CS228 : Logic for Computer Science

Module 2: The Problem of Satisfiability

Lecture 8: (Propositional) Resolution

Instructor: S. Akshay

IIT Bombay, India

Topics Covered till date (Module 1: Propositional Logic)

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity

Topics Covered till date (Module 1: Propositional Logic)

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness

Topics Covered till date (Module 1: Propositional Logic)

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Topics Covered till date (Module 1: Propositional Logic)

1. Syntax and Semantics
   - Questions of interest: Satisfiability, Validity
2. Formal proof systems
   - Natural Deduction
   - Soundness and Completeness
3. Normal Forms
   - Negation, Conjunctive and Disjunctive Normal Forms (NNF, CNF, DNF)
   - Tseitin's Encoding

Next... (Module 2: The Problem of Satisfiability)

Resolution and SAT-solvers

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

- If we could solve SAT efficiently, we could solve every problem in NP efficiently too!

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

- If we could solve SAT efficiently, we could solve every problem in NP efficiently too!
- Includes: scheduling, planning, circuit verification, cryptanalysis, puzzle-solving , . . . thousands of problems people care about, in industry and research.

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

- If we could solve SAT efficiently, we could solve every problem in NP efficiently too!
- Includes: scheduling, planning, circuit verification, cryptanalysis, puzzle-solving , . . . thousands of problems people care about, in industry and research.

The Cook–Levin Theorem

SAT is the first known NP-complete problem – discovered independently, twice.

Stephen Cook
Toronto, 1971

Leonid Levin
Moscow, 1973

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

- If we could solve SAT efficiently, we could solve every problem in NP efficiently too!
- Includes: scheduling, planning, circuit verification, cryptanalysis, puzzle-solving , . . . thousands of problems people care about, in industry and research.

The Cook–Levin Theorem

SAT is the first known NP-complete problem – discovered independently, twice.

Stephen Cook
Toronto, 1971

Leonid Levin
Moscow, 1973

So SAT isn't just a hard problem to study – in a precise sense, it is the hard problem.

Why does Satisfiability matter?

Recall from Lecture 7: Checking satisfiability of a CNF formula (called SAT) is NP-complete.

- If we could solve SAT efficiently, we could solve every problem in NP efficiently too!
- Includes: scheduling, planning, circuit verification, cryptanalysis, puzzle-solving , . . . thousands of problems people care about, in industry and research.

The Cook–Levin Theorem

SAT is the first known NP-complete problem – discovered independently, twice.

Stephen Cook
Toronto, 1971

Leonid Levin
Moscow, 1973

So SAT isn't just a hard problem to study – in a precise sense, it is the hard problem.

Coming up: Resolution, and the algorithms powering modern SAT solvers.

Clauses as sets and CNF formulas as set of sets

Recall from Lecture 7

- A clause $\ell_1 \vee \cdots \vee \ell_n$ is written as the set $\{\ell_1, \ldots, \ell_n\}$.
- A CNF formula $C_1 \wedge \cdots \wedge C_n$ as the set of clauses $\{C_1, \ldots, C_n\}$.
- Using commutativity, associativity and idempotence of $\vee, \wedge$.

Clauses as sets and CNF formulas as set of sets

Recall from Lecture 7

- A clause $\ell_1 \vee \cdots \vee \ell_n$ is written as the set $\{\ell_1, \ldots, \ell_n\}$.
- A CNF formula $C_1 \wedge \cdots \wedge C_n$ as the set of clauses $\{C_1, \ldots, C_n\}$.
- Using commutativity, associativity and idempotence of $\vee, \wedge$.

Notation

We extend the definition of clauses to the empty set of literals: $\perp$ is the empty clause.

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?
- Recall that the naive algo requires to examine all lines of truth table.
- If satisfiable we stop at some point, but if unsatisfiable we have explore all assignments!

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?
- Recall that the naive algo requires to examine all lines of truth table.
- If satisfiable we stop at some point, but if unsatisfiable we have explore all assignments!

Can we do better using the CNF structure?

- F is unsatisfiable if...

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?
- Recall that the naive algo requires to examine all lines of truth table.
- If satisfiable we stop at some point, but if unsatisfiable we have explore all assignments!

Can we do better using the CNF structure?

- F is unsatisfiable if... we can derive $\perp$ from this set of clauses.

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?
- Recall that the naive algo requires to examine all lines of truth table.
- If satisfiable we stop at some point, but if unsatisfiable we have explore all assignments!

Can we do better using the CNF structure?

- F is unsatisfiable if... we can derive $\perp$ from this set of clauses.

  E.g., $p \wedge \neg p$ gives clauses $\{p\}$ and $\{\neg p\}$, which combine to give $\perp$.

Satisfiability and unsatisfiability

Question: given a formula F, is it satisfiable?

- As we saw before, if F were in DNF this is easy, but converting to satisfiability-preserving DNF formula is hard!
- On the other hand we know that by Tseitin encoding, we can easily convert to a satisfiability-preserving CNF formula. Does this help?
- Eg., Is $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ satisfiable?
- Recall that the naive algo requires to examine all lines of truth table.
- If satisfiable we stop at some point, but if unsatisfiable we have explore all assignments!

Can we do better using the CNF structure?

- F is unsatisfiable if... we can derive $\perp$ from this set of clauses.

  E.g., $p \wedge \neg p$ gives clauses $\{p\}$ and $\{\neg p\}$, which combine to give $\perp$.
- If F is in CNF, then we can interpret F as a set of clauses. Does this help?

Derivations from CNF formulas

So, Goal is to derive $\perp$ from set of clauses of a CNF formula

Derivations from CNF formulas

So, Goal is to derive $\perp$ from set of clauses of a CNF formula

- how many rules do we need?

Derivations from CNF formulas

So, Goal is to derive $\perp$ from set of clauses of a CNF formula

- how many rules do we need?
- Recall Natural Deduction had 12 primitive rules to try (plus a few handy derived ones)!

Derivations from CNF formulas

So, Goal is to derive $\perp$ from set of clauses of a CNF formula

- how many rules do we need?
- Recall Natural Deduction had 12 primitive rules to try (plus a few handy derived ones)!
- To automate, we want as few choices as possible...

Derivations from CNF formulas

So, Goal is to derive $\perp$ from set of clauses of a CNF formula

- how many rules do we need?
- Recall Natural Deduction had 12 primitive rules to try (plus a few handy derived ones)!
- To automate, we want as few choices as possible...

Answer: We need only one ring to rule them all

Derivations from CNF formulas

So, Goal is to derive $\bot$ from set of clauses of a CNF formula

- how many rules do we need?
- Recall Natural Deduction had 12 primitive rules to try (plus a few handy derived ones)!
- To automate, we want as few choices as possible...

Answer: We need only one ~~ring to~~ rule ~~them all~~

$$\textsc{Resolution} \quad \frac{F \vee G \qquad \neg F \vee H}{G \vee H}$$

Resolution proof rule

In fact, we apply resolution rule on clauses:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

Resolution proof rule

In fact, we apply resolution rule on clauses:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- clauses $p \vee C$ and $\neg p \vee D$ are called antecedents

Resolution proof rule

In fact, we apply resolution rule on clauses:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- clauses $p \vee C$ and $\neg p \vee D$ are called antecedents
- p is a variable called pivot

Resolution proof rule

In fact, we apply resolution rule on clauses:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- clauses $p \vee C$ and $\neg p \vee D$ are called antecedents
- p is a variable called pivot
- $C \vee D$ is a clause too, called resolvent

Resolution proof rule

In fact, we apply resolution rule on clauses:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D}$$

- clauses $p \vee C$ and $\neg p \vee D$ are called antecedents
- p is a variable called pivot
- $C \vee D$ is a clause too, called resolvent

E.g.: I drink coffee, or I fall asleep in class. I do not drink coffee, or I understand the lecture. Therefore, I fall asleep in class, or I understand the lecture.

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$p \vee q \qquad \neg p \vee q$$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\frac{p \vee q \qquad \neg p \vee q}{q}$$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\frac{p \vee q \qquad \neg p \vee q}{q} \qquad\qquad \neg q \vee r$$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\dfrac{\dfrac{p \vee q \qquad \neg p \vee q}{q} \qquad \neg q \vee r}{r}$$

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\bot$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\dfrac{\dfrac{\dfrac{p \vee q \qquad \neg p \vee q}{q} \qquad \neg q \vee r}{r} \qquad \neg r}{\bot}$$

Resolution

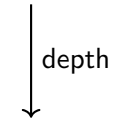
Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\bot$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\dfrac{\dfrac{\dfrac{p \vee q \qquad \neg p \vee q}{q} \qquad \neg q \vee r}{r} \qquad \neg r}{\bot}$$



Hence we conclude that F is unsatisfiable!

Resolution

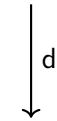
Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Let's try an example

Consider $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$
Writing as set of clauses: $\{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$

$$\dfrac{\dfrac{\dfrac{p \vee q \qquad \neg p \vee q}{q} \qquad \neg q \vee r}{r} \qquad \neg r}{\perp}$$



Hence we conclude that F is unsatisfiable! This is also called a Resolution Proof (of depth 3).

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Four Important Questions

1. Is this procedure correct/sound?

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Four Important Questions

1. Is this procedure correct/sound?
2. Does it terminate?

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Four Important Questions

1. Is this procedure correct/sound?
2. Does it terminate?
3. Can you also use it to show statements (not just to show unsat), e.g., $F \vdash G$?

Resolution

Main idea to check (un)satisfiability of formula F

- Convert F to CNF form as set of clauses (Use Tseitin Encoding!).
- Apply resolution rule and add resolvent to set of clauses, until either $\perp$ is derived or no more applications possible.

Four Important Questions

1. Is this procedure correct/sound?
2. Does it terminate?
3. Can you also use it to show statements (not just to show unsat), e.g., $F \vdash G$?
4. Is it complete? That is, does it suffice to show unsatisfiability? i.e., when it terminates, if we didn't see $\perp$ does this imply F is satisfiable?

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound!

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction.

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

Proof 2: Can we directly show that $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

Proof 2: Can we directly show that $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$

- Consider assignment π satisfying lhs.
- That is, $\pi \models p \vee C$ and $\pi \models \neg p \vee D$

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

Proof 2: Can we directly show that $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$

- Consider assignment π satisfying lhs.
- That is, $\pi \models p \vee C$ and $\pi \models \neg p \vee D$
- If $\pi(p) = 1$, then $\pi \models D$ implies $\pi \models C \vee D$.
- If $\pi(p) = 0$, then $\pi \models C$ implies $\pi \models C \vee D$.
- Thus $\pi \models C \vee D$, i.e., rhs.

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

Proof 2: Can we directly show that $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$

- Consider assignment π satisfying lhs.
- That is, $\pi \models p \vee C$ and $\pi \models \neg p \vee D$
- If $\pi(p) = 1$, then $\pi \models D$ implies $\pi \models C \vee D$.
- If $\pi(p) = 0$, then $\pi \models C$ implies $\pi \models C \vee D$.
- Thus $\pi \models C \vee D$, i.e., rhs.

Thus, we can repeatedly apply this rule and we have,

Corollary

If from F we obtain $\bot$ by a sequence of resolution rule applications then $F \models \bot$, i.e., F is unsat.

1. Soundness of propositional resolution

Theorem

The Resolution rule is sound! i.e., for any clauses C, D and var p: $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$.

Proof 1: Try using natural deduction. **Exercise 8.1**

Proof 2: Can we directly show that $(p \vee C) \wedge (\neg p \vee D) \models C \vee D$

- Consider assignment π satisfying lhs.
- That is, $\pi \models p \vee C$ and $\pi \models \neg p \vee D$
- If $\pi(p) = 1$, then $\pi \models D$ implies $\pi \models C \vee D$.
- If $\pi(p) = 0$, then $\pi \models C$ implies $\pi \models C \vee D$.
- Thus $\pi \models C \vee D$, i.e., rhs.

Thus, we can repeatedly apply this rule and we have,

Corollary

If from F we obtain $\bot$ by a sequence of resolution rule applications then $F \models \bot$, i.e., F is unsat.

Exercise 8.2: Prove this by induction on the length of the resolution sequence.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.
   - By the induction hypothesis (applied to $j, k < i$): $\pi \models C_j$ and $\pi \models C_k$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.
   - By the induction hypothesis (applied to $j, k < i$): $\pi \models C_j$ and $\pi \models C_k$.
   - By the (one-step) Soundness Theorem: $\pi \models C_j$ and $\pi \models C_k$ together imply $\pi \models C_i$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.
   - By the induction hypothesis (applied to $j, k < i$): $\pi \models C_j$ and $\pi \models C_k$.
   - By the (one-step) Soundness Theorem: $\pi \models C_j$ and $\pi \models C_k$ together imply $\pi \models C_i$.

Conclusion

- Taking $i = n$: if $\pi \models F$ then $\pi \models C_n = \bot$.

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.
   - By the induction hypothesis (applied to $j, k < i$): $\pi \models C_j$ and $\pi \models C_k$.
   - By the (one-step) Soundness Theorem: $\pi \models C_j$ and $\pi \models C_k$ together imply $\pi \models C_i$.

Conclusion

- Taking $i = n$: if $\pi \models F$ then $\pi \models C_n = \bot$.
- But no assignment satisfies $\bot$ (the empty clause)!

Proof of Exercise 8.2: Soundness of the Resolution Procedure

Suppose $C_1, \ldots, C_n$ is a resolution sequence from F ending in $C_n = \bot$: each C_i is either

- a clause of F, or
- the resolvent of two earlier clauses C_j, C_k $(j, k < i)$ in the sequence.

Claim (strong induction on i)

For every assignment π with $\pi \models F$: $\pi \models C_i$, for all $i = 1, \ldots, n$.

1. **Base case:** $C_i \in F$.
   - Since $\pi \models F$, π satisfies every clause of F.
   - So $\pi \models C_i$.
2. **Inductive step:** C_i is the resolvent of C_j, C_k with $j, k < i$.
   - By the induction hypothesis (applied to $j, k < i$): $\pi \models C_j$ and $\pi \models C_k$.
   - By the (one-step) Soundness Theorem: $\pi \models C_j$ and $\pi \models C_k$ together imply $\pi \models C_i$.

Conclusion

- Taking $i = n$: if $\pi \models F$ then $\pi \models C_n = \bot$.
- But no assignment satisfies $\bot$ (the empty clause)!
- So no π satisfies F, i.e., $F \models \bot$, i.e., F is unsatisfiable. □

2. Termination of Resolution

Given formula F, we can write the resolution algo as:

Resolution Algorithm

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till fixed point the following:
   - If possible, apply resolution and obtain new clause C i.e., $C \notin F'$. Set $F' := F' \cup \{C\}$.
3. At fixed point, check if $\perp \in F'$.
   - If yes, conclude F is UNSAT. (This follows from Soundness proof!)
   - Else, we wish to conclude that F is SAT. (To prove! - completeness)

2. Termination of Resolution

Given formula F, we can write the resolution algo as:

Resolution Algorithm

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till fixed point the following:
   - If possible, apply resolution and obtain new clause C i.e., $C \notin F'$. Set $F' := F' \cup \{C\}$.
3. At fixed point, check if $\bot \in F'$.
   - If yes, conclude F is UNSAT. (This follows from Soundness proof!)
   - Else, we wish to conclude that F is SAT. (To prove! - completeness)

But first, why does this terminate?

2. Termination of Resolution

Given formula F, we can write the resolution algo as:

Resolution Algorithm

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till fixed point the following:
   - If possible, apply resolution and obtain new clause C i.e., $C \notin F'$. Set $F' := F' \cup \{C\}$.
3. At fixed point, check if $\bot \in F'$.
   - If yes, conclude F is UNSAT. (This follows from Soundness proof!)
   - Else, we wish to conclude that F is SAT. (To prove! - completeness)

But first, why does this terminate?

- Given F we know set of propositional variables is say n.
- So number of clauses possible is

2. Termination of Resolution

Given formula F, we can write the resolution algo as:

Resolution Algorithm

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till fixed point the following:
   - If possible, apply resolution and obtain new clause C i.e., $C \notin F'$. Set $F' := F' \cup \{C\}$.
3. At fixed point, check if $\bot \in F'$.
   - If yes, conclude F is UNSAT. (This follows from Soundness proof!)
   - Else, we wish to conclude that F is SAT. (To prove! - completeness)

But first, why does this terminate?

- Given F we know set of propositional variables is say n.
- So number of clauses possible is 3^n **Proof: Exercise 8.3!**.

2. Termination of Resolution

Given formula F, we can write the resolution algo as:

Resolution Algorithm

1. Use Tseitin encoding to convert to F' in CNF form and write it as set of clauses.
2. Repeat till fixed point the following:
   - If possible, apply resolution and obtain new clause C i.e., $C \notin F'$. Set $F' := F' \cup \{C\}$.
3. At fixed point, check if $\bot \in F'$.
   - If yes, conclude F is UNSAT. (This follows from Soundness proof!)
   - Else, we wish to conclude that F is SAT. (To prove! - completeness)

But first, why does this terminate?

- Given F we know set of propositional variables is say n.
- So number of clauses possible is 3^n **Proof: Exercise 8.3!**.
- So after these many steps we must stop, as there are no new clauses for sure!

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),
- and hence $F \vdash G$ is derivable (by Completeness of Natural Deduction, recall L5–L6!).

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),
- and hence $F \vdash G$ is derivable (by Completeness of Natural Deduction, recall L5–L6!).

This is sometimes called a proof by refutation; such a proof system that works by deriving $\bot$ is a refutation proof system.

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),
- and hence $F \vdash G$ is derivable (by Completeness of Natural Deduction, recall L5–L6!).

This is sometimes called a proof by refutation; such a proof system that works by deriving $\bot$ is a refutation proof system.

Exercise 8.4

Convert the above steps into a formal Natural Deduction proof.

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),
- and hence $F \vdash G$ is derivable (by Completeness of Natural Deduction, recall L5–L6!).

This is sometimes called a proof by refutation; such a proof system that works by deriving $\bot$ is a refutation proof system.

Exercise 8.4

Convert the above steps into a formal Natural Deduction proof. hint: Proof by contradiction!

3. Using resolution to prove statements

We showed UNSAT using Resolution. Can we also show $F \vdash G$?

- Consider $F' = F \wedge \neg G$ in CNF as set of clauses.
- We apply the resolution proof method on F'.
- If we derive $\bot$ from F', then $F \models G$ (by Soundness of Resolution),
- and hence $F \vdash G$ is derivable (by Completeness of Natural Deduction, recall L5–L6!).

This is sometimes called a proof by refutation; such a proof system that works by deriving $\bot$ is a refutation proof system.

Exercise 8.4

Convert the above steps into a formal Natural Deduction proof. hint: Proof by contradiction!

Before going to completeness, is this algorithm deterministic?

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Example

The following resolutions are between two clauses, with different pivots

$$\frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Example

The following resolutions are between two clauses, with different pivots

$$\frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Example

The following resolutions are between two clauses, with different pivots

$$\frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

- Thus, the algorithm is a non-deterministic proof search!

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Example

The following resolutions are between two clauses, with different pivots

$$\frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

- Thus, the algorithm is a non-deterministic proof search!
- Notice: both resolvents here are tautologies (each contains a variable and its negation) – always true, hence useless for deriving $\bot$.
- In practice, we focus on trying to ensure that we do not do unnecessary resolutions!

Non-unique resolvents

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

Example

The following resolutions are between two clauses, with different pivots

$$\frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \qquad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

- Thus, the algorithm is a non-deterministic proof search!
- Notice: both resolvents here are tautologies (each contains a variable and its negation) – always true, hence useless for deriving $\bot$.
- In practice, we focus on trying to ensure that we do not do unnecessary resolutions!
- However, note that we only have a single rule (resolution) instead of all of ND.
- Thus, we may be more effective in finding the proof strategy. This is what SAT solvers do!

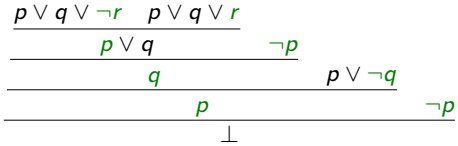
Exercises

Exercise 8.5. Show the following formulas using proof by resolution.

1. Show that $(p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$ is unsatisfiable.
2. Prove that $(p \to q) \wedge (q \to r) \vdash (p \to r)$ is valid.

Solutions: Exercise 8.5

(1) $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$



Hence F is unsatisfiable (depth 4).

Solutions: Exercise 8.5

(1) $F = (p \vee q \vee \neg r) \wedge \neg p \wedge (p \vee q \vee r) \wedge (p \vee \neg q)$

$$\dfrac{\dfrac{\dfrac{\dfrac{p \vee q \vee \neg r \quad p \vee q \vee r}{p \vee q} \quad \neg p}{q} \quad p \vee \neg q}{p} \quad \neg p}{\bot}$$

Hence F is unsatisfiable (depth 4).

(2) $(p \to q) \wedge (q \to r) \vdash (p \to r)$?

By refutation: negate the goal.

$F' = (p \to q) \wedge (q \to r) \wedge \neg(p \to r)$

But $\neg(p \to r) \equiv p \wedge \neg r$, giving clauses $\{(\neg p \vee q),\ (\neg q \vee r),\ p,\ \neg r\}$

$$\dfrac{\dfrac{\dfrac{\neg p \vee q \quad p}{q} \quad \neg q \vee r}{r} \quad \neg r}{\bot}$$

Hence unsatisfiable, so $(p \to q) \wedge (q \to r) \models (p \to r)$, i.e., $(p \to q) \wedge (q \to r) \vdash (p \to r)$. □

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F.

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$
$Res^2(F) = Res^1(F) \cup$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$
$Res^2(F) = Res^1(F) \cup \{r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$
$Res^2(F) = Res^1(F) \cup \{r, q \vee r,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.
$Res^0(F) = F$ $\qquad$ $Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$
$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p,$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p, \bot\}$

4. Completeness

- Is resolution proof system, i.e., proof system where the only rule is Resolution, complete?
- In other words, if F is unsatisfiable, are we guaranteed to derive $F \vdash \bot$ via resolution?
- We need to talk about the set of clauses derived by resolution...

Clauses derivable with proofs of depth at most n

We define the set $Res^n(F)$ of clauses that are derivable via resolution proof of depth at most n from the set of clauses F. Formally, given set F of clauses, we define

- $Res^0(F) := F$,
- $Res^{n+1}(F) := Res^n(F) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(F)\}$

Example

Let $F = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(F) = F$

$Res^1(F) = F \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(F) = Res^1(F) \cup \{r, q \vee r, p, \neg p, \bot\}$

$Res^3(F) = Res^2(F)$.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

- Let F be a set of clauses. Then there exists m such that $Res^{m+1}(F) = Res^m(F)$.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

- Let F be a set of clauses. Then there exists m such that $Res^{m+1}(F) = Res^m(F)$.
- We define $Res^*(F) \triangleq Res^m(F)$.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

- Let F be a set of clauses. Then there exists m such that $Res^{m+1}(F) = Res^m(F)$.
- We define $Res^*(F) \triangleq Res^m(F)$.
- Note: Res^* only grows as we add more clauses – if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$. We will call this weakening.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

- Let F be a set of clauses. Then there exists m such that $Res^{m+1}(F) = Res^m(F)$.
- We define $Res^*(F) \triangleq Res^m(F)$.
- Note: Res^* only grows as we add more clauses – if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$. We will call this weakening.

We will now show the main theorem:

Theorem

If a finite set of clauses F is unsatisfiable, $\bot \in Res^*(F)$.

All derivable clauses

Since there are only finitely many variables in F, we can only derive finitely many clauses, so $Res^n(F)$ must saturate at some time point. (Same argument as termination of algo!)

- Let F be a set of clauses. Then there exists m such that $Res^{m+1}(F) = Res^m(F)$.
- We define $Res^*(F) \triangleq Res^m(F)$.
- Note: Res^* only grows as we add more clauses – if $F \subseteq F'$ then $Res^*(F) \subseteq Res^*(F')$. We will call this weakening.

We will now show the main theorem:

Theorem

If a finite set of clauses F is unsatisfiable, $\bot \in Res^*(F)$.

Finiteness matters: for infinite sets of clauses this needs compactness, which we will not cover here.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\bot \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F. **Exercise 8.6: Show why we can assume this!**

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\bot \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F. **Exercise 8.6: Show why we can assume this!**

base case:
p is the only variable in F.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\perp \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F. **Exercise 8.6: Show why we can assume this!**

base case:

p is the only variable in F.
Now, if F is unsatisfiable, then either $\perp \in F$ already (trivial: $\perp \in Res^0(F)$), or we must have $\{p, \neg p\} \subseteq F$.

Completeness

Theorem

If a finite set of clauses F is unsatisfiable, $\bot \in Res^*(F)$.

Proof.

We prove the theorem using induction over number of propositional variables in F.
Wlog, we assume that there are no tautology clauses in F. **Exercise 8.6: Show why we can assume this!**

base case:

p is the only variable in F.
Now, if F is unsatisfiable, then either $\bot \in F$ already (trivial: $\bot \in Res^0(F)$), or we must have $\{p, \neg p\} \subseteq F$.
In the latter case, applying the resolution rule with $C = D = \emptyset$, we have the following resolution derivation of $\bot$.

$$\frac{p \qquad \neg p}{\bot}$$

...