

Formal Methods in Software Engineering

Propositional Logic — Spring 2025

Konstantin Chukharev

§1 Propositional Logic

Motivation

- Boolean functions are at the core of logic-based reasoning.
- A Boolean function $F(X_1, \dots, X_n)$ describes the output of a system based on its inputs.
- Boolean gates (AND, OR, NOT) form the building blocks of digital circuits.
- Propositional logic formalizes reasoning about Boolean functions and circuits.
- **Applications:**
 - Digital circuit design.
 - Verification and synthesis of hardware and software.
 - Expressing logical constraints in AI and optimization problems.
 - Automated reasoning and theorem proving.

Boolean Circuits and Propositional Logic

Boolean circuit is a directed acyclic graph (DAG) of Boolean gates.

- Inputs: Propositional variables.
- Outputs: Logical expressions describing the circuit's behavior.

“Can the output of a circuit ever be true?”

- Propositional logic provides a formal framework to answer such questions.

Real-world examples:

- Error detection circuits.
- Arithmetic logic units (ALUs) in processors.
- Routing logic in network devices.

What is Logic?

A formal logic is defined by its **syntax** and **semantics**.

□ **Syntax**

- An **alphabet** Σ is a set of symbols.
- A finite sequence of symbols (from Σ) is called an **expression** or **string** (over Σ).
- A set of rules defines the **well-formed** expressions.

□ **Semantics**

- Gives meaning to (well-formed) expressions.

Syntax of Propositional Logic

□ Alphabet

1. Logical connectives: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.
2. Propositional variables: A_1, A_2, \dots, A_n .
3. Parentheses for grouping: $(,)$.

□ Well-Formed Formulas (WFFs)

Valid (**well-formed**) expressions are defined **inductively**:

1. A single propositional symbol (e.g. A) is a WFF.
2. If α and β are WFFs, so are: $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$.
3. No other expressions are WFFs.

Syntax of Propositional Logic [2]

□ Conventions

- Large variety of propositional variables: $A, B, C, \dots, p, q, r, \dots$
- Outer parentheses can be omitted: $A \wedge B$ instead of $(A \wedge B)$.
- Operator precedence: $\neg > \wedge > \vee > \rightarrow > \leftrightarrow$.
- Left-to-right associativity for \wedge and \vee : $A \wedge B \wedge C = (A \wedge B) \wedge C$.
- Right-to-left associativity for \rightarrow : $A \rightarrow B \rightarrow C = A \rightarrow (B \rightarrow C)$.

Semantics of Propositional Logic

- Each propositional variable is assigned a truth value: T (true) or F (false).
- More formally, *interpretation* $\nu : V \rightarrow \{0, 1\}$ assigns truth values to all variables (atoms).
- Truth values of complex formulas are computed (evaluated) recursively:
 1. $\llbracket p \rrbracket_\nu \triangleq \nu(p)$, where $p \in V$ is a propositional variable
 2. $\llbracket \neg \alpha \rrbracket_\nu \triangleq 1 - \llbracket \alpha \rrbracket_\nu$
 3. $\llbracket \alpha \wedge \beta \rrbracket_\nu \triangleq \min(\llbracket \alpha \rrbracket_\nu, \llbracket \beta \rrbracket_\nu)$
 4. $\llbracket \alpha \vee \beta \rrbracket_\nu \triangleq \max(\llbracket \alpha \rrbracket_\nu, \llbracket \beta \rrbracket_\nu)$
 5. $\llbracket \alpha \rightarrow \beta \rrbracket_\nu \triangleq (\llbracket \alpha \rrbracket_\nu \leq \llbracket \beta \rrbracket_\nu) = \max(1 - \llbracket \alpha \rrbracket_\nu, \llbracket \beta \rrbracket_\nu)$
 6. $\llbracket \alpha \leftrightarrow \beta \rrbracket_\nu \triangleq (\llbracket \alpha \rrbracket_\nu = \llbracket \beta \rrbracket_\nu) = 1 - |\llbracket \alpha \rrbracket_\nu - \llbracket \beta \rrbracket_\nu|$

§2 Foundations

Truth Tables

α	β	γ	$\alpha \wedge (\beta \vee \neg \gamma)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Normal Forms

- **Conjunctive Normal Form (CNF):**

- A formula is in CNF if it is a conjunction of *clauses* (disjunctions of literals).

Example: $(A \vee B) \wedge (\neg A \vee C) \wedge (B \vee \neg C)$ – CNF with 3 clauses.

- **Disjunctive Normal Form (DNF):**

- A formula is in DNF if it is a disjunction of *cubes* (conjunctions of literals).

Example: $(\neg A \wedge B) \vee (B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$ – DNF with 3 cubes.

- **Algebraic Normal Form (ANF):**

- A formula is in ANF if it is a sum of *products* of variables (or a constant 1).

Example: $B \oplus AB \oplus ABC$ – ANF with 3 terms.

Logical Laws and Tautologies

- **Associative and Commutative** laws for \wedge , \vee , \leftrightarrow :
 - $A \circ (B \circ C) \equiv (A \circ B) \circ C$
 - $A \circ B \equiv B \circ A$
- **Distributive laws:**
 - $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
 - $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- **Negation:**
 - $\neg\neg A \equiv A$
- **De Morgan's laws:**
 - $\neg(A \wedge B) \equiv \neg A \vee \neg B$
 - $\neg(A \vee B) \equiv \neg A \wedge \neg B$

Logical Laws and Tautologies [2]

- **Implication:**

- $(A \rightarrow B) \equiv (\neg A \vee B)$

- **Contraposition:**

- $(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$

- **Law of Excluded Middle:**

- $(A \vee \neg A) \equiv \top$

- **Contradiction:**

- $(A \wedge \neg A) \equiv \perp$

- **Exportation:**

- $((A \wedge B) \rightarrow C) \equiv (A \rightarrow (B \rightarrow C))$

Completeness of Connectives

- All Boolean functions can be expressed using $\{\neg, \wedge, \vee\}$ (so called “*standard Boolean basis*”).
- Even smaller sets are sufficient:
 - $\{\neg, \wedge\}$ – AIG (And-Inverter Graph), see also: AIGER format.
 - $\{\neg, \vee\}$
 - $\{\overline{\wedge}\}$ – NAND
 - $\{\overline{\vee}\}$ – NOR

Incompleteness of Connectives

To prove that a set of connectives is incomplete, we find a property that is true for all WFFs expressed using those connectives, but that is not true for some Boolean function.

Example: $\{\wedge, \rightarrow\}$ is not complete.

Proof: Let α be a WFF which uses only these connectives. Let ν be an interpretation such that $\nu(A_i) = 1$ for all propositional variables A_i . Next, we prove by induction that $\llbracket \alpha \rrbracket_\nu = 1$.

- Base case:
 - $\llbracket A_i \rrbracket_\nu = \nu(A_i) = 1$
- Inductive step:
 - $\llbracket \beta \wedge \gamma \rrbracket_\nu = \min(\llbracket \beta \rrbracket_\nu, \llbracket \gamma \rrbracket_\nu) = 1$
 - $\llbracket \beta \rightarrow \gamma \rrbracket_\nu = \max(1 - \llbracket \beta \rrbracket_\nu, \llbracket \gamma \rrbracket_\nu) = 1$

Thus, $\llbracket \alpha \rrbracket_\nu = 1$ for all WFFs α built from $\{\wedge, \rightarrow\}$. However, $\llbracket \neg A_1 \rrbracket_\nu = 0$, so there is no such formula α tautologically equivalent to $\neg A_1$. □

§3 Semantical Aspects

Validity, Satisfiability, Entailment

□ Validity

- α is a **tautology** if α is true under all truth assignments.

Formally, α is **valid**, denoted " $\models \alpha$ ", iff $\llbracket \alpha \rrbracket_\nu = 1$ for all interpretations $\nu \in \{0, 1\}^V$.

- α is a **contradiction** if α is false under all truth assignments.

Formally, α is **unsatisfiable** if $\llbracket \alpha \rrbracket_\nu = 0$ for all interpretations $\nu \in \{0, 1\}^V$.

□ Satisfiability

- α is **satisfiable (consistent)** if there exists an interpretation $\nu \in \{0, 1\}^V$ where $\llbracket \alpha \rrbracket_\nu = 1$.

When α is satisfiable by ν , denoted $\nu \models \alpha$, this interpretation is called a **model** of α .

- α is **falsifiable (invalid)** if there exists an interpretation $\nu \in \{0, 1\}^V$ where $\llbracket \alpha \rrbracket_\nu = 0$.

□ Entailment

- Let Γ be a set of WFFs. Then Γ **tautologically implies (semantically entails)** α , denoted $\Gamma \models \alpha$, if every truth assignment that satisfies all formulas in Γ also satisfies α .
- Formally, $\Gamma \models \alpha$ iff for all interpretations $\nu \in \{0, 1\}^V$ and formulas $\beta \in \Gamma$, if $\nu \models \beta$, then $\nu \models \alpha$.
- Note: $\alpha \models \beta$, where α and β are WFFs, is just a shorthand for $\{\alpha\} \models \beta$.

Implication vs Entailment

The **implication** operator (\rightarrow) is a syntactic construct, while **entailment** (\models) is a semantical relation.

They are related as follows: $\alpha \rightarrow \beta$ is valid iff $\alpha \models \beta$.

Example: $A \rightarrow (A \vee B)$ is valid (a tautology), and $A \models A \vee B$

A	B	$A \vee B$	$A \rightarrow (A \vee B)$	$A \models A \vee B$
0	0	0	1	—
0	1	1	1	—
1	0	1	1	OK
1	1	1	1	OK

Examples

- $A \vee B \wedge (\neg A \wedge \neg B)$ is satisfiable, but not valid.
- $A \vee B \wedge (\neg A \wedge \neg B) \wedge (A \leftrightarrow B)$ is unsatisfiable.
- $\{A \rightarrow B, A\} \models B$
- $\{A, \neg A\} \models A \wedge \neg A$
- $\neg(A \wedge B)$ is tautologically equivalent to $\neg A \vee \neg B$.

Duality of SAT vs VALID

- **SAT:** Given a formula α , determine if it is satisfiable.

$$\exists \nu. \llbracket \alpha \rrbracket_\nu$$

- **VALID:** Given a formula α , determine if it is valid.

$$\forall \nu. \llbracket \alpha \rrbracket_\nu$$

- **Duality:** α is valid iff $\neg\alpha$ is unsatisfiable.
- Note: SAT is NP, but VALID is co-NP.

Solving SAT using Truth Tables

Algorithm for satisfiability:

To check whether α is satisfiable, construct a truth table for α . If there is a row where α evaluates to true, then α is satisfiable. Otherwise, α is unsatisfiable.

Algorithm for semantical entailment (tautological implication):

To check whether $\{\alpha_1, \dots, \alpha_k\} \models \beta$, check the satisfiability of $(\alpha_1 \wedge \dots \wedge \alpha_k) \wedge (\neg\beta)$. If it is unsatisfiable, then $\{\alpha_1, \dots, \alpha_k\} \models \beta$. Otherwise, $\{\alpha_1, \dots, \alpha_k\} \not\models \beta$.

Compactness

Recall:

- A WFF α is **satisfiable** if there exists an interpretation ν such that $\nu \models \alpha$.
- Hereinafter, let Γ denote a *finite* set of WFFs, and Σ denote a *possibly infinite* set of WFFs.
- A set of WFFs Σ is **satisfiable** if there exists an interpretation ν that satisfies all formulas in Σ .
- A set of WFFs Σ is **finitely satisfiable** if every finite subset of Σ is satisfiable.

Theorem 1 (Compactness Theorem): A set of WFFs Σ is satisfiable iff it is finitely satisfiable.

Proof (\Rightarrow): Suppose Σ is satisfiable, i.e. there exists an interpretation ν that satisfies all formulas in Σ .

This direction is trivial: any subset of a satisfiable set is clearly satisfiable.

- For each finite subset $\Sigma' \subseteq \Sigma$, ν also satisfies all formulas in Σ' .
- Thus, every finite subset of Σ is satisfiable.

□

Compactness [2]

Proof (\Leftarrow): Suppose Σ is finitely satisfiable, i.e. every finite subset of Σ is satisfiable.

Construct a *maximal* finitely satisfiable set Δ as follows:

- Let $\alpha_1, \dots, \alpha_n, \dots$ be a fixed enumeration of all WFFs.
 - *This is possible since the set of all sequences of a countable set is countable.*
- Then, let:

$$\Delta_0 = \Sigma,$$
$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable,} \\ \Delta_n \cup \{\neg\alpha_{n+1}\} & \text{otherwise.} \end{cases}$$

- *Note that each Δ_n is finitely satisfiable by construction.*

Compactness [3]

- Let $\Delta = \bigcup_{n \in \mathbb{N}} \Delta_n$. Note:
 1. $\Sigma \subseteq \Delta$
 2. $\alpha \in \Delta$ or $\neg\alpha \in \Delta$ for any WFF α
 3. Δ is finitely satisfiable by construction.

Now we need to show that Δ is satisfiable (and thus $\Sigma \subseteq \Delta$ is also satisfiable).

Define an interpretation ν as follows: for each propositional variable p , let $\nu(p) = 1$ iff $p \in \Delta$.

We claim that $\nu \models \alpha$ iff $\alpha \in \Delta$. The proof is by induction on well-formed formulas.

- Base case:
 - Suppose $\alpha \equiv p$ for some propositional variable p .
 - By definition, $\llbracket p \rrbracket_\nu = \nu(p) = 1$.
- Inductive step:
 - *(Note: we consider only two cases: \neg and \wedge , since they form a complete set of connectives.)*
 - Suppose $\alpha \equiv \neg\beta$.
 - $\llbracket \alpha \rrbracket_\nu = 1$ iff $\llbracket \beta \rrbracket_\nu = 0$ iff $\beta \notin \Delta$ iff $\neg\beta \in \Delta$ iff $\alpha \in \Delta$.

Compactness [4]

- ▶ Suppose $\alpha \equiv \beta \wedge \gamma$.
 - $\llbracket \alpha \rrbracket_\nu = 1$ iff both $\llbracket \beta \rrbracket_\nu = 1$ and $\llbracket \gamma \rrbracket_\nu = 1$ iff both $\beta \in \Delta$ and $\gamma \in \Delta$.
 - If both β and γ are in Δ , then $\beta \wedge \gamma$ is in Δ , thus $\alpha \in \Delta$.
 - Why? Because if $\beta \wedge \gamma \notin \Delta$, then $\neg(\beta \wedge \gamma) \in \Delta$. But then $\{\beta, \gamma, \neg(\beta \wedge \gamma)\}$ is a finite subset of Δ that is not satisfiable, which is a contradiction of Δ being finitely satisfiable.
 - Similarly, if either $\beta \notin \Delta$ or $\gamma \notin \Delta$, then $\beta \wedge \gamma \notin \Delta$, thus $\alpha \notin \Delta$.
 - Why? Again, suppose $\beta \wedge \gamma \in \Delta$. Since $\beta \notin \Delta$ or $\gamma \notin \Delta$, at least one of $\neg\beta$ or $\neg\gamma$ is in Δ . Wlog, assume $\neg\beta \in \Delta$. Then, $\{\neg\beta, \beta \wedge \gamma\}$ is a finite subset of Δ that is not satisfiable, which is a contradiction of Δ being finitely satisfiable.
 - Thus, $\llbracket \alpha \rrbracket_\nu = 1$ iff $\alpha \in \Delta$.

This shows that $\llbracket \alpha \rrbracket_\nu = 1$ iff $\alpha \in \Delta$, thus Δ is satisfiable by ν . □

Compactness [5]

Corollary 1.1: If $\Sigma \models \alpha$, then there is a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \models \alpha$.

Proof: Suppose that $\Sigma_0 \not\models \alpha$ for every finite $\Sigma_0 \subseteq \Sigma$.

Then, $\Sigma_0 \cup \{\neg\alpha\}$ is satisfiable for every finite $\Sigma_0 \subseteq \Sigma$, that is, $\Sigma \cup \{\neg\alpha\}$ is finitely satisfiable.

Then, by the compactness theorem, $\Sigma \cup \{\neg\alpha\}$ is satisfiable, thus $\Sigma \not\models \alpha$, which contradicts the theorem assumption that $\Sigma \models \alpha$. □

§4 Proof Systems

Natural Deduction

- **Natural deduction** is a proof system for propositional logic.
- **Axioms:**
 - **No axioms.**
- **Rules:**
 - **Introduction:** \wedge -introduction, \vee -introduction, \rightarrow -introduction, \neg -introduction.
 - **Elimination:** \wedge -elimination, \vee -elimination, \rightarrow -elimination, \neg -elimination.
 - **Reduction ad Absurdum**
 - **Law of Excluded Middle** (note: forbidden in *intuitionistic* logic)
- **Proofs** are constructed by applying rules to assumptions and previously derived formulas.

$$\underbrace{A_1, \dots, A_n \vdash A}_{\text{sequent}}$$

$$\frac{\Gamma_1 \vdash (\textit{premise 1}) \quad \Gamma_2 \vdash (\textit{premise 2}) \quad \dots}{\Gamma \vdash (\textit{conclusion})} \text{ rule name}$$

Inference Rules

$$\frac{}{\Gamma \vdash \varphi \vee \neg \varphi}$$
law of excluded middle

$$\frac{}{\Gamma, \varphi \vdash \varphi}$$
assumption

$$\frac{\Gamma \vdash \alpha \quad \Gamma \vdash \neg \alpha}{\Gamma \vdash \beta}$$
reduction ad absurdum

$$\frac{\Gamma \vdash \alpha \wedge \beta}{\Gamma \vdash \alpha}$$
 \wedge -elimination

$$\frac{\Gamma \vdash \alpha \wedge \beta}{\Gamma \vdash \beta}$$
 \wedge -elimination

$$\frac{\Gamma \vdash \alpha \quad \Gamma \vdash \beta}{\Gamma \vdash \alpha \wedge \beta}$$
 \wedge -introduction

$$\frac{\Gamma \vdash \alpha_1 \vee \alpha_2 \quad \Gamma, \alpha_1 \vdash \beta \quad \Gamma, \alpha_2 \vdash \beta}{\Gamma \vdash \beta}$$
 \vee -elim

$$\frac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha \vee \beta}$$
 \vee -intro

$$\frac{\Gamma \vdash \beta}{\Gamma \vdash \alpha \vee \beta}$$
 \vee -intro

$$\frac{\Gamma \vdash \alpha \quad \Gamma \vdash \alpha \rightarrow \beta}{\Gamma \vdash \beta}$$
 \rightarrow -elimination

$$\frac{\Gamma, \alpha \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta}$$
 \rightarrow -introduction

Example Derivation

Example: $\underbrace{p \wedge q, r}_{\text{premises}} \vdash \underbrace{q \wedge r}_{\text{conclusion}}$

Proof tree:

$$\frac{\frac{\overline{p \wedge q}}{q} \wedge e \quad \frac{\overline{r}}{r}}{q \wedge r} \wedge i$$

Linear proof (Fitch notation):

1. $p \wedge q$ **premise**
2. r **premise**
3. q **$\wedge e$ 1**
4. $q \wedge r$ **$\wedge i$ 2,3**

Exercises

1. $\vdash (b \rightarrow c) \rightarrow ((\neg b \rightarrow \neg a) \rightarrow (a \rightarrow c))$
2. $a \vee b \vdash b \vee a$
3. $a \rightarrow c, b \rightarrow c, a \vee b \vdash c$
4. $\neg a \vee b \vdash a \rightarrow b$
5. $a \rightarrow b \vdash \neg a \vee b$
6. $a \rightarrow b, a \rightarrow \neg b \vdash \neg a$
7. $\neg p \rightarrow \perp \vdash p$ (with allowed $\neg\neg$ E)
8. $\vdash p \vee \neg p$
9. $a \vee b, b \vee c, \neg b \vdash a \wedge c$
10. $a \vee (b \rightarrow a) \vdash \neg a \rightarrow \neg b$
11. $p \rightarrow \neg p \vdash \neg p$
12. $a \rightarrow b, \neg b \vdash \neg a$
13. $((a \rightarrow b) \rightarrow a) \rightarrow a$
14. $\neg a \rightarrow \neg b \vdash b \rightarrow a$
15. $\vdash (a \rightarrow b) \vee (b \rightarrow a)$

Soundness and Completeness

- A formal system is **sound** if every provable formula is true in all models.

- ▶ **Weak soundness:** “every provable formula is a tautology”.

If $\vdash \alpha$, then $\models \alpha$.

- ▶ **Strong soundness:** “every derivable (from Γ) formula is a logical consequence (of Γ)”.

If $\Gamma \vdash \alpha$, then $\Gamma \models \alpha$.

- A formal system is **complete** if every formula true in all models is provable.

- ▶ **Weak completeness:** “every tautology is provable”.

If $\models \alpha$, then $\vdash \alpha$.

- ▶ **Strong completeness:** “every logical consequence (of Γ) is derivable (from Γ)”.

If $\Gamma \models \alpha$, then $\Gamma \vdash \alpha$.

Some Random Links

- <https://plato.stanford.edu/entries/proof-theoretic-semantics/>
- <https://math.stackexchange.com/a/3318545>

TODO

- ☒ Normal forms
- ☐ Canonical normal forms
- ☐ BDDs
- ☒ Natural deduction
- ☐ Sequent calculus
- ☐ Fitch notation
- ☐ Proof checkers
- ☐ Proof assistants
- ☐ Automatic theorem provers
- ☐ Abstract proof systems
- ☐ Intuitionistic logic
- ☒ Soundness and completeness
- ☐ Proof of soundness
- ☐ Proof of completeness