Introduction to Mathematical Logic

For CS Students

CS104

Yida TAO (陶伊达)

2025 年 4 月 21 日

Table of Contents

- Warm up
- Normal Form
- Resolution
- Applications

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.**

Table of Contents

2 Normal Form

- Warm up
- Normal Form
- Resolution
- Applications

Definition

2 Normal Form

- Normal form: a standardized representation of logical formulas
- 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 8.3 CNF (合取范式)

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)

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. $\neg p \vee (q \wedge \neg r)$
2. $\neg p \wedge (q \vee \neg r) \wedge (q \vee r)$
3. $(\neg p \wedge q) \vee r \vee (q \wedge \neg r \wedge s)$
4. $((p \vee (\neg q)) \wedge r \wedge ((\neg r) \vee p \vee q))$
5. $(p \rightarrow q)$
6. $(\neg((\neg p) \wedge q))$
7. $(p \vee q) \wedge ((p \wedge r) \vee (q \wedge s))$
8. $\neg p \wedge q$
9. $((\neg r) \vee p \vee q)$.
10. p

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)

Recall: Every n-ary boolean function is **definable** in terms of only the connectives from the adequate set $\{\neg, \vee, \wedge\}$.

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

Example: find the CNF and DNF of $(p \wedge q) \rightarrow (\neg q \wedge r)$

Converting to CNF/DNF: 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.
   - $A \wedge (B_1 \vee \ldots \vee B_n) \equiv (A \wedge B_1) \vee \ldots \vee (A \wedge B_n)$
   - $A \vee (B_1 \wedge \ldots \wedge B_n) \equiv (A \vee B_1) \wedge \ldots \wedge (A \vee B_n)$

A formula may have different CNF/DNF.

$$
\begin{aligned}
A \vee (B \wedge C) &\equiv (A \vee B) \wedge (A \vee C) && \text{(Distributive law)} \\
&\equiv (A \vee C) \wedge (A \vee B) && \text{(Commutative law)} \\
&\equiv (A \vee C) \wedge (A \vee B) \wedge (A \vee B \vee C) && \text{(Redundant clauses added)}
\end{aligned}
$$

Principal CNF/DNF

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.

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?

▶ Warm up

▶ Normal Form

▶ Resolution

▶ Applications

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.

Inference Rules

3 Resolution

The Resolution inference rule: for any proposition variable p and formulas α and β:

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

In a Resolution proof, we consider only CNF formulas.

Each step of the proof produces one clause (Resolvent, 消解子句) from two previous clauses by applying the resolution rule.

Intuition: Assuming both premises are true, then at least one of α and β must be true.

The Resolution inference rule: for any proposition variable p and formulas α and β:

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

Special case: Unit resolution:

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

Special case: Contradiction:

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

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

A note on step 3

A CNF is a set of disjunctive clauses; A clause is a set of literals (the order doesn't matter; no redundancy). Thus, we can use set notations for CNF.
For example, the formula

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

can be described by the set of clauses

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

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.

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 $\Sigma \cup \{\neg\varphi\} \vdash_{Res} \perp$ instead.
2. The resolution rule only applies to disjunctions ($\vee$). Hence, we first convert each formula in Σ 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 $\perp$. 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, φ cannot be proven from Σ.

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

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

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

Step 2: Converting all premises to CNF.

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

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

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

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.

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

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

Soundness and Completeness

The Resolution proof system is sound and complete.

Theorem 8.7

If $\{\alpha_1, \ldots, \alpha_m\} \vdash_{Res} \perp$ (i.e., there is a resolution refutation with premises as CNF clauses $\alpha_1, \ldots, \alpha_m$ and conclusion $\perp$), then the set $\{\alpha_1, \ldots, \alpha_m\}$ is not satisfiable.

Theorem 8.8

If there is no proof of $\perp$ from a finite set Σ of premises in CNF, then Σ is satisfiable.

Table of Contents

4 Applications

▶ Warm up

▶ Normal Form

▶ Resolution

▶ Applications

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 (some open sourced) can often solve hard real-world instances with over a million propositional variables and several million clauses.

Resolving Software Dependencies

Software has many dependencies. 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? For example:

- A requires B or C
- B requires D
- C conflicts with D

We encode these using propositional logic in CNF:

1. $A \to (B \lor C) \equiv \neg A \lor B \lor C$
2. $B \to D \equiv \neg B \lor D$
3. $\neg(C \land D) \equiv \neg C \lor \neg D$

Resolving Software Dependencies

How SAT Solvers help:

- Each dependency and version constraint is translated into propositional formulas.
- The resulting CNF formulas are passed to a SAT solver.
- The SAT solver uses resolution to try deriving contradictions (i.e., unsatisfiable) or find satisfying assignments.

Readings

Optional

- TextF: Chapter 4

Introduction to Mathematical Logic

Thank you for listening!
Any questions?