front of literals.

Example.

- $p \rightarrow q$ *is not in NNF*
- $\neg p \vee q$ *is in NNF*
- $\neg(p \vee \forall x \neg q(x))$ *is not in NNF*
- $\neg p \wedge \exists x\, q(x)$ *is in NNF*

Theorem 2.1.

Every formula in first-order logic can be transformed into an equivalent formula in NNF.

Proof.

To convert an arbitrary formula to a formula in NNF, remove implications, and push negations inwards, preserving equivalence, using the following:

$$A \to B \equiv \neg A \vee B$$
$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$
$$\neg(\forall x\, A) \equiv \exists x\, \neg A$$
$$\neg(\exists x\, A) \equiv \forall x\, \neg A$$
$$\neg(\neg A) \equiv A$$

□

Outline

Conjunctive Normal Form

Definition 3.1 (Conjunctive Normal Form).

A formula is in conjunctive normal form (CNF) if it is a conjunction of disjunctions of literals.

Example.

$(p \vee \neg q) \wedge (\neg p \vee q)$ *is in CNF.*

$(p \vee \neg q) \wedge (\neg p \vee (q \wedge q))$ *is not in CNF.*

What about just p *or* $(p \vee q)$*? Yes, if we consider a literal to be both a conjunction and a disjunction.*

Theorem 3.1.

Every formula in propositional logic can be transformed into an equivalent formula in CNF.

Proof.

To convert an arbitrary propositional formula to a formula in CNF perform the following steps, each of which preserves logical equivalence:

(1) Convert to negation normal form.

(2) Use the distributive laws to move conjunctions inside disjunctions to the outside

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

□

Outline

Clausal Form

Definition 4.1 (Clausal Form).

A clause is a set of literals. A clause is considered to be an implicit disjunction of its literals. A unit clause is a clause consisting of exactly one literal. The empty set of literals is the empty clause, denoted by $\square$. *A formula in clausal form is a set of clauses. A formula is considered to be an implicit conjunction of its clauses. The formula that is the empty set of clauses is denoted by* $\emptyset$.

The only significant difference between clausal form and the standard syntax is that clausal form is defined in terms of sets.

$(p \vee \neg q) \wedge (\neg p \vee q)$ in clausal form: $\{\{p, \neg q\}, \{\neg p, q\}\}$

Transformation to Clausal Form

Corollary 4.1.

Every formula ϕ in propositional logic can be transformed into an logically equivalent formula in clausal form.

Proof.

This follows from the previous theorem, where we transformed a formula to CNF. Each disjunction is then transformed to a clause (of literals), and the clausal form is the set of these clauses. □

Empty Clause and Empty Set of Clauses

Lemma 4.1.

$\Box$, *the empty clause, is unsatisfiable.*

$\emptyset$, *the empty set of clauses, is valid.*

Proof.

A clause is satisfiable iff there is some interpretation under which at least one literal in the clause is true. Let $\mathcal{I}$ be an arbitrary interpretation. Since there are no literals in $\Box$, there are no literals whose value is true under $\mathcal{I}$. But $\mathcal{I}$ was an arbitrary interpretation, so $\Box$ is unsatisfiable.

A set of clauses is valid iff every clause in the set is true in every interpretation. But there are no clauses in $\emptyset$ that need be true, so $\emptyset$ is valid. $\Box$

Outline

The Resolution Rule

The resolution calculus is a refutation procedure.

- in order to determine whether a formula F (in clausal form) is valid, we check whether $\neg F$ is unsatisfiable

Definition 5.1 (Complementary Literal).

The complementary literal $\overline{L}$ of a literal L is A if L is of the form $\neg A$, otherwise it is $\neg L$.

Definition 5.2 (Resolution Rule).

Let C_1, C_2 be clauses with $L \in C_1$ and $\overline{L} \in C_2$. The resolvent C' of C_1 and C_2 is $(C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$. C_1 and C_2 are the parents of C'.

- the resolution rule maintains satisfiability: If $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$ then $\mathcal{I} \models C'$
- if a set of clauses S is satisfiable and $C_1, C_2 \in S$, then $S \cup \{C'\}$ is satisfiable.

The Resolution Rule – Example

Example: Let $C_1 = \{a, b, \neg c\}$ and $C_2 = \{b, c, \neg e\}$.

$$\{a, b, \neg c\} \searrow \quad \swarrow \{b, c, \neg e\}$$
$$\{a, b, \neg e\}$$

The resolvent of C_1 and C_2 is $\{a, b, \neg e\}$.

Observations:

- if $\{a, b, \neg c\}$ and $\{b, c, \neg e\} \equiv (a \vee b \vee \neg c) \wedge (b \vee c \vee \neg e)$ are true in $\mathcal{I}$, then $(a \vee b)$ is true (if c is true) or $(b \vee \neg e)$ is true (if c is false); hence $(a \vee b \vee \neg e)$ is true
- if resolvent is unsatisfiable, then conj. of parents is unsatisfiable
- the empty clause $\Box$ is unsatisfiable
- goal: derive empty clause $\Box$

The Resolution Calculus

- a set of clauses is unsatisfiable iff the empty clause can be derived
- a clause C is true iff at least one of its literals is true; if there is no literal in C, then C is false and every set of clauses (in CNF) that contains C is false, i.e.unsatisfiable

Definition 5.3 (Resolution Procedure).

Given a set of clauses S.

1. *apply the resolution rule to a pair of clauses $\{C_1, C_2\} \subseteq S$ that has not been chosen before; let C' be the resolvent*
2. $S' := S \cup \{C'\}$, $S := S'$
3. *if $C' = \Box$, then output "unsatisfiable"; if all possible resolvents have been considered, then output "satisfiable"; otherwise continue with 1.*

Resolution Calculus – Example 1

- Prove validity of: $(p \wedge q) \rightarrow p$
- Show unsatisfiability of: $\neg((p \wedge q) \rightarrow p)$
- CNF: $p \wedge q \wedge \neg p$
- Clause set: $\{\{p\}, \{q\}, \{\neg p\}\}$
- Resolve $\{p\}$ with $\{\neg p\}$
- Resolvent: $\square$

Resolution Calculus – Example 2

- Prove validity of: $p \wedge (p \rightarrow q) \rightarrow q$
- Show unsatisfiability of: $\neg(p \wedge (p \rightarrow q) \rightarrow q)$
- Equivalent to: $p \wedge (p \rightarrow q) \wedge \neg q$
- CNF: $p \wedge (\neg p \vee q) \wedge \neg q$
- Clause set: $\{\{p\}, \{\neg p, q\}, \{\neg q\}\}$
- Resolution step 1: between $\{p\}$ and $\{\neg p, q\}$
- Resolvent: $\{q\}$
- New clause set: $\{\{p\}, \{\neg p, q\}, \{\neg q\}, \{q\}\}$
- Resolution step 2: between $\{\neg q\}$ and $\{q\}$
- Resolvent: $\square$

Resolution Calculus – Example 3

- Prove validity of: $(p \to (q \to r)) \to (p \wedge q \to r)$
- Show unsatisfiability of: $\neg((p \to (q \to r)) \to (p \wedge q \to r))$
- Equivalent to $(p \to (q \to r)) \wedge (p \wedge q) \wedge \neg r$
- Clauses:

1. $\{\neg p, \neg q, r\}$
2. $\{p\}$
3. $\{q\}$
4. $\{\neg r\}$
5. $\{\neg q, r\}$ — resolvent of 1. and 2.
6. $\{r\}$ — resolvent of 3. and 5.
7. $\Box$ — resolvent of 4. and 6.

Resolution Calculus – Example 4

- Prove validity of: $(p \rightarrow q) \rightarrow ((p \rightarrow r) \rightarrow (p \rightarrow (q \wedge r)))$
- Clauses:

1. $\{\neg p, q\}$
2. $\{\neg p, r\}$
3. $\{p\}$
4. $\{\neg q, \neg r\}$
5. $\{q\}$ — resolvent of 1. and 3.
6. $\{r\}$ — resolvent of 2. and 3.
7. $\{\neg r\}$ — resolvent of 4. and 5.
8. $\Box$ — resolvent of 6. and 7.

- May have to use same clause several times
- Order of resolution steps does not matter for completeness

The Formal Resolution Calculus

Definition 5.4 (Resolution Calculus).

The resolution calculus has one axiom and one (inference) rule.

$$\frac{}{C_1, \ldots, \Box, \ldots C_n}\ \textit{axiom}$$

$$\frac{C_1, \ldots, C_i \cup \{L\}, \ldots, C_j \cup \{\overline{L}\}, \ldots, C_n, C_i \cup C_j}{C_1, \ldots, C_i \cup \{L\}, \ldots, C_j \cup \{\overline{L}\}, \ldots, C_n}\ \textit{resolution}$$

A resolution proof of a set of clauses S is a derivation of S in the resolution calculus.

- in contrast to natural deduction or the sequent calculus, the resolution calculus has no rule with more than one premise
- hence, a derivation in the resolution calculus has only one branch
- terminates, if all clauses $C_i \cup \{L\}, C_j \cup \{\overline{L}\}$ have been considered

Outline

Soundness of Resolution

- Recall: to prove A, we 'refute' $\neg A$
- I.e. we derive a 'contradiction' (the empty clause) from $\neg A$...
- ...meaning that $\neg A$ was unsatisfiable, and therefore A valid.

We need to prove the following statements:

1. If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.
2. A set of clauses containing the empty clause is unsatisfiable

Resolution Preserves Satisfiability

Lemma 6.1.

If a set of clauses S is satisfiable, then the result of adding the resolvent of two clauses $C_1, C_2 \in A$ to S is also satisfiable.

Proof.

Let S be a set of clauses, and $C_1, C_2 \in S$ with $L \in C_1$ and $\overline{L} \in C_2$. Let $\mathcal{I}$ be an interpretation with $\mathcal{I} \models S$.

A clause set is a *conjunction* of its clauses, so $\mathcal{I} \models C_1$ and $\mathcal{I} \models C_2$.

Now either $\mathcal{I} \models L$ or $\mathcal{I} \models \overline{L}$:

- $\mathcal{I} \models L$ $\mathcal{I} \models C_2$, and clauses are *disjunctions* of their literals, so $\mathcal{I}$ satisfies one of the literals in C_2, but not $\overline{L}$. So: $\mathcal{I} \models C_2 \setminus \{\overline{L}\}$.
- $\mathcal{I} \models \overline{L}$ By the same reasoning $\mathcal{I} \models C_1 \setminus \{L\}$.

So $\mathcal{I}$ satisfies at least one literal in either $C_1 \setminus \{L\}$ or $C_2 \setminus \{\overline{L}\}$.

I.e. $\mathcal{I} \models (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$, the resolvent of C_1 and C_2. □

The Empty Clause is unsatisfiable

Lemma 6.2.

A set of clauses containing the empty clause is unsatisfiable.

Proof.

Let S be a set of clauses and $\square \in S$.
Assume for the sake of contradiction that $\mathcal{I} \models S$.
A clause set is a *conjunction* of its clauses, so in particular $\mathcal{I} \models \square$.
Since clauses are *disjunctions*, to satisfy a clause C, an interpretation has to satisfy *at least one* of its literals $L \in C$.
But the empty clause $\mathcal{I}$ contains no literals, so that is a contradiction.

□

Outline

Prove Completeness like for LK?

- Plan:
  - Starting from a set of clauses S...
  - ...build a fair limit derivation where all resolutions are applied...
  - ...giving a set of clauses S' with $\Box \notin S'$.
  - Define an interpretation $\mathcal{I}_{S'}$ based on the "smallest" clauses (literals)
  - Show by structural induction that $\mathcal{I}_{S'}$ satsifies all clauses in S'
  - So in particular the ones in S.
- Nice plan, but unfortunately...
  - Resolution does not make clauses smaller (resolvent can be larger!)
  - So we don't always get lots of one-literal clauses in $\mathcal{I}_S$
  - And we can't use structural induction either
- This can be fixed
  - $\mathcal{I}_{S'}$ is not defined on only the one-literal clauses
  - Argument doesn't use structural induction on clauses
  - The proof is rather advanced!
- We will go through Robinson's original proof

Semantic Trees

The completeness proof uses the following concept:

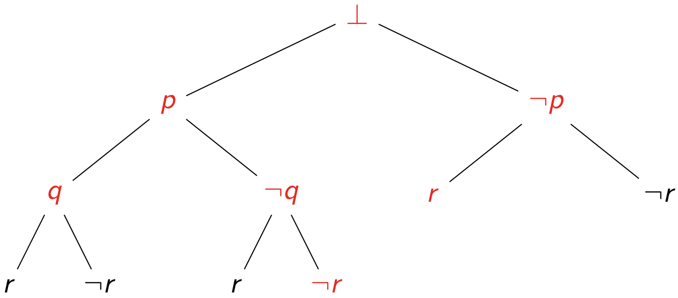
Definition 7.1 (Semantic Trees).

A semantic tree is a binary tree where:

- *The root is labelled by the symbol* $\bot$*,*
- *Every node has either no children or two children,*
- *For every node that has children, there is some atom* A *such that one child is labeled with* A *and the other with* $\neg A$
- *There are not two complementary literals* A *and* $\neg A$ *on any path starting at the root.*

Not a data structure, just needed for the completeness proof

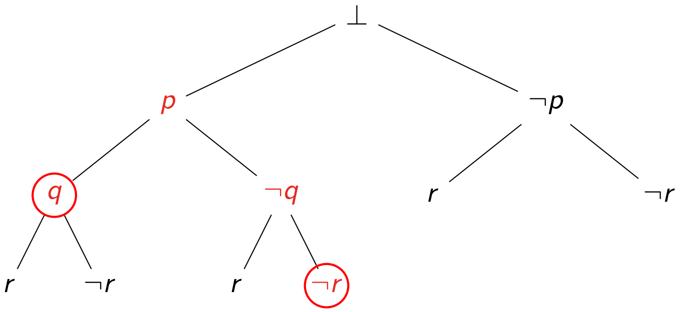
Semantic Trees — Example



- Root labelled with ⊥
- Either two children, or no children
- Complementary siblings
- No complementary pairs on a path

Partial Interpretations

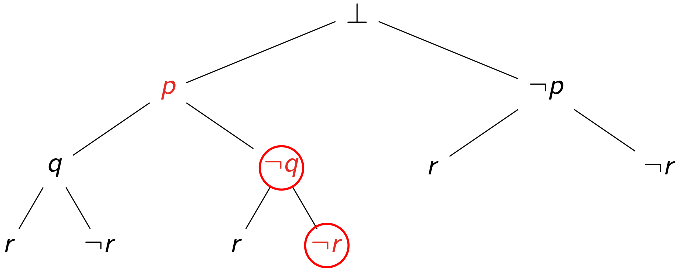
The path to every node n in a semantic tree gives a 'partial interpretation' $\mathcal{I}_n$:



$\mathcal{I}_n \models p$, $\mathcal{I}_n \models \neg q$, $\mathcal{I}_n \models \neg r$ $\mathcal{I}_n \models p$, $\mathcal{I}_n \models q$

Failure Nodes – Motivation

Sometimes, such a 'partial interpretation' is enough to falsify a clause:



- At the marked node, the clause $\neg p \vee q \vee r$ is false
- At the marked node, the clause $\neg p \vee r$ is false
- At the marked node, the clause $\neg p \vee q$ is false
- The clause $\neg p \vee q$ is already false at the parent node!
- It remains false further down.

Failure Nodes – Definition

Definition 7.2.

A node n in a semantic tree is a falsifies a clause C if for every literal $L \in C$, the complement $\overline{L}$ is on the branch leading to n.

Definition 7.3.

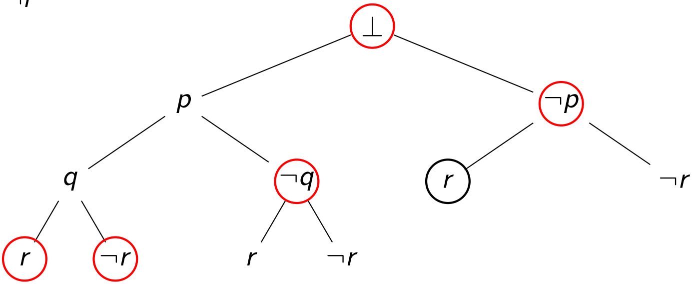
A node n in a semantic tree is a failure node for a clause set S if it falsifies some clause $C \in S$, but the parent of n does not.

Failure nodes have just enough information to make sure some clause is falsified.

Note: A has the root as a failure node iff $\Box \in S$.

Failure Nodes – Example

1. $\neg p \vee \neg q \vee \neg r$
2. $\neg q \vee r$
3. $\neg p \vee q$
4. p
5. $p \vee \neg r$
6. $\Box$



Not a failure node: parent node falsifies clause 4. The empty clause is falsified by the root node

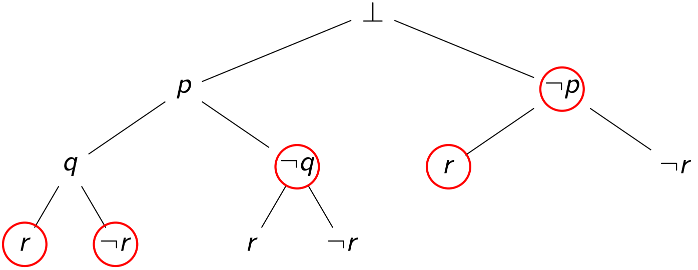
Closed Semantic Trees

Definition 7.4.

Given a semantic tree and a clause set S, a branch of the tree is closed if it contains a failure node.
The semantic tree is closed if all branches contain failure nodes.

Closed Semantic Tree – Example

1. $\neg p \vee \neg q \vee \neg r$
2. $\neg q \vee r$
3. $\neg p \vee q$
4. p
5. $p \vee \neg r$



The semantic tree is closed for these 5 clauses. Without p, it is not closed.

Complete Semantic Trees

Definition 7.5.

A semantic tree is complete if for every atomic formula A and every branch (from root to leaf) either A or $\neg A$ occurs

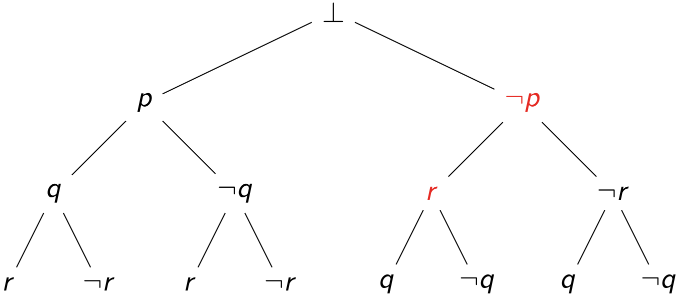
Every branch $\mathcal{B}$ in a *complete* semantic tree corresponds to an interpretation $\mathcal{I}_{\mathcal{B}}$ with $\mathcal{I} \models A$ iff A is on the branch.

Lemma 7.1.

For every interpretation $\mathcal{I}$ there is a branch $\mathcal{B}$ in a complete semantic tree with $\mathcal{I} = \mathcal{I}_{\mathcal{B}}$.

A complete semantic tree 'enumerates' all possible interpretations.

Example: Complete Semantic Tree



Not complete, since neither q nor $\neg q$ on branch Complete for vocabulary $\{p, q, r\}$

Complete Semantic Trees

Definition 7.5.

A semantic tree is complete if for every atomic formula A and every branch (from root to leaf) either A or $\neg A$ occurs

Every branch $\mathcal{B}$ in a *complete* semantic tree corresponds to an interpretation $\mathcal{I}_{\mathcal{B}}$ with $\mathcal{I} \models A$ iff A is on the branch.

Lemma 7.1.

For every interpretation $\mathcal{I}$ there is a branch $\mathcal{B}$ in a complete semantic tree with $\mathcal{I} = \mathcal{I}_{\mathcal{B}}$.

A complete semantic tree 'enumerates' all possible interpretations.

Unsatisfiable Clause Sets close Semantic Trees

Theorem 7.1.

A clause set is unsatisfiable iff there is a closed semantic tree for it.

Proof.

$\Rightarrow$ Let S be an unsatisfiable clause set. Construct a complete semantic tree. For each branch $\mathcal{B}$, $\mathcal{I}_{\mathcal{B}} \not\models S$, so $\mathcal{I}_{\mathcal{B}} \not\models C$ for some clause $C \in S$, so there is a node on the branch that falsifies C.
The falsifying nodes highest up on each branch are failure nodes. So the semantic tree is closed.

$\Leftarrow$ Let S be a clause set and let a closed semantic tree be given. For any interpretation $\mathcal{I}$, there is a branch in the tree such that $\mathcal{I} \models L$ for all literals L on that branch. Since there is a failure node for some clause $C \in S$ on that branch, the atoms on the branch entail $\neg C$, so $\mathcal{I} \not\models C$, and thus $\mathcal{I} \not\models S$.
This holds for arbitrary interpretations $\mathcal{I}$, so S is unsatisfiable. □

Resolution Steps from Closed Semantic Trees

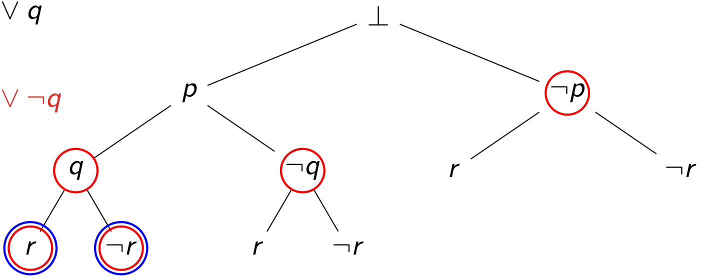
Lemma 7.2.

Let S be an unsatisfiable clause set, with a closed semantic tree, and $\Box \notin S$. Then

- *a resolution step is possible from S,*
- *and the resulting clause set S' has a smaller closed semantic tree*

Idea of proof

1. $\neg p \lor \neg q \lor \neg r$
2. $\neg q \lor r$
3. $\neg p \lor q$
4. p
5. $\neg p \lor \neg q$



- There are two sibling failure nodes
- They falsify two clauses with complementary literals
- They can be resolved to a new clause $\neg p \lor \neg q$
- Which is falsified by the parent node

There are two sibling failure nodes

- Let n_0 be the root.
- Since $\Box \notin S$, n_0 is not a failure node.
- n_0 has two children.
- If both are failure nodes, we are done.
- Otherwise, let n_1 be one of the siblings that is not a failure node.
- n_1 has two children.
- If both are failure nodes, we are done.
- ...
- This either finds sibling failure nodes...
- or it constructs a path in the tree without a failure node, but that is not possible.

Sibling Failure Nodes give Resolution Opportunities

- Let n_1 and n_2 be sibling failure nodes
  - falsifying C_1 and C_2,
  - labeled A and $\neg A$.
- The parent node n of n_1 and n_2 does not falsify C_1 and C_2.
- Let N be the set of literals on the nodes up to and including n.
- Every literal in C_1 has its negation in $N \cup \{A\}$
- But *not every* literal in C_1 has its negation in N
- Therefore $\neg A \in C_1$
- Similarly $A \in C_2$
- C_1 and C_2 can be resolved to $C := (C_1 \setminus \{\neg A\}) \cup (C_2 \setminus \{A\})$
- Every literal in C has its negation in N
- Adding C to the clause set will make n into a failure node.
- This gives a closed semantic tree with two nodes less than before.

Completeness of Resolution

Theorem 7.2.

If S is an unsatisfiable clause set, then there is a resolution derivation of the empty clause from S.

Proof.

- There exists a closed semantic tree for S
- As long as S does not contain the empty clause,
  - It is possible to apply a resolution step to S
  - Leading to a clause set with a smaller closed semantic tree
- Since the tree is finite, this cannot go on forever.
- Therefore, eventually the semantic tree must consist of only the root. . .
- . . . and S contain the empty clause $\square$.

$\square$