# Introduction to Mathematical Logic

For CS Students

CS104/CS108

**Yida TAO (陶伊达)**

2024 年 4 月 9 日

# Table of Contents

# Types of Proof Systems



We will introduce 3 formal proof systems.

- Hilbert-style system ($\Sigma \vdash_H A$): many axioms and only one rule. The deduction is linear.
- Natural Deduction System ($\Sigma \vdash_{ND} A$): Few axioms (even none) and many rules. The deductions are tree-like.
- **Resolution ($\Sigma \vdash_{Res} A$): used to prove contradictions.**

# Definition

## 2 Normal Form

- Normal form: a standardized representation of logical formulas
- Transforming a formula into normal form can be useful for various purposes, such as simplification, analysis, model checking, satisfiability testing, or theorem proving.
- Two normal forms in propositional logic:
  - Conjunctive Normal Form: (CNF, 合取范式)
  - Disjunctive Normal Form: (DNF, 析取范式)

**Definition 8.1 Literal (单式/文字)**

A *literal* is an atomic formula or the negation of an atomic formula.

**Definition 8.2 Clause (子式/子句)**

*Disjunctive clause*: a disjunction of literals (literals connected by $\vee$, 析取子式)
*Conjunctive clause*: a conjunction of literals (literals connected by $\wedge$, 合取子式)

# Definition



**Definition 8.3 CNF (合取范式)** 外部 $\wedge$

A formula is in *conjunctive normal form* (CNF) if it is a conjunction of disjunctive clauses.

$$(L_{11} \vee \ldots \vee L_{1n_1}) \wedge \ldots \wedge (L_{k1} \vee \ldots \vee L_{kn_k})$$

**Definition 8.3 DNF (析取范式)** 外部 $\vee$

A formula is in *disjunctive normal form* (DNF) if it is a disjunction of conjunctive clauses.

$$(L_{11} \wedge \ldots \wedge L_{1n_1}) \vee \ldots \vee (L_{k1} \wedge \ldots \wedge L_{kn_k})$$

where each $L_{i,j}$ is a literal, i.e., either an atomic or a negated atomic formula.

# CNF, DNF, Neither, or Both?



1. $\neg p \vee (q \wedge \neg r)$ DNF
2. $\neg p \wedge (q \vee \neg r) \wedge (\neg q \vee r)$ CNF
3. $(\neg p \wedge q) \vee r \vee (q \wedge \neg r \wedge s)$ DNF
4. $((p \vee (\neg q)) \wedge r \wedge ((\neg r) \vee p \vee q))$ CNF
5. $(p \rightarrow q)$ X
6. $(\neg(\neg q))$ X q / ¬q
7. $(\neg((\neg p) \wedge q))$ X
8. $\neg p \vee q$ both
9. $((\neg r) \vee p \vee q)$. DNF
10. $p$ both

**Lemma 8.4**

Let $A$ be a formula in CNF and $B$ be a formula in DNF.
Then $\neg A$ is logically equivalent to a formula in DNF and $\neg B$ is logically equivalent to a formula in CNF.

**Theorem 8.5**

Every formula $A \in Form(\mathscr{L}^p)$ is logically equivalent to some formula in CNF and DNF.
(Proved in class)

If a CNF/DNF is logically equivalent to a formula $A$, we refer to it as $A$'s CNF/DNF.
A formula may have different CNF/DNF.

# Example

CNF: $\equiv (\neg p \vee \neg q) \vee (\neg q \wedge r)$

$\equiv (\neg p \vee \neg q \quad) \wedge (\neg p \vee \neg q \vee r)$

DNF: $\equiv \neg(p \wedge q) \vee (\neg q \wedge r)$

$\equiv \neg p \vee \neg q \vee (\neg q \wedge r)$

What's the CNF and DNF of $(p \wedge q) \rightarrow (\neg q \wedge r)$?

# Recipe



1. Remove $\rightarrow$ and $\leftrightarrow$
   - $A \rightarrow B \equiv \neg A \vee B$
   - $A \leftrightarrow B \equiv (\neg A \vee B) \wedge (A \vee \neg B)$
   - $A \leftrightarrow B \equiv (A \wedge B) \vee (\neg A \wedge \neg B)$
2. Get rid of all double negations and apply DeMorgan's rules wherever possible.
   - $\neg\neg A \equiv A$
   - $\neg(A_1 \wedge \ldots \wedge A_n) \equiv \neg A_1 \vee \ldots \vee \neg A_n$
   - $\neg(A_1 \vee \ldots \vee A_n) \equiv \neg A_1 \wedge \ldots \wedge \neg A_n$
3. Apply the distributivity rule wherever possible.
   - $A \wedge (B_1 \vee \ldots \vee B_n) \equiv (A \wedge B_1) \vee \ldots \vee (A \wedge B_n)$
   - $A \vee (B_1 \wedge \ldots \wedge B_n) \equiv (A \vee B_1) \wedge \ldots \wedge (A \vee B_n)$

**Theorem 8.6**

A formula $B$ is called the *principal* CNF/DNF (主合取/析取范式) of the formula $A$ if:

- $B$ is a CNF/DNF of $A$.
- Each clause in $B$ contains all propositional variables in $A$ exactly once, and no two clauses are the same.

每个变量在每个从句出现一次

从句不重复

# Example

What's the principal CNF and principal DNF of $(p \wedge q) \to (\neg q \wedge r)$?

# Example

Let’s send three people A, B, and C to complete a task, subject to the following conditions:

- If A goes, then C also goes.
- If B goes, then C cannot go.
- If C does not go, then either A goes or B goes.

What are the possible solutions that satisfy these conditions?

# Table of Contents

**Resolution** (归结/消解原理) is one of the most widely used systems for computer-aided proofs. It has two distinctive features.

- It applies only to formulas in CNF. Thus we do some preliminary work before starting an actual proof.
- It is used to prove contradictions. That is, a proof aims to conclude a special "contradiction formula" $\perp$.

For this reason, Resolution is sometimes called a refutation (反驳) system.

The Resolution inference rule: for any proposition variable $p$ and formulas $\alpha$ and $\beta$:

$$\frac{(\alpha \vee p) \qquad ((\neg p) \vee \beta)}{(\alpha \vee \beta)}$$

In a Resolution proof, we consider only CNF formulas.
Each step of the proof produces one clause from two previous clauses.

The Resolution inference rule: for any proposition variable $p$ and formulas $\alpha$ and $\beta$:

$$\frac{(\alpha \vee p) \qquad ((\neg p) \vee \beta)}{(\alpha \vee \beta)}$$

Special case: Unit resolution:

$$\frac{(\alpha \vee p) \qquad (\neg p)}{\alpha}$$

Special case: Contradiction:

$$\frac{p \qquad (\neg p)}{\perp}$$

无文字

In CNF, we treat $\perp$ as a clause containing no literal, i.e., the empty set $\varnothing$. Since it contains no true literal, it is false.

To prove $\Sigma \vdash_{Res} \varphi$ via a Resolution refutation:

1. Resolution only yields contradictions. Hence, rather than proving $\Sigma \vdash_{Res} \varphi$, we prove $\Sigma \cup \{\neg\varphi\} \vdash_{Res} \bot$ instead.
2. The resolution rule only applies to disjunctions ($\vee$). Hence, we first convert each formula in $\Sigma$ and $\neg\varphi$ to CNF.
3. Split the CNF formulas at the $\wedge$s, yielding a set of disjunctive clauses (see next slide).

分隔 $\wedge$处

# A note on step 3

In a CNF formula, no clause appears more than once, and the order of clauses does not matter. Thus, we can think of a CNF formula as simply a *set* of clauses.

For example, the formula

$$((p \vee q) \wedge ((q \vee (\neg r)) \wedge s))$$

can be described by the set of clauses

$$\{p, q\}, \{q, \neg r\}, \{s\}$$

# The Resolution Proof Procedure

To prove $\Sigma \vdash_{Res} \varphi$ via a Resolution refutation:

1. Resolution only yields contradictions. Hence, rather than proving $\Sigma \vdash_{Res} \varphi$, we prove $\Sigma \cup \{\neg\varphi\} \vdash_{Res} \bot$ instead.
2. The resolution rule only applies to disjunctions ($\vee$). Hence, we first convert each formula in $\Sigma$ and $\neg\varphi$ to CNF.
3. Split the CNF formulas at the $\wedge$s, yielding a set of clauses.
4. From the resulting set of clauses, keep applying the resolution inference rule until either:
   - We have the empty clause $\bot$. In this case, we proved that $\Sigma \vdash_{Res} \varphi$
   - The rule can no longer be applied to give a new formula. In this case, $\varphi$ cannot be proven from $\Sigma$.

Prove that $\{p, q\} \vdash_{res} (p \wedge q)$

Step 1: Negating the conclusion and move it to the premise.

$$\{p, q, \neg(p \wedge q)\}$$

Step 2: Converting all premises to CNF.

$$\{p, q, ((\neg p) \vee (\neg q))\}$$

Step 3: Split CNF at the $\wedge$s, resulting a set-notation for premises.

$$\{p\}, \{q\}, \{\neg p, \neg q\}$$

# Example

## 3 Resolution

Step 4: Keep applying the resolution inference rule, until we get a contradiction.

| | | |
|---|---|---|
| 1. | $\{p\}$ | Premise |
| 2. | $\{q\}$ | Premise |
| 3. | $\{\neg p, \neg q\}$ | Premise |
| 4. | $\{\neg q\}$ | 1,3 |
| 5. | $\perp$ | 2,4 |

(Not that $\{\}$ and $\perp$ are the same thing).
In this case, we finished the proof.

不用 {}

You may also write the proof without using the set notation.

| | | |
|---|---|---|
| 1. | $p$ | Premise |
| 2. | $q$ | Premise |
| 3. | $((\neg p) \vee (\neg q))$ | Premise |
| 4. | $(\neg q)$ | 1,3 |
| 5. | $\perp$ | 2,4 |

# Exercise

## 3 Resolution

Prove that $\{(p \to q), (q \to r) \vdash_{res} (p \to r)\}$

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

$\neg(\neg p \vee r)$

6. $q$ — 1, 4

$p \wedge \neg r$

7. $r$ — 2, 6
8. $\bot$ — 5, 7

# Satisfiability (SAT) Solvers



Determining the satisfiability of a set of propositional formulas is a fundamental problem in computer science.

Examples:

- software and hardware verification
- automatic generation of test patterns
- Planning
- Scheduling

......many problems of practical importance can be formulated as determining the satisfiability of a set of formulas.

Modern SAT solvers can often solve hard real-world instances with over a million propositional variables and several million clauses.

Many SAT solvers are open source systems.

# Resolving Software Dependencies



Software has many dependencies, which construct a dependency graph.
We might see constraints like “liba requires libbase ≥ 1.5” and “libb requires libbase ≥ 1.4.7.”

限制

Google’s package management system: Version-satisfiability solvers are very much akin to SAT-solvers in logic : given a set of constraints (version requirements on dependency edges), can we find a set of versions for the nodes in question that satisfies all constraints?

# Symbolic Execution and Test Generation

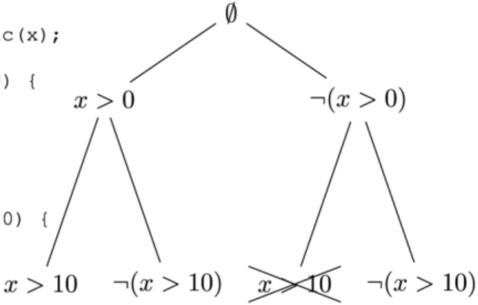


Use symbolic values for inputs, and use SAT solver to resolve what input values (test cases) cause each path of a program to execute.

```
int x;
mksymbolic(x);

if (x > 0) {
  ...
} else {
  ...
}

if (x > 10) {
  ...
} else {
  ...
}
```



# Coursework



- Assignment 4

# Readings

## Optional

- TextF: Chapter 4.1, 4.2, 4.3

# Introduction to Mathematical Logic

*Thank you for listening!*
*Any questions?*