Discrete Mathematics[1]
http://iscasmc.ios.ac.cn/DM2018

Lijun Zhang

March 14, 2018

[1] Materials about the axiomatization are provided by Dr. Wanwei Liu.

Contents

Chapter 1

The Foundations: Logic and Proofs

Logic in Computer Science During the past fifty years there has been extensive, continuous, and growing interaction between logic and computer science. In many respects, logic provides computer science with both a unifying foundational framework and a tool for modeling computational systems. In fact, logic has been called the calculus of computer science. The argument is that logic plays a fundamental role in computer science, similar to that played by calculus in the physical sciences and traditional engineering disciplines. Indeed, logic plays an important role in areas of computer science as disparate as machine architecture, computer-aided design, programming languages, databases, artificial intelligence, algorithms, and computability and complexity. Moshe Vardi

- The origins of logic can be dated back to Aristotle's time.

- One of the central problem for logicians is that: "why is this proof correct/incorrect?"

- The invention of modern logic (Boolean algebra) owes to

George Boole.

- Now, we are interested in: “is the program correct?”

1.1 Propositional Logic – An Appetizer

Definition 1.1.1 (Sets).

- *Fix an universal set U. Set operations: union $\cup$, intersection $\cap$, complement $\overline{A}$.*

- *Set inclusion: $A \subseteq B$ iff for all $a \in A$ it holds $a \in B$. $A = B$ iff $A \subseteq B$ and $B \subseteq A$.*

- *Given a set S, the* power set *of S is the set of all subsets of the set S. The power set is denoted by $\mathcal{P}(S)$, or 2^S.*

- *The* Cartesian product *of sets $A_1, A_2, \ldots, A_n$ is defined by: $A_1 \times \cdots \times A_n := \{(a_1, \ldots, a_n) \mid a_i \in A_i$ for $i = 1, \ldots, n\}$.*

- *The* cardinality *of finite set A, denoted by $|A|$, is the number of its elements. The* principle of inclusion-exclusion*:*

$$|A \cup B| = |A| + |B| - |A \cap B|$$

- *The set of natural numbers $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$ is* countable *infinite.*

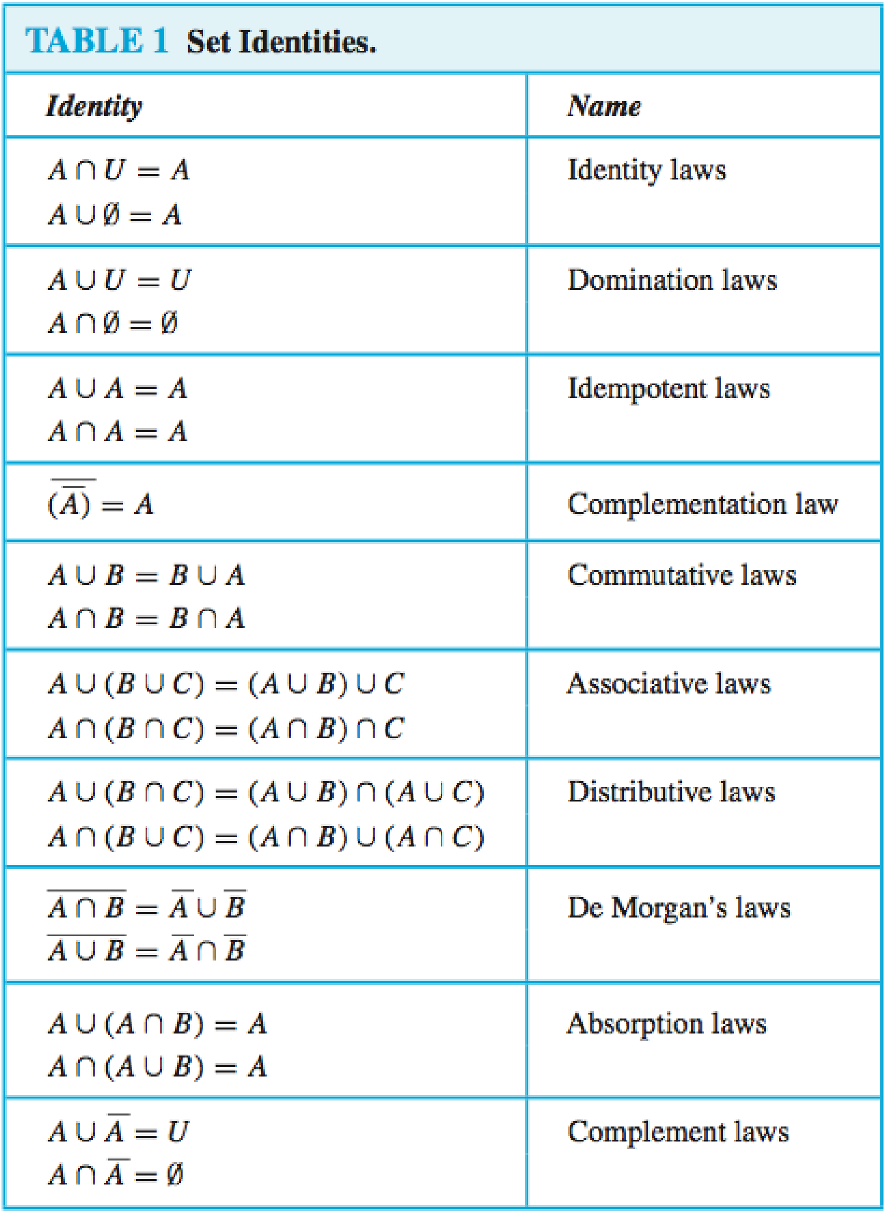


TABLE 1 **Set Identities.**

Identity	*Name*
$A \cap U = A$ $A \cup \emptyset = A$	Identity laws
$A \cup U = U$ $A \cap \emptyset = \emptyset$	Domination laws
$A \cup A = A$ $A \cap A = A$	Idempotent laws
$\overline{(\overline{A})} = A$	Complementation law
$A \cup B = B \cup A$ $A \cap B = B \cap A$	Commutative laws
$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$	Associative laws
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	Distributive laws
$\overline{A \cap B} = \overline{A} \cup \overline{B}$ $\overline{A \cup B} = \overline{A} \cap \overline{B}$	De Morgan's laws
$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$	Absorption laws
$A \cup \overline{A} = U$ $A \cap \overline{A} = \emptyset$	Complement laws

Definition 1.1.2 (Propositional Logic). *Fix a countable set AP of atomic propositions. Syntax of (propositional) formulas in BNF (Backus-Naur form) is given by:*

$$\varphi ::= p \in AP \mid (\neg\varphi) \mid (\varphi \to \varphi)$$

Accordingly,

- *Atomic proposition $p \in AP$ is a formula.*
- *Compound formulas: $(\neg\varphi)$ (negation) and $(\varphi \to \psi)$ (implication), provided that φ and ψ are formulas.*

We omit parentheses if it is clear from the context.
As usually, we view the followings as derived connectives:

- Disjunction: $\varphi \vee \psi \triangleq \neg\varphi \to \psi$
- Conjunction: $\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi)$
- Bi-implication: $\varphi \leftrightarrow \psi \triangleq (\varphi \to \psi) \wedge (\psi \to \varphi)$

Definition 1.1.3 (Precedence of Logical Operators)**.** *Operators* $\neg, \wedge, \vee, \to, \leftrightarrow$ *have precedence* $1, 2, 3, 4, 5$*, respectively.*

1.1.1 Applications of Propositional Logic

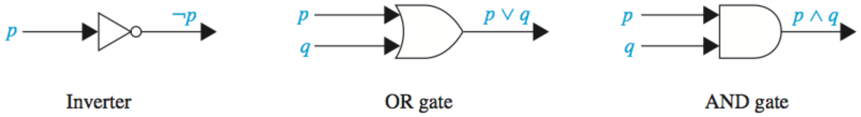
Definition 1.1.4 (Applications)**.**

- *System Specifications: The value of x is not equal to* 0*.*
- *Boolean Searches: (one | two) - (three)*
- *Logic Puzzles. Knights always tell the truth, and the opposite knaves always lie. A says: "B is a knight". B says "The two of us are opposite types"*

Definition 1.1.5 (Logic Circuit)**.**

- *Propositional logic can be applied to the design of computer hardware.* ***Claude Shannon***

- *A logic circuit receives input signals* $p_1, p_2, \dots, p_n$ *and produces an output* s*. Complicated digital circuits are constructed from three basic circuits, called* gates*.*



- *Build a digital circuit producing* $(p \vee \neg r) \wedge (\neg p \vee (q \vee \neg r))$*.*

1.2 Induction and Recursion

PRINCIPLE OF MATHEMATICAL INDUCTION To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

- BASIS STEP: We verify that $P(1)$ is true.
- INDUCTIVE STEP: We show that the conditional statement $P(k) \to P(k+1)$ is true for all positive integers k.

Expressed as a rule of inference for *first-order logic*, this proof technique can be stated as:

$$\Phi := (P(1) \wedge \forall k.(P(k) \to P(k+1))) \to \forall n.P(n)$$

Exercise 1.2.1. *Prove* $1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$*.*

STRONG INDUCTION (Second principle of mathematical induction) To prove that $P(n)$ is true for natural all positive integers n, where $P(n)$ is a propositional function, we complete two steps:

- BASIS STEP: We verify that the proposition $P(1)$ is true.

- INDUCTIVE STEP: We show that the conditional statement $(P(1) \wedge P(2) \wedge \ldots \wedge P(k)) \rightarrow P(k+1)$ is true for all positive integers k.

Expressed as a rule of inference for *first-order logic*, this proof technique can be stated as:

$$\Psi := (P(1) \wedge \forall k.(\wedge_{i=1}^{k} P(i) \rightarrow P(k+1))) \rightarrow \forall n.P(n)$$

Exercise 1.2.2. *Prove that if n is a natural number greater than* 1*, then n can be written as the product of primes.*

Recursively Defined Sets and Structures and Structural Induction

Definition 1.2.3 (Strings)**.** *The set Σ^* of* strings *over the alphabet Σ is defined recursively by*

- *BASIS STEP: $\lambda \in \Sigma^*$ (where λ is the empty string containing no symbols).*
- *RECURSIVE STEP: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.*

Definition 1.2.4. *We define the set of* well-formed formulas in propositional logic*, denoted by L, from alphabet $\Sigma := AP \cup \{\neg, \rightarrow, (,)\}$.*

- *BASIS STEP: each $p \in AP$ is a well-formed formula.*
- *RECURSIVE STEP: If φ and ψ are well-formed formulas, i.e., $\varphi, \psi \in L$, then $(\neg\varphi)$, $(\varphi \rightarrow \psi)$ are well-formed formulas.*

Thus, the set of well-formed formulas is a subset of $L \subseteq \Sigma^*$.

STRUCTURAL INDUCTION A proof by structural induction consists of two parts.

- BASIS STEP: Show that the result holds for all elements specified in the basis step.
- RECURSIVE STEP: Show that if the statement is true for each of the elements used to construct new elements in the recursive step of the definition, the result holds for these new elements.

Remark: The validity of structural induction follows from the principle of mathematical induction for the nonnegative integers.

Exercise 1.2.5. *Show that every well-formed formula for compound propositions contains an equal number of left and right parentheses.*

1.3 Propositional Logic and Deduction Systems: a Sound and Complete Axiomatization

This section considers a complete axiomatization system such that, a formula is a tautology if and only if it can be derived by means of the axioms and the deduction rules of the system.

1.3.1 Syntax of Propositional Logic

Fix a countable proposition set AP, then formulas of propositional logic are defined by:

Definition 1.3.1 (Syntax)**.** *Syntax of propositional formulas in BNF (Backus-Naur form) is given by:*

$$\varphi ::= p \in AP \mid (\neg\varphi) \mid (\varphi \to \varphi)$$

It generates recursively the set of well-formed formulas, denoted by L:

- Atomic formula: $p \in AP$ implies $p \in L$.
- Compound formulas: $(\neg\varphi)$ and $(\varphi \to \psi)$, provided that $\varphi, \psi \in L$.

We omit parentheses if it is clear from the context.

As usually, we view the followings as derived connectives:

$$\begin{aligned}
\varphi \vee \psi &\triangleq \neg\varphi \to \psi \\
\varphi \wedge \psi &\triangleq \neg(\neg\varphi \vee \neg\psi) \\
\varphi \leftrightarrow \psi &\triangleq (\varphi \to \psi) \wedge (\psi \to \varphi)
\end{aligned}$$

1.3.2 Normal Forms

We first define the following normal forms.

Conjunctive Normal Form (CNF):

- Every formula φ is a conjunction of clauses $\mathcal{C}_1, \ldots, \mathcal{C}_n$.
- A clause $\mathcal{C}$ is the disjunction of literals $\mathtt{l}_1, \ldots, \mathtt{l}_m$.
- A literal $\mathtt{l}$ is an atomic proposition or the negation of an atomic proposition.

$$\begin{aligned}
\varphi &::= \mathcal{C} \mid \mathcal{C} \wedge \varphi \\
\mathcal{C} &::= \mathtt{l} \mid \mathtt{l} \vee \mathcal{C} \\
\mathtt{l} &::= p \mid \neg p \qquad\qquad \text{(where } p \in AP\text{)}
\end{aligned}$$

Disjunctive Normal Form (DNF):

- Every formula φ is a disjunction of clauses $\mathcal{C}_1, \ldots, \mathcal{C}_n$.
- A clause $\mathcal{C}$ is the conjunction of literals $\mathtt{l}_1, \ldots, \mathtt{l}_m$.
- A literal $\mathtt{l}$ is an atomic proposition or the negation of an atomic proposition.

$$\begin{aligned}
\varphi &::= \mathcal{C} \mid \mathcal{C} \vee \varphi \\
\mathcal{C} &::= \mathtt{l} \mid \mathtt{l} \wedge \mathcal{C} \\
\mathtt{l} &::= p \mid \neg p \qquad\qquad \text{(where } p \in AP\text{)}
\end{aligned}$$

Negation-free Normal Form (NNF):

- Negation may appear only in front of atomic propositions.

$$\varphi ::= p \mid \neg p \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \qquad \text{(where } p \in AP\text{)}$$

1.3.3 The Axiom System for Propositional Logic: the Hilbert's System

The axiom system for propositional logic consists of four parts:

1. the set of symbols, including atomic propositions and logical connectives;
2. the set of well-formed formulas;
3. a set of axioms;
4. a set of rules.

Axioms Given through three axiom schemas:

A1: $\varphi \to (\psi \to \varphi)$.

A2: $(\varphi \to (\psi \to \eta)) \to ((\varphi \to \psi) \to (\varphi \to \eta))$.

A3: $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$.

Definition 1.3.2 (MP Rule)**.** *(Latin for* implication elimination*):*

$$\frac{\varphi \to \psi \quad \varphi}{\psi}$$

Deductive Sequence. Given a formula set Γ, a deductive sequence of φ from Γ is a sequence

$$\varphi_0, \varphi_1, \ldots, \varphi_n$$

where $\varphi_n = \varphi$ and each φ_i should be one of the following cases:

1. $\varphi_i \in \Gamma$.
2. φ_i is an instance of some axiom.
3. There exists some $j, k < i$, such that φ_k is of the form $\varphi_j \to \varphi_i$.

And, we denote by $\Gamma \vdash \varphi$ if there exists such deductive sequence. We write $\Gamma, \psi \vdash \varphi$ for $\Gamma \cup \{\psi\} \vdash \varphi$.

1.3.4 Properties and Examples

With Hilbert's axiom system, we have the following elementary properties:

(Fin) If $\Gamma \vdash \varphi$, then there exists some finite subset Γ' of Γ, such that $\Gamma' \vdash \varphi$.

($\in$) If $\varphi \in \Gamma$, then $\Gamma \vdash \varphi$.

($\in_+$) If $\Gamma \vdash \varphi$ and $\Gamma \subseteq \Gamma'$ then $\Gamma' \vdash \varphi$.

($\overline{\mathbf{MP}}$) If $\Gamma_1 \vdash \varphi$ and $\Gamma_2 \vdash \varphi \to \psi$, and $\Gamma_1, \Gamma_2 \subseteq \Gamma$, then $\Gamma \vdash \psi$.

Definition 1.3.3 (Syntactically Equivalence)**.** *If* $\varphi \vdash \psi$ *and* $\psi \vdash \varphi$*, we say that* φ *and* ψ *are* syntactically equivalent*.*

Example 1.3.4. *(Identity):* $\vdash \varphi \to \varphi$

Solution. 1. $\varphi \to (\varphi \to \varphi)$

2. $\varphi \to ((\varphi \to \varphi) \to \varphi)$

3. $(\varphi \to ((\varphi \to \varphi) \to \varphi)) \to ((\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi))$

4. $(\varphi \to (\varphi \to \varphi)) \to (\varphi \to \varphi)$

5. $\varphi \to \varphi$

Example 1.3.5. *($\to_-$): If* $\Gamma \vdash \varphi \to \psi$ *then* $\Gamma, \varphi \vdash \psi$*.*

Solution. A simple application of $\overline{\text{MP}}$ and ($\in$).

Example 1.3.6. *($\to_+$): If* $\Gamma, \varphi \vdash \psi$ *then* $\Gamma \vdash \varphi \to \psi$*.*

Solution. By induction of the deductive sequence of $\Gamma, \varphi \vdash \psi$.

Example 1.3.7. (τ)*−transitivity: If* $\Gamma \vdash \varphi \to \psi$ *and* $\Gamma \vdash \psi \to \eta$*, then* $\Gamma \vdash \varphi \to \eta$*.*

Solution. By ($\to_-$), ($\to_+$) and ($\in_+$).

Example 1.3.8. *(Absurb):* $\vdash \neg\varphi \to (\varphi \to \psi)$*.*

Solution. 1. $\vdash \neg\varphi \to (\neg\psi \to \neg\varphi)$

2. $\vdash (\neg\psi \to \neg\varphi) \to (\varphi \to \psi)$

3. $\vdash \neg\varphi \to (\varphi \to \psi)$

Example 1.3.9. *(Absurb'):* $\vdash \varphi \to (\neg\varphi \to \psi)$

Example 1.3.10. *$(\neg_w)$:* $\neg\varphi \to \varphi \vdash \varphi$

Solution. 1. $\neg\varphi \to \varphi \vdash \neg\varphi \to \varphi$

2. $\vdash \neg\varphi \to (\varphi \to \neg(\neg\varphi \to \varphi))$

3. $\vdash (\neg\varphi \to (\varphi \to \neg(\neg\varphi \to \varphi))) \to ((\neg\varphi \to \varphi) \to (\neg\varphi \to \neg(\neg\varphi \to \varphi)))$

4. $\neg\varphi \to \varphi \vdash \neg\varphi \to \neg(\neg\varphi \to \varphi)$

5. $\vdash (\neg\varphi \to \neg(\neg\varphi \to \varphi)) \to ((\neg\varphi \to \varphi) \to \varphi)$

6. $\neg\varphi \to \varphi \vdash (\neg\varphi \to \varphi) \to \varphi$

7. $\neg\varphi \to \varphi \vdash \varphi$

Example 1.3.11. *$(\neg\neg_{-})$:* $\neg\neg\varphi \vdash \varphi$

Solution. 1. $\vdash \neg\neg\varphi \to (\neg\varphi \to \varphi)$

2. $\vdash (\neg\varphi \to \varphi) \to \varphi$

3. $\vdash \neg\neg\varphi \to \varphi$

4. $\neg\neg\varphi \vdash \varphi$

Example 1.3.12. *$(\neg_s)$:* $\varphi \to \neg\varphi \vdash \neg\varphi$

Solution. 1. $\neg\neg\varphi \vdash \varphi$

2. $\varphi \to \neg\varphi \vdash \varphi \to \neg\varphi$

3. $\varphi \to \neg\varphi, \neg\neg\varphi \vdash \varphi$

4. $\varphi \to \neg\varphi \vdash \neg\neg\varphi \to \neg\varphi$

5. $\vdash (\neg\neg\varphi \to \neg\varphi) \to \neg\varphi$

6. $\varphi \to \neg\varphi \vdash \neg\varphi$

Example 1.3.13. *($\neg\neg_+$):* $\varphi \vdash \neg\neg\varphi$

Solution. 1. $\vdash \varphi \to (\neg\varphi \to \neg\neg\varphi)$

2. $\vdash (\neg\varphi \to \neg\neg\varphi) \to \neg\neg\varphi$

3. $\vdash \varphi \to \neg\neg\varphi$

4. $\varphi \vdash \neg\neg\varphi$

Example 1.3.14. (R0) $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$

(R1) $\varphi \to \neg\psi \vdash \psi \to \neg\varphi$

(R2) $\neg\varphi \to \psi \vdash \neg\psi \to \varphi$

(R3) $\neg\varphi \to \neg\psi \vdash \psi \to \varphi$

Solution. 1. $\varphi \to \psi, \neg\neg\varphi \vdash \varphi$

2. $\varphi \to \psi, \neg\neg\varphi \vdash \psi$

3. $\vdash \psi \to \neg\neg\psi$

4. $\vdash (\neg\neg\varphi \to \neg\neg\psi) \to (\neg\psi \to \neg\varphi)$

5. $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$

1.3.5 Semantics

Semantics of a formula φ is given w.r.t. an assignment $\sigma \in 2^{AP}$, which is a subset of AP. Intuitively, it assigns true (or, $\mathbf{T}$) to propositions belonging to it, and assigns false (or, $\mathbf{F}$) to others. Thus, it can also be viewed as a function from AP to $\{\mathbf{T}, \mathbf{F}\}$.

Definition 1.3.15 (Semantics)**.** *Inductively, we may define the relation $\Vdash$ as follows:*

- $\sigma \Vdash \mathbf{T}$ *for all* σ*.*
- $\sigma \Vdash \mathbf{F}$ *for no* σ*.*
- $\sigma \Vdash p$ *iff* $p \in \sigma$*.*
- $\sigma \Vdash \neg\varphi$ *iff not* $\sigma \Vdash \varphi$ *(denoted by* $\sigma \nVdash \varphi$*).*
- $\sigma \Vdash \varphi \to \psi$ *iff either* $\sigma \nVdash \varphi$ *or* $\sigma \Vdash \psi$*.*
- $\sigma \Vdash \varphi \vee \psi$ *iff either* $\sigma \Vdash \varphi$ *or* $\sigma \Vdash \psi$*.*
- $\sigma \Vdash \varphi \wedge \psi$ *iff* $\sigma \Vdash \varphi$ *and* $\sigma \Vdash \psi$*.*

where

- $\varphi \vee \psi \triangleq \neg\varphi \to \psi$
- $\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi)$
- $\mathbf{T} \triangleq p \vee \neg p$ *for some* $p \in AP$
- $\mathbf{F} \triangleq \neg\mathbf{T}$

For a formula set Γ and an assignment σ, the *satisfaction relation* $\Vdash$ is defined by: $\sigma \Vdash \Gamma$ iff $\sigma \Vdash \varphi$ for every $\varphi \in \Gamma$. Observe $\sigma \Vdash \emptyset$ always holds.

1.3.6 Propositional Equivalences

The formula φ is called a *tautology* if $\sigma \Vdash \varphi$ for all assignment, it is *satisfiable* if $\sigma \Vdash \varphi$ for some assignment. Intuitively, φ is a

- *tautology* if it is always true, no matter what the truth values of the propositional variables are;
- *satisfiable* if it is satisfied by some assignment.

Moreover, φ is *unsatisfiable* if it is not satisfiable.

We say φ is a logical consequent of Γ, denoted as $\Gamma \models \varphi$, if $\sigma \Vdash \Gamma$ implies $\sigma \Vdash \varphi$ for each assignment σ.

Thus, φ is a tautology if φ is the logical consequent of $\emptyset$, denoted as $\models \varphi$.

Theorem 1.3.16 (The main theorem)**.** *With Hilbert's axiom system, we have that $\Gamma \vdash \varphi$ iff $\Gamma \models \varphi$.*

Prove the axioms are tautologies.

Formulas φ and ψ are called *logically (semantically) equivalent* if $\varphi \leftrightarrow \psi$ is a tautology. This is denoted by $\varphi \equiv \psi$.

Lemma 1.3.17. *φ and ψ syntactically equivalent iff they are logically equivalent.*

Definition 1.3.18 (Logical Equivalence)**.** *Show the following logical equivalences:*

1. *Identity laws:*

   $\varphi \wedge \mathbf{T} \equiv \varphi$, $\varphi \vee \mathbf{F} \equiv \varphi$

2. *Dominations laws*

   $\varphi \vee \mathbf{T} \equiv \mathbf{T}$, $\varphi \wedge \mathbf{F} \equiv \mathbf{F}$

3. *Idenpotent laws*

   $\varphi \vee \varphi \equiv \varphi, \varphi \wedge \varphi \equiv \varphi$

4. *Double negation law*

   $\neg(\neg\varphi) \equiv \varphi$

5. *Commutative laws*

   $\varphi \vee \psi \equiv \psi \vee \varphi, \varphi \wedge \psi \equiv \psi \wedge \varphi$

6. *Associative laws*

   $(\varphi_1 \vee \varphi_2) \vee \varphi_3 \equiv \varphi_1 \vee (\varphi_2 \vee \varphi_3), (\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \equiv \varphi_1 \wedge (\varphi_2 \wedge \varphi_3)$

7. *Distributive laws*

   $\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3),$
   $\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$

8. *De Morgan's laws*

   $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi, \neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$

9. *Absorption laws*

   $\varphi \vee (\varphi \wedge \psi) \equiv \varphi, \varphi \wedge (\varphi \vee \psi) \equiv \varphi$

10. *Negation laws*

    $\varphi \vee \neg\varphi \equiv \mathbf{T}, \varphi \wedge \neg\varphi \equiv \mathbf{F}$

Definition 1.3.19 (Logical Equivalence)**.** *Logical equivalences involving conditional statements:*

1. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$

2. $\varphi \rightarrow \psi \equiv \neg\psi \rightarrow \neg\varphi$

3. $\varphi \vee \psi \equiv \neg\varphi \rightarrow \psi$

4. $\varphi \wedge \psi \equiv \neg(\varphi \to \neg\psi)$
5. $\varphi \wedge \neg\psi \equiv \neg(\varphi \to \psi)$
6. $(\varphi_1 \to \varphi_2) \wedge (\varphi_1 \to \varphi_3) \equiv \varphi_1 \to (\varphi_2 \wedge \varphi_3)$
7. $(\varphi_1 \to \varphi_3) \wedge (\varphi_2 \to \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \to \varphi_3$
8. $(\varphi_1 \to \varphi_2) \vee (\varphi_1 \to \varphi_3) \equiv \varphi_1 \to (\varphi_2 \vee \varphi_3)$
9. $(\varphi_1 \to \varphi_3) \vee (\varphi_2 \to \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \to \varphi_3$
10. $\varphi \leftrightarrow \psi \equiv \neg\varphi \leftrightarrow \neg\psi$
11. $\varphi \leftrightarrow \psi \equiv (\varphi \wedge \psi) \vee (\neg\varphi \wedge \neg\psi)$
12. $\neg(\varphi \leftrightarrow \psi) \equiv \varphi \leftrightarrow \neg\psi$

Definition 1.3.20 (Applications of Satisfiability)**.** *Encode the Sudoku puzzle:*

- *For each $i, j, n \in \{1, \ldots, 9\}$, $p(i, j, n)$ is true when the number in the i-th row, j-th column is n.*
- *Every row contains every number:*

$$\bigwedge_{i=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{j=1}^{9} p(i, j, n)$$

- *Every column contains every number:*

$$\bigwedge_{j=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{i=1}^{9} p(i, j, n)$$

- *Each of the 3×3 blocks contains every number:*

$$\bigwedge_{r=0}^{2} \bigwedge_{s=0}^{2} \bigwedge_{n=1}^{9} \bigvee_{i=1}^{3} \bigvee_{j=1}^{3} p(3r + i, 3s + j, n)$$

- *No cell contains more than one number:*

$$\bigwedge_{i=0}^{9}\bigwedge_{j=0}^{9}\bigwedge_{n=0}^{9}\bigwedge_{n'=0}^{9}(n \neq n') \to (p(i,j,n) \to \neg p(i,j,n'))$$

1.3.7 The Axiom System: Soundness

Theorem 1.3.21 (Soundness)**.** *With Hilbert's axiom system, we have that* $\Gamma \vdash \varphi$ *implies* $\Gamma \models \varphi$*.*

Proof. By induction of the length of deductive sequence of $\Gamma \vdash \varphi$. □

Corollary 1.3.22. *If* $\vdash \varphi$*, then* $\models \varphi$*.*

1.3.8 The Axiom System: Completeness

Definition 1.3.23 (Consistency)**.** *We say a formula set* Γ *is consistent, iff there is some* φ *such that* $\Gamma \nvdash \varphi$*.*

Lemma 1.3.24. Γ *is consistent iff for each* φ*, either* $\Gamma \nvdash \varphi$ *or* $\Gamma \nvdash \neg\varphi$*.*

Proof. Exploit $\neg\varphi, \varphi \vdash \psi$. □

Lemma 1.3.25. $\Gamma \cup \{\varphi\}$ *is consistent iff* $\Gamma \nvdash \neg\varphi$*.*

Proof. Suppose that $\Gamma \nvdash \neg\varphi$ and $\Gamma \cup \{\varphi\}$ is inconsistent, then we have $\Gamma, \varphi \vdash \neg\varphi$ hence $\Gamma \vdash \varphi \to \neg\varphi$. Recall that we have $\varphi \to \neg\varphi \vdash \neg\varphi$, and this implies $\Gamma \vdash \neg\varphi$, contradiction! The other direction is easy. □

Lemma 1.3.26. *If the formula set* Γ *is inconsistent, then it has some finite inconsistent subset* Δ*.*

Theorem 1.3.27. Γ *is consistent iff* Γ *is satisfiable.*

Proof. The "if" direction is easy: suppose that $\sigma \Vdash \Gamma$ but $\Gamma \vdash \varphi$ and $\Gamma \vdash \neg\varphi$, then $\sigma \Vdash \varphi$ and $\sigma \Vdash \neg\varphi$, contradiction.

For the "only if" direction, let us enumerate all propositional formulas as following (note the cardinality of all such formulas is $\aleph_0$):

$$\varphi_0, \varphi_1, \ldots, \varphi_n, \cdots$$

Let $\Gamma_0 = \Gamma$ and

$$\Gamma_{i+1} = \begin{cases} \Gamma_i \cup \{\varphi_i\} & \text{if } \Gamma_i \nvdash \neg\varphi_i \\ \Gamma_i \cup \{\neg\varphi_i\} & \text{otherwise} \end{cases}$$

and finally let $\Gamma^* = \lim\limits_{i\to\infty} \Gamma_i$.

The formula set Γ^* has the following properties:

(a) *Each* Γ_{i+1} *is consistent, and* Γ^* *is also consistent.*

- Assume Γ_i is consistent. If $\Gamma_i \nvdash \neg\varphi_i$, by Lemma 1.3.25, Γ_{i+1} is consistent. Otherwise, $\Gamma_i \vdash \neg\varphi_i$. Note there exits η with $\Gamma_i \nvdash \eta$, and obviously $\Gamma_i \cup \{\neg\varphi_i\} \nvdash \eta$.
- Assume Γ^* is not consistent, there exists a finite set $\Delta \subseteq \Gamma^*$ which is inconsistent. We can always find an index i such that $\Delta \subseteq \Gamma_i$, implying that Γ_i is inconsistent, contradiction.

(b) Γ^* is a maximal set, i.e., for each formula φ, either $\varphi \in \Gamma^*$ or $\neg\varphi \in \Gamma^*$.

(c) For each formula φ, we have $\Gamma^* \models \varphi$ iff $\varphi \in \Gamma^*$.

- If $\varphi \in \Gamma^*$, then $\Gamma^* \vdash \varphi$, then $\Gamma^* \models \varphi$ by the soundness result.

- If $\Gamma^* \models \varphi$, assume $\varphi \notin \Gamma^*$. By maximality, $\neg\varphi \in \Gamma^*$, then $\Gamma^* \vdash \neg\varphi$, then $\Gamma^* \models \neg\varphi$ by the soundness result, contradiciton.

Let $\sigma = \Gamma^* \cap AP$, then, we prove $\sigma \Vdash \Gamma^*$. We prove $\sigma \Vdash \varphi$ iff $\varphi \in \Gamma^*$. It follows by structural induction over φ:

- The base case $\varphi = p \in AP$ is easy.
- Let $\varphi = \neg\psi$. Then, $\psi \notin \Gamma^*$. By induction hypothesis, $\psi \in \Gamma^*$ iff $\sigma \Vdash \psi$.
- Let $\varphi = \psi \to \eta$. Exercise!

□

Theorem 1.3.28 (Completeness)**.** *If* $\Gamma \models \varphi$*, then* $\Gamma \vdash \varphi$*.*

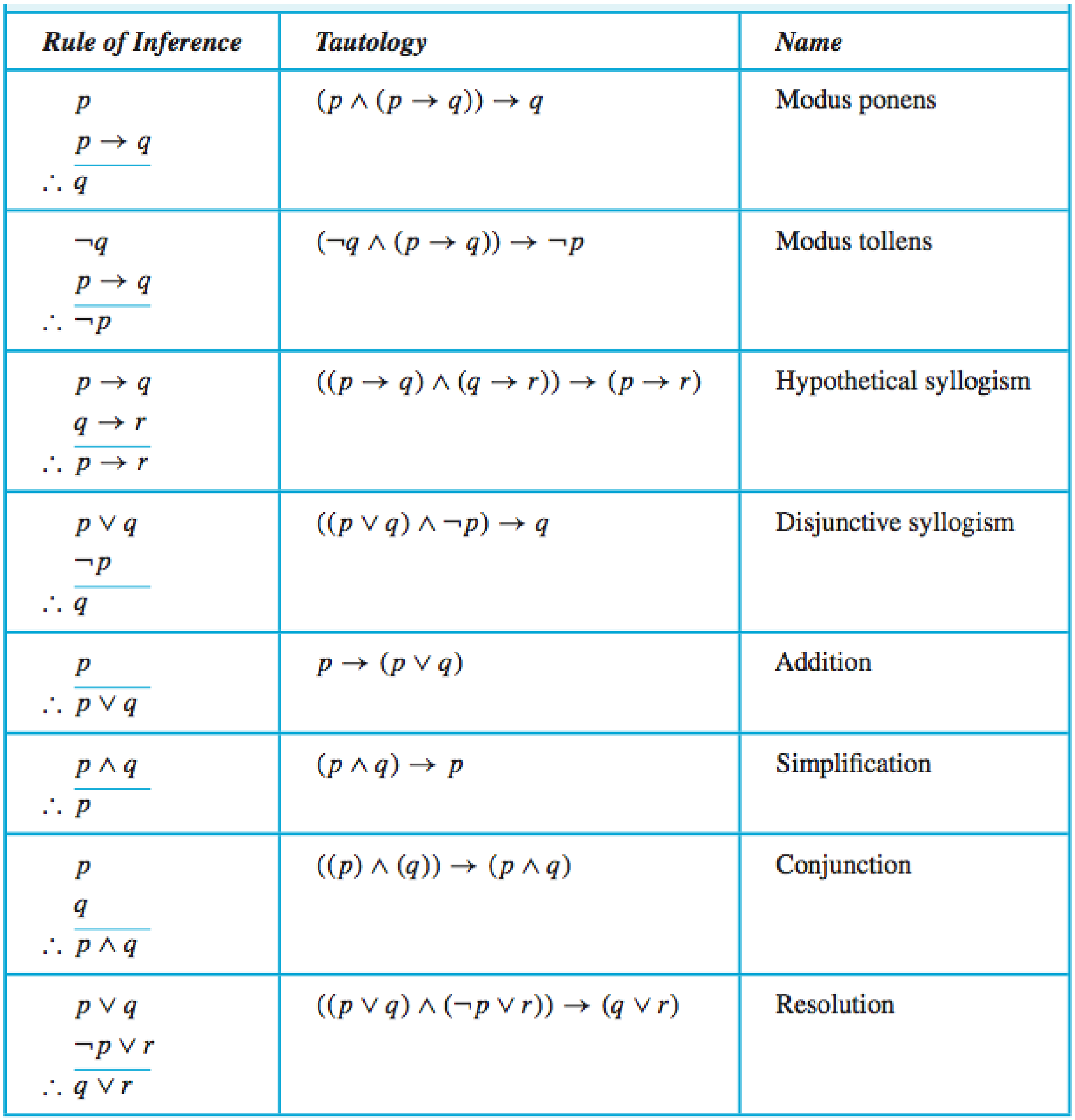
Proof. Assume by contradiction that $\Gamma \nvdash \varphi$, then $\Gamma \cup \{\neg\varphi\}$ is consistent. Thus it is satisfiable, and there is an assignment σ such that $\sigma \Vdash \Gamma \cup \{\neg\varphi\}$. However, this implies that $\sigma \Vdash \Gamma$ and $\sigma \nVdash \varphi$, which violates the assumption $\Gamma \models \varphi$. □

Corollary 1.3.29. $\models \varphi$ *implies that* $\vdash \varphi$*.*

Theorem 1.3.30 (Compactness)**.** *Given a formula set* Γ*, we have*

1. Γ *is consistent iff each of its finite subsets is consistent;*
2. Γ *is satisfiable iff each of its finite subsets is satisfiable.*

Rules of Inference for Propositional Logic (cf. page 72):



Rule of Inference	*Tautology*	*Name*
p $p \to q$ $\therefore q$	$(p \land (p \to q)) \to q$	Modus ponens
$\neg q$ $p \to q$ $\therefore \neg p$	$(\neg q \land (p \to q)) \to \neg p$	Modus tollens
$p \to q$ $q \to r$ $\therefore p \to r$	$((p \to q) \land (q \to r)) \to (p \to r)$	Hypothetical syllogism
$p \lor q$ $\neg p$ $\therefore q$	$((p \lor q) \land \neg p) \to q$	Disjunctive syllogism
p $\therefore p \lor q$	$p \to (p \lor q)$	Addition
$p \land q$ $\therefore p$	$(p \land q) \to p$	Simplification
p q $\therefore p \land q$	$((p) \land (q)) \to (p \land q)$	Conjunction
$p \lor q$ $\neg p \lor r$ $\therefore q \lor r$	$((p \lor q) \land (\neg p \lor r)) \to (q \lor r)$	Resolution

1.3.9 Satisfiability

For propositional logic, we consider two important (decision) problems:

1. The SATISFIABILITY problem. i.e, given a formula φ, to decide that if φ is satisfiable.
2. The VADILITY problem. i.e., give a formula φ, to decide that if φ is a tautology. We say it is *valid* in this case.

Notably, these two problems are closely related:

- φ is satisfiable iff $\neg\varphi$ is not valid.
- φ is valid iff $\neg\varphi$ is not satisfiable.

Instead of the canonical VALIDITY problem, we are sometimes concerns about whether $\Gamma \models \varphi$, equivalently, whether $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable. The validity problem can then be considered as the special case of $\Gamma = \emptyset$.

Remark 1.3.31. *At least, we have a naïve approach to test all assignments restricted to propositions occurring in φ.*

However, such approach is usually inefficient, as the set of assignments is exponential in $|AP|$.

We now introduce two other classical approaches respectively for satisfiability and validity checking — i.e., the tableau approach and the resolution approach.

Tableau Approach

The main idea of tableau approach is to eventually "decompose" the formula into a set of literals, and finally perform a local satisfiability checking.

Central part of this approach is a set of rewriting rules.

Example 1.3.32. *Tableau Rules*

$$\frac{\Gamma, \neg\neg\varphi}{\Gamma, \varphi}\ (\neg\neg) \qquad \frac{\Gamma, \varphi \to \psi}{\Gamma, \neg\varphi}\ (\to_l)$$

$$\frac{\Gamma, \varphi \to \psi}{\Gamma, \psi}\ (\to_r) \qquad \frac{\Gamma, \neg(\varphi \to \psi)}{\Gamma, \varphi, \neg\psi}\ (\neg \to)$$

Definition 1.3.33 (Tableau)**.** *Given a formula φ, a tableau of φ is a series of formula set Γ_0, Γ_1,..., Γ_n, where:*

- $\Gamma_0 = \{\varphi\}$.
- *Each Γ_{i+1} is obtained from Γ_i by applying some tableau rule.*
- *Γ_n consists of only literals.*

A tableau is consistent if its last formula set contains no conflicting literals.

Theorem 1.3.34. *A formula φ is satisfiable iff it has a consistent tableau.*

Example 1.3.35. *Suppose that $\varphi = (p \to \neg p) \to p$, then we have the following tableau for φ: $\{(p \to \neg p) \to p\}$, $\{\neg(p \to \neg p)\}$, $\{p, \neg\neg p\}$, $\{p\}$. Hence φ is satisfiable.*

Example 1.3.36. *Suppose that $\psi = \neg((\neg p \to p) \to p)$, then we have two possible tableaux: $\{\neg((\neg p \to p) \to p)\}$, $\{\neg p \to p, \neg p\}$, $\{\neg\neg p, \neg p\}$, $\{p, \neg p\}$, $\{\neg((\neg p \to p) \to p)\}$, $\{\neg p \to p, \neg p\}$, $\{p, \neg p\}$ — none all of is consistent, and hence ψ is not satisfiable.*

Resolution Approach

We first transform the formula φ into CNF. A CNF formula φ can also be seen as a set of clauses, and a clause is a set of literals.

Remark 1.3.37. *The empty clause, denoted by $\Box$, is unsatisfiable by definition. The empty formula describes an empty set of clauses and is satisfiable by definition.*

We allow set operations on clauses as expected.

Definition 1.3.38 (Resolution)**.** *Let C_1 and C_2 be two clauses and L) be a literal with the following property: $L \in C_1$ and $\neg L \in C_2$. Then one can compute the clause*

$$R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\neg L\})$$

that is denoted as the resolvent *of the clauses C_1 and C_2 over L.*

Lemma 1.3.39. *Let φ be a CNF formula and R be the resolvent of two clauses C_1 and C_2 from φ. Then φ and $\varphi \cup \{R\}$ are equivalent.*

Define $Res(\varphi)$ the formula $\varphi \cup \{R\}$ where R is a resolvant of two clauses in φ. Let $Res^0(\varphi) = \varphi$, and $Res^{i+1}(\varphi) = Res(Res^i(\varphi))$. Let $Res^*(\varphi) = \lim_i Res^i(\varphi)$. Then, we have:

Theorem 1.3.40. *A CNF formula φ is unsatisfiable iff $\Box \in Res^*(\varphi)$.*

The resolution based algorithm. Construct $Res^*(\varphi)$:

- if for some $i > 0$: $\Box \in Res^i(\varphi)$, φ is unsatisfiable.
- if for some $\Box \notin Res^i(\varphi) = Res^{i+1}(\varphi)$, φ is satisfiable.

Complexity of this naive method. Since in a clause a variable occurs either as a positive literal, or negative literal, or it does not occur at all, for a formula having n variables the runtime and memory consumption lie in the order of 3^n in the worst case.

DPLL

The SAT-algorithm, which was proposed in 1960 by Davis and Putnam, is based on the elimination method and uses a couple of optimizations:

- Subsumption checks
- *Pure literal* detection: literal occurring only positive or only negative. If there is a pure literal l in Γ, then remove all clauses containing l.
- Variable elimination (by adding all resolvent clauses)

The optimizations improve the runtime behavior in practice, but not the worst case complexity of the naive method. To further improve the efficiency, Davis, Putnam, Logemann and Loveland proposed in 1962 the following process to accelerate the resolution process of Γ, and it consist of four rules.

1. Tautology rule: remove all tautologies from Γ.
2. Single literal rule: if there is some $l \in \Gamma$ and l is a literal, then remove all clauses containing l, and delete all complementary literals occurring in the rest clauses.
3. Pure literal detection: if there is a pure literal l in Γ, then remove all clauses containing l.
4. Splitting-rule Suppose that

   $$\Gamma = \{C_1 \vee p, \ldots, C_n \vee p, C'_1 \vee \neg p, \ldots C'_m \vee \neg p, D_1, \ldots, D_k\}$$

   then split Γ into

   $$\Gamma' = \{C_1, \ldots, C_n, D_1, \ldots D_k\}$$

and

$$\Gamma'' = \{C'_1, \ldots, C'_m, D_1, \ldots D_k\}$$

Γ is satisfiable iff Γ' or Γ'' is satisfiable.

Example 1.3.41. *Suppose that* $\Gamma = \{p, p \to q, q \to r, \neg(p \to r)\}$*. Then we first normalize the set as* $\{p, \neg p \vee q, \neg q \vee r, \neg r\}$.

Further Reading.

- A. Biere, M. J. H. Heule, H. van Maaren, T. Walsh: Handbook of Satisfiability, IOS Press, 2009

- The International Conferences on Theory and Applications of Satisfiability Testing (SAT): `http://www.satisfiability.org/`

- Tons of workshop, conference, and journal papers