1 Propositional Logic

Basic Definition of Proposition

- ▶ A declarative sentence (陈述句) is a sentence that "declares" a fact or some facts.
- ▶ A **proposition (命题)** should be a declarative sentence for which we can determine whether it is true or false.

Definition: Propositions

A proposition is a **declarative** sentence/assertion that is **either true or false** (but not both).

- ▶ Based on the above definition, we know that a proposition has the following properties:

1. First, it has to be declarative. In other words, imperative (祈使), interrogative (疑问) and exclamatory (感叹) sentences are not propositions.

Examples

- – The followings are propositions:
  "*SJTU is a good.*" "*SJTU is a bad.*" "*Run rises in east.*"
- – The followings are NOT propositions:
  "*Make American great again!*" "*Are you crazy?*" "*Great!*"

2. Second, we must be able to determine its **true value (真值)**.

Examples

- – The followings are propositions:
  - "*I am a SJTUer.*": This is course true.
  - "$1 + 1 = 10$.": This is false in decimal system but true in binary system. But in any case, it can only be either true or false under a given system.
  - "*Any even number greater than* 2 *is the sum of two primes.*": This is the well-known Goldbach's conjecture. We do not know yet whether it is true or false, but the truth is out there!
- – The followings are NOT propositions:
  "*This sentence is false.*" "*I am lying.*"
  Because if they are true, then it implies that them are false, and vice versa. Therefore, we cannot find truth values for them.

- ▶ We usually use "T" or "1" to represent "true" and use "F" or "0" to represent "false".

Atomic Propositions and Compound Propositions

- A proposition can be very complicated in the sense that it is composed by a set of propositions and logical operators. For example, we have
  - If snow is black, then $1 + 1 = 3$.
  - Either I am right or you are right, but none of them is right.
- Those propositions that cannot be further decomposed, such as "snow is black" or "$1+1 = 3$", are called **atomic propositions (原子命题)**.
- Those propositions composed by multiple atomic propositions and logical operators, such as "*if snow is black, then* $1 + 1 = 3$", are called **compound propositions (复合命题).**
- In propositional logic, we are not interested in determining whether a given specific proposition is true or false. Instead, we are interested in some general structural properties like "*if something is true, then we must know that something else is false*". Therefore, we usually use symbols $P, Q, R, \ldots$ to represents generic (atomic) propositions. For example, if we use P to represent "*snow is black*" and use Q to represent "$1 + 1 = 3$", then "*if snow is black, then* $1 + 1 = 3$" becomes $P \to Q$.
- However, in general, $P, Q, R, \ldots$ can be arbitrary propositions like variables $x, y, z, \ldots$ in functions. Therefore, those undetermined (atomic) propositions $P, Q, R, \ldots$ are called (atomic) **propositional variables (命题变元)**.
- By putting propositional variables together using logical operators, we obtain a **compound proposition formula (复合命题公式)** such as $P \wedge Q \to R$, $\neg P \vee Q$. Hereafter, we will introduce several commonly used logical operators such as "not $\neg$", "and $\wedge$", "or $\vee$" and "if-then $\to$".
- Roughly speaking, when we write P, it is a propositional variable not a proposition because it has no specific meaning for true or false. Similarly, when we write $P \wedge Q$, it is a compound proposition formula not a compound proposition.

  For the sake of simplicity, however, we usually just call P a proposition and call $P \wedge Q$ a compound proposition, without mentioning "variable" or "formula". Or even just call both of them a proposition. But we should know what are they exactly.

Negation

- The first logical operator we study is called **negation (否定)** defined as follows:

Definition: Negation

Let P be a proposition. Then $\neg P$ is a new compound proposition such that it is true iff P is false, where "iff" standards for "if and only if".

- For example, if P denotes "$1+1=2$", then $\neg P$ is "$1+1\neq 2$". You cannot say that $\neg P$ is "$1+1=3$", which is different.

Truth Table

Based on the above definition, we can draw its **truth table (真值表)**, which shows how the truth value of the compound proposition changes when the truth values of the atomic propositions in it change.

P	$\neg P$
0	1
1	0

Conjunction and Disjunction

- Given two propositions, **conjunction (合取) "$\wedge$"** is used to represent "and", and **disjunction (析取) "$\vee$"** is used to represent "or". They are defined as follows.

Definition: Conjunction and Disjunction

Let P and Q be two propositions. Then

- $P \wedge Q$ is a new proposition such that it is true iff both P and Q are true;
- $P \vee Q$ is a new proposition such that it is false iff both P and Q are false.

According to the above definition, we also know that

- $P \wedge Q$ is false iff either P or Q is false
- $P \vee Q$ is true iff either P or Q is true

We can also draw their truth tables.

P	Q	$P \wedge Q$	$P \vee Q$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

- We should be careful that our natural language "and/or" are not exactly the same as "$\wedge/\vee$". For example:
  - "*This machine is good, but expensive.*". Here, "but" also means "$\wedge$";
  - "*You die or I die!*". Here "or" does not mean "$\vee$", it means **exclusive or (异或)** denoted by $P \oplus Q$. Specifically, $P \oplus Q$ is true iff the truth values of P and Q are different, e.g., $0 \oplus 0 = 0$ and $1 \oplus 0 = 1$.

Example: Half Adder

Half adder (半加器) is a very basic component in digital circuits. It adds two single binary digits, A and B and has two outputs, sum S and carry (进位) C. The carry signal represents an overflow into the next digit of a multi-digit addition. We call it half adder because it does not consider the carry from the previous stage.

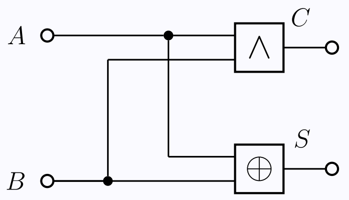
Formally, the half adder should have the functionality described by the following table:

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

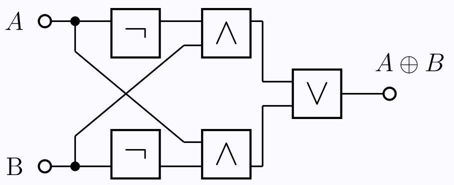
By looking at the table, we find the following relation between A, B and C, S:

$$S = A \oplus B = (\neg A \wedge B) \vee (A \wedge \neg B)$$
$$C = A \wedge B$$

We can design a half adder as follows



Circuits for the half adder.



Circuits for $A \oplus B$.

Implication

▶ The last logical operator we study is called **implication (蕴含)** defined as follows:

Definition: Implication

Let P and Q be two propositions. Then $P \to Q$ is a new compound proposition such that <u>it is false iff P is true and Q is false</u>.

▶ Formula $P \to Q$ can be read as "if P, then Q". To understand this, we need to consider $P \to Q$ as a whole. Namely, we do not care about whether P is true or false. What we are really interested in is <u>whether the implication relation from P to Q is true or false</u>.

- By definition, $P \to Q$ is always true when P is false.
- Let P be "*sun rises in west*" and Q be "*DM has no exam*". Then $P \to Q$ is true because the premise is already false. What I am promising "<u>if</u> P, <u>then</u> Q". Since the premise does not hold, you cannot say that I violate my promise.

P	Q	$P \to Q$
0	0	1
0	1	1
1	0	0
1	1	1

▶ Similarly, we can also define the **biconditional statement (双条件)** $P \leftrightarrow Q$ such that <u>it is true iff the truth values of P and Q are the same</u>.

Well-Formed Formula

▶ Now we have a set of atomic proposition variables $P, Q, R, \dots$ and a set of logical operators $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \dots$. To obtain more complex compound proposition formulas, we need to put them together correctly in a meaningful way such as $P \wedge Q$ or $\neg P \rightarrow Q$. However, putting them together arbitrarily may have no sense. Therefore, we need to define the syntax (语法) for writing compound proposition formulas using proposition variables and logical operators. Such a correct way for writing compound proposition formulas is called the **Well-Formed Formulas (合式公式)** defined as follows.

Definition: Well-Formed Formula (WFF)

Well-formed formulas are defined inductively as follows:

1. Each propositional variable is a WFF.
2. If α is a formula, then $\neg\alpha$ is a WFF.
3. If α and β are WFFs, and $\bullet$ is any binary logical operator, then $(\alpha \bullet \beta)$ is a WFF, where $\bullet$ could be (not limited to) the usual operators $\neg, \wedge, \vee, \rightarrow$ or $\leftrightarrow$.

▶ **Example:** According to the above definition, the followings are WFFs:

$$(\neg(P \wedge Q)), \quad ((\neg(\neg P)) \rightarrow (P \wedge Q)), \quad ((P \rightarrow Q) \rightarrow (\neg R)).$$

However, the followings are Not WFFs:

$$(PQ \rightarrow R), \quad \neg(\neg P), \quad (\vee Q \rightarrow (\neg P)).$$

▶ Note that, WFFs are just telling us how to write formulas in a meaningful way without ambiguity. Sometimes, we can take some simplifications when its meaning is clear.

Some Simplification Rules for WFFs

1. Outside $(\cdot)$ can always be omitted, e.g., $(P \rightarrow Q)$ can be written as $P \rightarrow Q$.
2. $(\neg P)$ can be simplified as $\neg P$, e.g., $(\neg(Q \wedge (\neg P)))$ can be written as $\neg(Q \rightarrow \neg P)$.
3. We can always omit $(\cdot)$ by understanding the precedence (优先级) of the operators. Commonly, we assume precedence in following order $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and for the same operator from left to right. For example,
   $\star$ $(P \rightarrow (Q \vee R))$ can be simplified as $P \rightarrow (Q \vee R)$ or further as $P \rightarrow Q \vee R$.
   $\star$ $(P \rightarrow (P \rightarrow R))$ can be simplified as $P \rightarrow (P \rightarrow R)$, but NOT $P \rightarrow P \rightarrow R$

▶ In summary, WFF is just a syntax telling us how to write correctly. Of course, you can define your own WFF (if someone accepts it). There is an elegant way for defining this called the Backus–Naur form (BNF) as follows

$$\varphi ::= P \mid (\neg\varphi_1) \mid (\varphi_1 \wedge \varphi_2) \mid (\varphi_1 \vee \varphi_2) \mid (\varphi_1 \rightarrow \varphi_2) \mid (\varphi_1 \leftrightarrow \varphi_2)$$

Interpret Formulas via Assignments

- Now, we know how to write a compound proposition formula using the rules described by WFFs. Essentially, a WFF is a Boolean function $\alpha = f(P_1, P_2, \ldots, P_n)$, where each P_i is an atomic propositional variable.

- The truth value of formula α is determined by the **assignments (指派)** of $P_1, \ldots, P_n$. Specifically, a truth assignment is a vector $I = (v_1, \ldots, v_n)$, where each v_i is either 0 or 1. There are 2^n assignments in total and each one corresponds to a row in the truth table.

- Given a formula α and an assignment $I = (v_1, \ldots, v_n)$, we write $I \models \alpha$ if it is true under assignment I. For example, let us consider formula $\alpha = (P_1 \wedge P_2) \vee P_3$. Then
  - for $I_1 = (1, 1, 0)$, we have $I_1 \models \alpha$ because $(\mathsf{T} \wedge \mathsf{T}) \vee \mathsf{F} = \mathsf{T} \vee \mathsf{F} = \mathsf{T}$;
  - for $I_2 = (1, 0, 0)$, we have $I_2 \not\models \alpha$ because $(\mathsf{T} \wedge \mathsf{F}) \vee \mathsf{F} = \mathsf{F} \vee \mathsf{F} = \mathsf{F}$.

- In many cases, we are not just interested in whether or not proposition formula α is true or false under some specific assignment I. More generally, we are interested in some inherent property of a formula. For example, if $\alpha = f(P_1, \ldots, P_n)$ is always true no matter what truth value each P_i takes, then it means α characterizes some inherent relationship between each P_i. This is actually a theorem should be: some statement that is globally true no matter what instant you choose. This leads to the following definitions.

Definition: Tautology/Satisfiable/Contradiction

A compound proposition formula is said to be

- a **tautology (重言式)** if it is true under any assignment;
- **satisfiable (可满足的)** if it is true under some assignment;
- a **contradiction (矛盾式)** if it is false under any assignment.

- **Examples:** We use the following examples to illustrate the above definitions:
  1. $P \vee \neg P$ is a clearly a tautology because either P or $\neg P$ is true.
  2. $(P \rightarrow \neg P) \vee P$ is also a tautology because if P is false, then $P \rightarrow \neg P$ is true.
  3. $P \vee Q$ is not a tautology but is satisfiable, e.g., it is true under $I = (1, 0)$ or $(0, 1)$.
  4. $P \wedge \neg P$ is a contradiction because either P or $\neg P$ is false.

- Finally, we make the following observations:
  1. α is a tautology iff $\neg\alpha$ is a contradiction.
  2. α is a satisfiable iff $\neg\alpha$ is not a tautology.
  3. Tautologies connected by " $\vee$ "," $\wedge$ "," $\rightarrow$ "," $\leftrightarrow$ " are still tautologies.

Boolean Satisfiability Problem and Its Applications

- In logic and computer science, the Boolean satisfiability problem (SAT) is one of the most fundamentally important problems. Specifically, given a proposition formula $\alpha = f(P_1, \ldots, P_n)$ involving n atomic propositional variables, it asks us to find, if possible, an assignment I such that $I \models \alpha$.

- Clearly, this problem can be solve by testing all 2^n possible assignments. However, this is very inefficient (but avoidable because it has been proved to be NP-hard). Here we will not discuss how to solve SAT. Instead, we illustrate how it can model different problems.

- **Example 1: Logical Puzzles**

  Suppose there is an island where everyone is either a Knight or a Knave. A Knight always tells the truth, but a Knave always lies. Now, you meet two people Alice and Bob:

  - Alice says: "*Bob is a Knight*".
  - Bob says: "*Alice and I are different*".

  So what can you deduce based on the above information? We try to solve it by SAT.

  The key in solving problems using SAT is to first select the right atomic propositional variables and then find some of their relations that should be true. The specific truth values of the proposition variables you choose is actually your answer.

  Here we choose the following two atomic propositions

  - P: "Alice is a Knight".
  - Q: "Bob is a Knight".

  Then we draw the following truth tables

Variable 1	Variable 2	Alice Said	Bob Said	Island Rule
P	Q	Q	$P \oplus Q$	$(P \leftrightarrow Q) \wedge (Q \leftrightarrow (P \oplus Q))$
0	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	1	0	0

  To be consistent with the fact of the island, we need to find assignments for P and Q such that the Island Rule holds. The only possibility is the first row, where P and Q are both false.

  Therefore, we know that both Alice and Bob are Knaves!

Boolean Satisfiability Problem and Its Applications

▶ **Example 2: Sudoku Puzzles**

Sudoku (数独) is a logic-based number-placement puzzle. Its rules are as follows.

The objective of Sudoku is to fill a 9×9 grid with numbers so that each column, each row, and each 3×3 sub-grids that compose the grid contain all of the numbers from 1 to 9. Numbers in some grids have already been provided. What you need to do is to assign 1 to 9 to those blank grid such that the requirements are fulfilled.

	1	8				7		
						2		
	7							
				7	1			
6							4	
4			5					3
	2			8				
							6	

An example of Sudoku puzzle.

To solve Sudoku using SAT, still, first we need to choose right propositional variables. Here what we need to assign to each grid is 1 to 9, which is not a binary value. However, as long as our decision variables are discrete, we can encode them as propositional variables.

The idea is to choose $9 \times 9 \times 9$ variables $P_{i,j,n}$, where each $P_{i,j,n}$ means that "the number in grid (i, j) is n". The truth assignments for $P_{i,j,n}$ should satisfy the following constraints.

- First, if the number in grid $[i, j]$ is already given as n, then $P_{i,j,n}$ has to be true. Suppose that we have K grids whose numbers are already given and they are $S[i_k, j_k] = n_k, k = 1, \ldots, K$. Then we have
$$\alpha_1 = \bigwedge_{k=1,\ldots,K} P_{i_k,j_k,n_k}$$
- Second, for each grid, we can put at most one number
$$\alpha_2 = \bigwedge_{i=1,\cdots,9} \bigwedge_{j=1,\cdots,9} \bigwedge_{n=1,\cdots,8} \bigwedge_{n'=n+1,\cdots,9} \neg(P_{i,j,n} \bigwedge P_{i,j,n'})$$
- Third, for each row, each number has to occur somewhere
$$\alpha_3 = \bigwedge_{i=1,\cdots,9} \bigwedge_{n=1,\cdots,9} \bigvee_{j=1,\cdots,9} P_{i,j,n}$$
- Also, for each column, each number has to occur somewhere
$$\alpha_4 = \bigwedge_{j=1,\cdots,9} \bigwedge_{n=1,\cdots,9} \bigvee_{i=1,\cdots,9} P_{i,j,n}$$
- Finally, for each 3×3 sub-grid, each number has to occur somewhere
$$\alpha_5 = \bigwedge_{n=1}^{9} \bigwedge_{s=0}^{2} \bigwedge_{t=0}^{2} \bigvee_{i=1}^{3} \bigvee_{j=1}^{3} P_{3s+i,3t+j,n}$$

Putting all the above together, we need to find an assignment I such that $I \models \wedge_{i=1,2,3,4,5}\alpha_i$.

Logical Equivalence

- Given two functions $f(x, y, z)$ and $g(x, y, z)$, if their outputs are the same no matter what inputs for x, y, z are taken, then we consider them as the "same" function. This observation leads to the following definition.

Definition: Logical Equivalence

Let α and β be two compound proposition formulas over variables $P_1, \dots, P_n$. If the truth values of α and β are the same under any assignment, then α and β are said to be **logical equivalent (等值/逻辑等价)**, denoted by $\alpha \Leftrightarrow \beta$ or $\alpha = \beta$.

- **Examples:** We show that implication can actually be expressed by negation and disjunction, while disjunction can further be expressed by negation and conjunction.

① $P \to Q = \neg P \vee Q$

P	Q	$P \to Q$	$\neg P \vee Q$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

② $P \vee Q = \neg(\neg P \wedge \neg Q)$

P	Q	$P \vee Q$	$\neg(\neg P \wedge \neg Q)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

- We note that any tautologies are actually equivalent. Therefore, we use a special symbol **T** to represent an arbitrary tautology, which means "something always true", e.g.,

$$P \vee \neg P = P \to P = (P \to Q) \vee \neg P = \mathbf{T}$$

Similarly, any contradictions are also equivalent, and we use a special symbol **F** to represent an arbitrary contradiction, which means "something always false", e.g.,

$$P \wedge \neg P = (P \to \neg P) \wedge P = (P \to Q) \wedge P \wedge \neg Q = \mathbf{F}$$

- Finally, we remark that symbols "$\leftrightarrow$" and "$\Leftrightarrow$" are different. In particular,
  - $\leftrightarrow$ is a symbol in the formal system, i.e., $\alpha \leftrightarrow \beta$ is a formula (syntax).
  - $\Leftrightarrow$ is a symbol outside of the system, i.e., $\alpha \Leftrightarrow \beta$ is an interpretation (semantics).

Therefore, $\alpha \Leftrightarrow \beta$ asserts the equivalence but $\alpha \leftrightarrow \beta$ does not imply any relationship between α and β. Only when $\alpha \leftrightarrow \beta$ is a tautology, we have $\alpha = \beta$, which is stated as the following **logical equivalence theorem (等值定理)**

Theorem: Logical Equivalence Theorem

For any two propositions α, β, we have $\alpha = \beta$ iff $\alpha \leftrightarrow \beta$ is a tautology.

Proof: $\alpha \Leftrightarrow \beta$ iff for any assignment, the truth values of α and β are the same.
iff for any assignment, $\alpha \leftrightarrow \beta$ is true.
iff $\alpha \leftrightarrow \beta$ is a tautology.

Some Basic Logical Equivalence Rules

▶ The followings are some commonly used logical equivalence rules:

① Identity Laws (同一律)

$$P \wedge \mathbf{T} = P$$
$$P \vee \mathbf{F} = P$$

② Domination Laws (零率)

$$P \vee \mathbf{T} = \mathbf{T}$$
$$P \wedge \mathbf{F} = \mathbf{F}$$

③ Idempotent Laws (等幂律)

$$P \vee P = P$$
$$P \wedge P = P$$

④ Double Negation Law (双重否定)

$$\neg\neg P = P$$

⑤ Commutative Laws (交换律)

$$\begin{aligned} P \vee Q &= Q \vee P \\ P \wedge Q &= Q \wedge P \\ P \leftrightarrow Q &= Q \leftrightarrow P \end{aligned}$$

⑥ Associative Laws (结合律)

$$\begin{aligned} (P \vee Q) \vee R &= P \vee (Q \vee R) \\ (P \wedge Q) \wedge R &= P \wedge (Q \wedge R) \\ (P \leftrightarrow Q) \leftrightarrow R &= P \leftrightarrow (Q \leftrightarrow R) \end{aligned}$$

⑦ Distributive Laws (分配率)

$$\begin{aligned} P \vee (Q \wedge R) &= (P \vee Q) \wedge (P \vee R) \\ P \wedge (Q \vee R) &= (P \wedge Q) \vee (P \wedge R) \\ P \to (Q \to R) &= (P \to Q) \to (P \to R) \end{aligned}$$

⑧ De Morgan's Laws (德摩根律)

$$\begin{aligned} \neg(P \vee Q) &= \neg P \wedge \neg Q \\ \neg(P \wedge Q) &= \neg P \vee \neg Q \end{aligned}$$

⑨ Absorbtion Laws (吸收率)

$$\begin{aligned} P \vee (P \wedge Q) &= P \\ P \wedge (P \vee Q) &= P \end{aligned}$$

⑩ Negation Laws (补余律)

$$\begin{aligned} P \vee \neg P &= \mathbf{T} \\ P \wedge \neg P &= \mathbf{F} \end{aligned}$$

▶ Logical equivalence is always used for simplifying formulas. Specifically, if we **replace (置换)** a sub-formula in α by an equivalent one and obtain β, then we have $\alpha = \beta$. For example, we can prove $(\neg P \wedge (\neg Q \wedge R)) \vee (Q \wedge R) \vee (P \wedge R) = R$ as follows.

$$\begin{aligned} \text{LHS (Left Hand Side)} &= (\neg P \wedge (\neg Q \wedge R)) \vee ((Q \vee P) \wedge R) \\ &= ((\neg P \wedge \neg Q) \wedge R) \vee ((P \vee Q) \wedge R) \\ &= (\neg(P \vee Q) \wedge R) \vee ((P \vee Q) \wedge R) \\ &= (\neg(P \vee Q) \vee (P \vee Q)) \wedge R \\ &= \mathbf{T} \wedge R \\ &= R \end{aligned}$$

Duality in Propositional Logic

- In the above equivalence rules, we can actually see some duality (对偶性). Here, we present some more general results using the concept of **dual formulas (对偶公式)**.

Definition: Dual Formulas

Let α be a formula involving only $\neg, \vee, \wedge$. We define two new formulas:

- α^* is obtained by replacing $\vee, \wedge, \mathbf{T}$ and $\mathbf{F}$ by $\wedge, \vee, \mathbf{F}$ and $\mathbf{T}$, respectively;
- α^- is obtained by replacing each variable P_i by $\neg P_i$.

For example, for $\alpha = (R \vee Q) \wedge (P \vee R) \vee \mathbf{T}$, we have $\alpha^* = (R \wedge Q) \vee (P \wedge R) \wedge \mathbf{F}$ and $\alpha^- = (\neg R \vee \neg Q) \wedge (\neg P \vee \neg R) \vee \mathbf{T}$.

- **Result 1:** We can obtain the **generalized De Morgan's Law** as follows

$$\boxed{\neg\alpha = (\alpha^*)^-}$$

Proof: We prove by induction on the number of logical operators in α denoted by $|\alpha|$.

- Induction Basic: If $|\alpha| = 0$, i.e., there is no operator in α, then $\alpha = P$. Clearly, we have $(P^*)^- = P^- = \neg P = \neg\alpha$. Therefore, the induction basis holds.
- Induction Hypothesis: We assume that $\neg\alpha = (\alpha^*)^-$ for α such that $|\alpha| \leq k$.
- Induction Step: We consider α with $|\alpha| = k+1$ operators. There are three cases:

Case 1: $\alpha = \neg\alpha_1$, where α_1 has k operators. Then

$$\neg\alpha = \neg(\neg\alpha_1) = \neg((\alpha_1^*)^-) = ((\neg\alpha_1)^*)^- = (\alpha^*)^-$$

Case 2: $\alpha = \alpha_1 \wedge \alpha_2$, where $|\alpha_1| \leq k$ and $|\alpha_2| \leq k$. Then

$$\neg\alpha = \neg(\alpha_1 \wedge \alpha_2) = \neg\alpha_1 \vee \neg\alpha_2 = (\alpha_1^*)^- \vee (\alpha_2^*)^- = (\alpha_1^* \vee \alpha_2^*)^- = ((\alpha_1 \wedge \alpha_2)^*)^- = (\alpha^*)^-$$

Case 3: $\alpha = \alpha_1 \vee \alpha_2$, where $|\alpha_1| \leq k$ and $|\alpha_2| \leq k$. Then

$$\neg\alpha = \neg(\alpha_1 \vee \alpha_2) = \neg\alpha_1 \wedge \neg\alpha_2 = (\alpha_1^*)^- \wedge (\alpha_2^*)^- = (\alpha_1^* \wedge \alpha_2^*)^- = ((\alpha_1 \vee \alpha_2)^*)^- = (\alpha^*)^-$$

- **Result 2:** If $\alpha \to \beta$ is a tautology, then $\beta^* \to \alpha^*$ is a tautology.

Proof: We note that $\alpha \to \beta$ is a tautology iff $\neg\beta \to \neg\alpha$ is a tautology. Since $\neg\alpha = (\alpha^*)^-$ and $\neg\beta = (\beta^*)^-$, we have $(\beta^*)^- \to (\alpha^*)^-$ is a tautology. Hence $\beta^* \to \alpha^*$ is a tautology.

- **Result 3:** If $\alpha = \beta$, then $\alpha^* = \beta^*$.

Proof: We note that $\alpha = \beta$ iff $\alpha \leftrightarrow \beta$ is a tautology iff $\neg\alpha \leftrightarrow \neg\beta$ is a tautology. Since $\neg\alpha = (\alpha^*)^-$ and $\neg\beta = (\beta^*)^-$, we have $(\alpha^*)^- \leftrightarrow (\beta^*)^-$ is a tautology. This means that $\alpha^* \leftrightarrow \beta^*$ is a tautology, which further implies that $\alpha^* = \beta^*$.

Therefore, based on the above result, from equivalence $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$, we can actually get equivalence $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$ directly.

From Truth Tables to Formulas

- Now given a compound proposition formula α, we already know how to draw its truth table using the semantics of each logical operator. On the other hand, suppose that we are given a truth table, e.g., it could be a desired functionality of a circuit that we want to design. Then how can we find a compound proposition formula α such that the truth table of α is indeed the given one?

- **Approach 1: Write from each True assignment**.
  The idea of this approach is as follows: "*the truth table returns true iff one of its true rows holds*". Therefore, we can look at each row of the truth table that needs to be T. To make the condition in each true row holds, we need to list all the atomic propositions correctly by conjunction $\wedge$ (called a conjunctive clause). Furthermore, the entire formula is true if any of the conditions in the true rows is satisfied. Therefore, we need to connect all the above conjunctive clauses using disjunction $\vee$. This gives us the desired formula in the following form

$$(P_1 \text{ is } \mathsf{T}/\mathsf{F} \textbf{ and } \cdots \textbf{ and } P_n \text{ is } \mathsf{T}/\mathsf{F}) \textbf{ or } \cdots \textbf{ or } (P_1 \text{ is } \mathsf{T}/\mathsf{F} \textbf{ and } \cdots \textbf{ and } P_n \text{ is } \mathsf{T}/\mathsf{F})$$

- **Approach 2: Write from each False assignment**.
  We can think from the other way around: "*the truth table returns true iff none of its falses row holds*". Therefore, we can look at each row of the truth table that needs to be F. This row does not hold if one of its atomic propositional variable is not correct. Therefore, we get a set of disjunctive clauses. Furthermore, these disjunctive clauses need to hold simultaneously. This gives us the desired formula in the following form

$$(P_1 \text{ is } \mathsf{T}/\mathsf{F} \textbf{ or } \cdots \textbf{ or } P_n \text{ is } \mathsf{T}/\mathsf{F}) \textbf{ and } \cdots \textbf{ and } (P_1 \text{ is } \mathsf{T}/\mathsf{F} \textbf{ or } \cdots \textbf{ or } P_n \text{ is } \mathsf{T}/\mathsf{F})$$

- **Example :** We consider the following truth table and we want to get its formula.

P	Q	R	α
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

  – If we write from true, we need to make sure that (only) rows $1, 2, 4, 5, 8$ hold. Therefore, we have

$$\begin{aligned}\alpha =& (\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \\ & \vee (P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge R)\end{aligned}$$

  – If we write from false, we need to make sure that (only) rows $3, 6, 7$ do not hold. Therefore, we have

$$\alpha = \underbrace{(P \vee \neg Q \vee R)}_{=\neg(\neg P \wedge Q \wedge \neg R)} \wedge \underbrace{(\neg P \vee Q \vee \neg R)}_{=\neg(P \wedge \neg Q \wedge R)} \wedge \underbrace{(\neg P \vee \neg Q \vee R)}_{=\neg(P \wedge Q \wedge \neg R)}$$

Complete Set for Operators

- So far we have investigated some basic logical operators including negation $\neg$, conjunction $\wedge$, disjunction $\vee$, implication $\rightarrow$ and biconditional statement $\leftrightarrow$. There are also some other commonly used logical operators, e.g.,
  - Exclusive OR (异或): $P \oplus Q = (\neg P \wedge Q) \vee (P \wedge \neg Q)$
  - NAND (与非): $P \uparrow Q = \neg(P \wedge Q)$
  - NOR (或非): $P \downarrow Q = \neg(P \vee Q)$
- Essentially, a logical operator can be considered as a multi-variable Boolean function $f : \underbrace{\{0,1\} \times \cdots \times \{0,1\}}_{n \text{ times}} \rightarrow \{0,1\}$. For example, $\neg : \{0,1\} \rightarrow \{0,1\}$ is a one-variable function, and $\wedge : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$ is a two-variable function. More specifically, there are $2^2 = 4$ one-variable functions shown below (three of them seems not useful)

P	$f_0(P)$	$f_1(P)$	$f_2(P)$	$f_3(P)$
0	0	0	1	1
1	0	1	0	1

Also, there are $2^{2^2} = 16$ two-variable functions shown below, where g_1 is actually "$\wedge$"

P	Q	$g_0(P,Q)$	$g_1(P,Q)$	$\cdots$	$g_{15}(P,Q)$
0	0	0	0		1
0	1	0	0		1
1	0	0	0		1
1	1	0	1		1

- In general, given n atomic propositional variables $P_1, \cdots, P_n$, we can design 2^{2^n} different logical operators. So the question is that do we really need them. To this end, we introduce the concept of **functionaly complete set (功能完备集)** of logical operators.

Definition: Completeness

Let C be a set of logical operators. Then we say that C is functionally complete if we can express any truth table using only operators in C.

The followings are some examples of complete set:

1. The set of all multivariable functions (infinite) is clearly functionally complete.
2. Since any truth table can be written from true or false as discussed above, we know that $C = \{\neg, \wedge, \vee\}$ is functionally complete.
3. Furthermore, since $P \wedge Q = \neg(\neg P \vee \neg Q)$, we know that $\{\neg, \vee\}$ is also complete.
4. However, one can show that $\{\vee\}$ and $\{\vee, \wedge\}$ are not complete. For example, negation cannot be expressed only by using $\vee$.

Normal Forms

- We note that there may have may formulas that look different but are essentially equivalent, e.g., $P \to Q = \neg(P \wedge \neg Q) = \neg P \vee Q$. Therefore, it may be useful to express a formula in a better structured manner, which we call **normal forms (范式)**.

- In fact, we have provided already provided such a structured way for writing a formula. If we write from false, then we get the **Conjunctive Normal Form (合取范式/CNF)**

$$(\bullet \vee \cdots \vee \bullet) \wedge (\bullet \vee \cdots \vee \bullet) \wedge \cdots \wedge (\bullet \vee \cdots \vee \bullet)$$

If we write from true, then we get the **Disjunctive Normal Form (析取范式/DNF)**

$$(\bullet \wedge \cdots \wedge \bullet) \vee (\bullet \wedge \cdots \wedge \bullet) \vee \cdots \vee (\bullet \wedge \cdots \wedge \bullet)$$

- More formally, we have the following terminologies:

Definition: Literals, Clauses & Normal Forms

– a literal (文字) is an atomic proposition or its negation, e.g., $P, \neg P, Q, \neg Q, \ldots$

– a conjunctive clause (合取子句) is a finite conjunction of literals, e.g., $P, \neg P, P \wedge Q, P \wedge Q \wedge \neg P, \ldots$

– a disjunctive clause (析取子句) is a finite disjunction of literals, e.g., $P, \neg P, P \vee Q, P \vee Q \vee \neg P, \ldots$

– a conjunctive normal form (合取范式) is a finite conjunction of disjunctive clauses in the form of $\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n$, where $\alpha_i = (\bullet \vee \cdots \vee \cdots \vee \bullet)$

– a disjunctive normal form (析取范式) is a finite disjunction of conjunctive clauses in the form of $\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n$, where $\alpha_i = (\bullet \wedge \cdots \wedge \cdots \wedge \bullet)$

- Normal forms are very useful in solving the SAT problems. For example, for CNF $\alpha = (\neg P_1 \vee \neg P_2 \vee P_3) \wedge (P_1 \vee P_3) \wedge (\neg P_3)$, from the last clause, we know that $P_3 = 0$. This together with the second clause tell that $P_1 = 1$. Then for the first clause, we have to have $P_2 = 0$. This helps us to find an assignment $(1, 0, 0) \models \alpha$ quickly.

- We can also use normal forms to check tautology or contradiction. Specifically,

  – For a CNF α, if each disjunction clause in it contains some proposition and its negation, then is tautology.

  – For a DNF α, if each conjunction clause in it contains some proposition and its negation, then is contradiction.

Convert a Formula into CNF/DNF

▶ Clearly, any formula can be converted to a CNF or a DNF. A naive approach is to first draw its true table and then we can get the DNF (or CNF) by writing from T (or F).

Example: Obtain CNF/DNF from Truth Tables

P	Q	$(P \vee Q) \to (P \wedge Q)$
0	0	1
0	1	0
1	0	0
1	1	1

For formula $\alpha = (P \vee Q) \to (P \wedge Q)$, its truth table is shown on the LHS, then

– $\alpha_{\text{DNF}} = (\neg P \wedge \neg Q) \vee (P \wedge Q)$

– $\alpha_{\text{CNF}} = \underbrace{(P \vee \neg Q)}_{=\neg(\neg P \wedge Q)} \wedge \underbrace{(\neg P \vee Q)}_{=\neg(P \wedge \neg Q)}$

▶ The above approach, however, is very inefficient because it needs to write down all 2^n rows in the truth table. In fact, we can simply use logical equivalence to obtain CNFs or DNFs by the following steps:

Procedure: Obtain CNF/DNF by Logical Equivalence

1. Cancel " $\leftrightarrow$ " and " $\to$ " using the followings
   ① $\alpha \to \beta = \neg\alpha \vee \beta$,
   ② for DNF: $\alpha \leftrightarrow \beta = (\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$
   ③ for CNF: $\alpha \leftrightarrow \beta = (\alpha \vee \neg\beta) \wedge (\neg\alpha \vee \beta)$

2. "$\neg$" runs into "( )" using the followings
   ① $\neg(\alpha \wedge \beta) = \neg\alpha \vee \neg\beta$ ② $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$.

3. " $\wedge$ " and " $\vee$ " run into "( )" using the followings
   ① for DNF: $\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
   ② for CNF: $\alpha \vee (\beta \wedge \gamma) = (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

4. Step 4: Simplify if needed using, e.g.,
   ① $\alpha \wedge \mathbf{F} = \mathbf{F}$ ② $\alpha \vee \mathbf{F} = \alpha$ ③ $\alpha \vee \mathbf{T} = \mathbf{T}$ ④ $\alpha \wedge \mathbf{T} = \alpha$.

▶ Example: we obtain DNF as follows

$$\begin{aligned}
&\neg(P \vee Q) \leftrightarrow (P \wedge Q)\\
=&(\neg(P \vee Q) \wedge (P \wedge Q)) \vee (\neg\neg(P \vee Q) \wedge \neg(P \wedge Q))\\
=&(\neg P \wedge \neg Q \wedge P \wedge Q) \vee ((P \vee Q) \wedge (\neg P \vee \neg Q))\\
=&(\neg P \wedge \neg Q \wedge P \wedge Q) \vee ((P \wedge \neg P) \vee (P \wedge \neg Q) \vee (Q \wedge \neg P) \vee (Q \wedge \neg Q))\\
=&(\neg P \wedge \neg Q \wedge P \wedge Q) \vee (P \wedge \neg P) \vee (P \wedge \neg Q) \vee (Q \wedge \neg P) \vee (Q \wedge \neg Q)\\
=&\mathbf{F} \vee \mathbf{F} \vee (P \wedge \neg Q) \vee (Q \wedge \neg P) \vee \mathbf{F}\\
=&(P \wedge \neg Q) \vee (\neg P \wedge Q)
\end{aligned}$$

Logical Inference

- In many problems, we have the following structure for **inference (推理)**: "*if proposition α is true, then proposition β is true.*" The "if" part is called the hypothesis or premises (前提) and the "then" part is called the conclusion (结论).

- For example, if we have two premises "*If today is Friday, then we take discrete math.*" and "*Today is Friday.*", then we can get the conclusion that "*We take discrete math*". This actually leads to the following form of inference

$$(P \to Q) \wedge P \Rightarrow Q$$

However, our common knowledge tells us that $(P \to Q) \wedge \neg P \Rightarrow \neg Q$ is not a valid inference because, e.g., "*Today is Monday*" does not necessarily imply that "*We do not take discrete make*". Actually, we see that the requirement that "if α, then β" fits the definition of implication. Furthermore, we need the implication always holds. This leads to the definition of **tautological implication (重言蕴含)**.

Definition: Tautological Implication

Let α and β be two compound proposition formulas. We say α tautologically implies β, or β is the logical conclusion of α, denoted by $\alpha \Rightarrow \beta$, if β is true whenever α is true.

- Based on the above definition, we have the following equivalent results:

Theorem

The following four statements are equivalent:
① $\alpha \Rightarrow \beta$ ② $\alpha \to \beta$ is a tautology. ③ $\alpha \wedge \neg\beta$ is a contradiction. ④ $\neg\beta \Rightarrow \neg\alpha$.

Proof: Clearly, ① iff ②, which is just the definition of tautology and $\to$.
Also, ② iff ③ since $\neg(\alpha \to \beta) = \alpha \wedge \neg\beta$ and φ is tautology iff $\neg\varphi$ is contradiction.
Finally, we have ① iff ④ since

$$\alpha \to \beta = \neg\alpha \vee \beta = \neg(\neg\beta) \vee \neg\alpha = \neg\beta \to \neg\alpha.$$

Therefore, when we want to prove $\alpha \Rightarrow \beta$, we can either show ③, which is the so called **proof by contradiction (反证法)**, or show ④, which is the so-called **proof by contraposition (换位证明)**.

- **Notation:** In general, when we write $\alpha \Rightarrow \beta$, the premise α is usually the conjunction several sub-premises. Therefore, for $\boxed{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}$, we usually just write it as $\boxed{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n \vdash \beta}$.

Basic Formula of Inference

- Here we discuss some commonly used inferences:

① $P \wedge Q \Rightarrow P$	② $P \Rightarrow P \vee Q$
③ $\neg P \Rightarrow P \to Q$	④ $Q \Rightarrow P \to Q$
⑤ $\neg(P \to Q) \Rightarrow P$	⑥ $\neg(P \to Q) \Rightarrow \neg Q$

⑦ $\neg P \wedge (P \vee Q) \Rightarrow Q$

⑧ $P \wedge (P \to Q) \Rightarrow Q$ (**Modus Ponens, 肯定前件**)

⑨ $(P \to Q) \wedge \neg Q \Rightarrow \neg P$ (**Modus Tollens, 否定后件**)

⑩ $(P \to Q) \wedge (Q \to R) \Rightarrow P \to R$

⑪ $(P \leftrightarrow Q) \wedge (Q \leftrightarrow R) \Rightarrow P \leftrightarrow R$

⑫ $(P \to R) \wedge (Q \to S) \wedge (P \vee Q) \Rightarrow R \vee S$

Proof by Rules of Inferences

- The proof procedure for $\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n \vdash \beta$, can be understood as follows:
  1. Initially, we have a set of knowledge from α_1 to α_n. This is called the **rule of premise**, i.e., we can always take any premise as a correct conclusion.
  2. Each time, we use existing knowledge and inference rules to generate new knowledge. For example, we can combing some existing results using basic formula of inference, or basic formula of equivalence.
  3. Once the conclusion β is generated as an knowledge, we complete the proof.

- Sometimes, we may need to prove inference of form $\alpha_1 \Rightarrow \alpha_2 \to \beta$. This suffices to prove $\alpha_1 \wedge \alpha_2 \Rightarrow \beta$ and vice versa. Therefore, we can also use the premise in the conclusion as the actual premise. This is called the **rule of conditional proof**.

Example: Proof by Rules of Inferences

Prove $P \to Q, Q \to R, P \vdash R$

(1) P	premise
(2) $P \to Q$	premise
(3) Q	MP (1)(2)
(4) $Q \to R$	premise
(5) R	MP (3)(4)

Prove $P \to (Q \to S), \neg R \vee P, Q \vdash (R \to S)$

(1) $\neg R \vee P$	premise
(2) $R \to P$	equivalence (1)
(3) R	conditional proof
(4) P	MP (2)(3)
(5) $P \to (Q \to S)$	premise
(6) $Q \to S$	MP (4)(5)
(7) Q	premise
(8) S	MP (6)(7)
(9) $R \to S$	conditional proof

Proof by Resolutions

- When we use rules of inferences, we need to think at each step what rules to adopt. For computer-aided proofs, we may need a more algorithmic procedure, in which each step has clear actions. One of the most popular such approaches is the **resolution method (归结法/消解法)**. The basic idea of resolution is very simple:

  To prove that $\alpha \to \beta$ is a tautology, it suffices to prove that $\alpha \land \neg\beta$ is a contradiction.

- The following procedure guarantees to find a contradiction for a formula (if exists).

Resolution Method

Step 1: Convert $\alpha \land \neg\beta$ into CNF. For example,

$$\alpha \land \neg\beta = P \land (P \lor R) \land (\neg P \lor \neg Q) \land (\neg P \lor R)$$

Step 2: Write down the set of all disjunction clauses in the CNF. For example

$$S = \{P, P \lor R, \neg P \lor \neg Q, \neg P \lor R\}$$

Step 3: Resolve clause set S.

- Idea: $(P \lor R) \land (\neg P \lor Q) \Rightarrow R \lor Q$
- Method: If S has two clauses of forms $C_1 = L \lor C_1'$ and $C_2 = \neg L \lor C_2'$, then we can generate a new clause $C_1' \lor C_2'$ in S.

Step 4: Repeat until we find a contradiction, e.g., two clauses P and $\neg P$.

- The above procedure is actually both sound and complete but we will not prove it here. We illustrate the procedure by the following examples.

Examples: Proof by Resolutions

Example 1: Prove $(P \to Q) \land P \Rightarrow Q$

1. $(P \to Q) \land P \land \neg Q = (\neg P \lor Q) \land P \land \neg Q$
2. $S = \{\neg P \lor Q, P, \neg Q\}$
3. $\left.\begin{array}{r}\neg P \lor Q\\ P \lor \mathbf{F}\end{array}\right\} \to Q \lor \mathbf{F} = Q$ and $S' = \{\dots, Q, \neg Q\}$
4. We find a contradiction Q and $\neg Q$.

Example 2: Prove $((P \to Q) \land (Q \to R)) \Rightarrow (P \to R)$

1. $(\neg P \lor Q) \land (\neg Q \lor R) \land \neg(\neg P \lor R) = (\neg P \lor Q) \land (\neg Q \lor R) \land P \land \neg R$
2. $S = \{\neg P \lor Q, P, \neg Q \lor R, \neg R\}$
3. $\left.\begin{array}{r}\neg P \lor Q\\ P \lor \mathbf{F}\end{array}\right\} \to Q, \quad \left.\begin{array}{r}\neg Q \lor R\\ Q \lor \mathbf{F}\end{array}\right\} \to R$
4. We find a contradiction R and $\neg R$.