Chapter 1 The Foundation: Logic and Proofs

Propositional Logic & its applications

the basic building blocks of logic is proposition: a declarative (statement) sentence that is either true or false but not both.

使用字母 p.q.r.s.... 来 denote propositional variables

truth value { true (T), if it's a true proposition / false (F) if it's a false proposition

Truth Table: rows are all possible true values (combinations) and its result; columns are full true values of a proposition

p that cannot be expressed in terms of simpler propositions are called atomic propositions

The area of logic that deals with propositions is called the propositional calculus (logic)

从现有 p. q form compound propositions using logic operators / connectives

Def 1. negation of p $\neg p$ "It's not that case that p." its truth value is the opposite of p's. ($\bar{p}$, $-p$, p', Np, $!p$)

also can be considered as the result of the operation of the negation operator on a proposition

Def 2. conjunction of p and q $p \wedge q$

"p and q" (or "p but q")

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Def 3. disjunction of p and q $p \vee q$

"p or q"

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

$p \vee q$ is true when both true or exactly one true. in other words "at least one"

the use of this "or" corresponds to one of the two ways the word "or" is used in English. namely "inclusive or"

Def 4. "exclusive or" of p and q $p \oplus q$ (or $p \, XOR \, q$)

true when exactly one of p and q is true and is false otherwise

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

other important ways in which propositions can be combined:

Def 5. conditional statement (implication) $p \to q$ "condition" means we asserts that q is true on the condition the p holds

只说了 p is true 可以以 q is true 作为条件推得

"if p, then q" ⇒ is false when p is true and q is false and true otherwise → we let the truth value be so to make it a proposition.

p is called the hypothesis (or antecedent or premise)

q is called the conclusion (or consequence)

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

都映射到大的事件 [illegible] 有 q需要满足m条件们是p的m子集

故也即 p要想成立，必须满足q成立.

A useful way to understand the truth value of a conditional statement is to think of an obligation or a contract, such as "If I am elected, then I will lower taxes." "If you get 100% on the final, then you will get an A."

doesn't say anything about the cases when p is false. so it's not broken in these cases.

different ways to express the conditional statement:

"p only if q" ⇔ "if p, then q" ⇔ (p is sufficient for q)

⇔ (q is necessary for p)

只说了 p cannot be true when q is not true. 故只在 p T. q F 时是 F 的.

[$\neg q \to \neg p$ ⇔ $p \to q$]

⇔ "q unless $\neg p$": q if not $\neg p$: q if p.

We can form some new conditional statements starting with a conditional statement $p \to q$:

$q \to p$ is called the converse (逆) of $p \to q$ (TF/FT 互换了)

$\neg q \to \neg p$ is called the contrapositive of $p \to q$ ⇒ only it has the same truth value as $p \to q$

$\neg p \to \neg q$ is called the inverse (反) of $p \to q$

Two compound propositions that have the same truth values in all possible cases are called equivalent.

Def 6. biconditional statement (bi-implications) $p \leftrightarrow q$ $(p \to q) \wedge (q \to p)$.

"p if and only if q"

"p is necessary and sufficient for q"

"if p then q, and conversely"

"p iff q" "p exactly when q"

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

IMPLICIT USE OF BICONDITIONS

in nature languages, biconditionals are not always explicit — the converse may be implied, but not stated. :(

so we will always distinguish $p \to q$ and $p \leftrightarrow q$ for precision.

Precedence of Logic Operators

$\{\neg, \land\}$ $\{\neg, \lor\}$ $\{\neg, \to\}$ 均是最小全功能连结词集

$\neg$: 1 !

$\land$ 2 takes precedence over $\lor$ 3

$\to$ 4 $\leftrightarrow$ 5 low

Logic and Bit Operations

Computers represent information using bits. A **bit** is a symbol with two possible values: 0, 1, coming from binary digit (digits used in binary representations of numbers)

a bit can be used to present a truth table. as is customarily done we use 1 bit to represent true and 0 bit: false.

and a variable is called a Boolean variable if its value is either true or false, can be represented using a bit

Computer bit operations correspond to the logic connectives — just replace T by 1, F by 0, being called OR, AND, XOR

OR ↔ $\lor$, AND ↔ $\land$, XOR ↔ $\oplus$

A <u>bit string</u> is a sequence of zero or more bits. and its length is # bits in it

We can extend bit operations to bit strings. define bitwise OR, bitwise AND and bitwise XOR of two strings of the same length. 指相对应的bit作操作后的bit构成的string.

Propositional Equivalences

An important type of step used in mathematical argument is the replacement of a statement with another statement with the same truth value.

we focus on these methods that produce propositions with the same truth value as a given compound proposition.

Def 1.
- tautology ⇒ always true. $\equiv T$
  (→ negation)
- contradiction ⇒ always false. $\equiv F$
- contingency: neither a tautology nor a contradiction

tautology 用来描述 fact → right proof
$p \equiv q$ 即永真情况下的 $p \leftrightarrow q$ ✓

Def 2 Compound propositions that have the same truth values in all possible cases are called **logically equivalent**

also means ("$p \leftrightarrow q$" is a tautology) $p \equiv q$ or $p \Leftrightarrow q$ → not a logical connective and $p \equiv q$ is not a compound proposition.

example
De Morgan laws: $\neg(p \vee q) \equiv \neg p \wedge \neg q$
$\neg(p \wedge q) \equiv \neg p \vee \neg q$

Conditional-disjunction equivalence $p \to q \equiv \neg p \vee q$

2^n rows are required if a compound p. involves n propositional variables.

→ we really need more efficient ways to establish logic equivalences. such as by using ones we already know

just replace :)
(not change truth values)

Identity laws $\wedge T$. $\vee F$

Domination laws $\vee T$. $\wedge F$

Idempotent laws p p

Double negation law $\neg\neg$

Commutative laws
Associative laws same
Distributive laws $\vee$ with $\wedge$

$\neg$ (De Morgan's laws) $\neg$ with $\wedge/\vee$ can extend to n p.variables compound p. $\neg(\bigvee_{j=1}^{n} p_j) \equiv \bigwedge_{j=1}^{n} \neg p_j$ $\neg(\bigwedge_{j=1}^{n} p_j) \equiv \bigvee_{j=1}^{n} \neg p_j$

Absorption laws $p \wedge q$ get "smaller than p"; $p \vee q$ get "bigger than p" $(p \wedge q) \vee q \equiv q$. $(p \vee q) \wedge q \equiv q$

Negation laws $p \vee \neg p \equiv T$. $p \wedge \neg p \equiv F$

<Involving Conditional Statements>

conditional-disjunction equivalence $p \to q \equiv \neg p \vee q$
$(p \to q \equiv \neg q \to \neg p)$

$p \vee q \equiv \neg p \to q$

$p \wedge q \equiv \neg(p \to \neg q)$

$\neg(p \to q) \equiv p \wedge \neg q$

$(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$ 前同则同
(∨) (∨)

$(p \to r) \wedge (q \to r) \equiv (p \vee q) \to r$ 后同则异
(∨) (∧)

<involving biconditional statements>

$p \leftrightarrow q \equiv (p \to q) \vee (q \to p) \equiv \neg p \to \neg q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$
(TT. FF)

反置

$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$ rows TF, FT → TT, FF ✓

[with several p.variables]
compound proposition 的范式.

disjunction 范式: ⊙ ∨ ⊙ ∨ … ∨ ⊙
全是由"propositional variables 及其 negation"构成的一个个 conjunction compound p.

conjunction 范式. ⊗ ∧ ⊗ ∧ … ∧ ⊗
全是由 propositional variables 及其 negation 构成的 disjunction compound p.

主 d.范式 ⇒ 各○均含且仅含一次"出现的所有 p.v."(或其 negation)
eg $p \wedge \neg q \wedge r$ [2^n种 "极大项"]

主 c.范式 ⇒ 同上. eg $p \vee \neg q \vee r$ [2^n种 "极小项"]

Satisfiability computer helps us :)

a compound proposition is satisfiable if there is an assignment of truth values to its variables that make it true
[tautology or contingency]

unsatisfiable [contradiction]

3. Predicates and Quantifiers

predicates

Propositional logic till now can't adequately express the meaning of all statements in mathematics and in natural language. So we introduce a more powerful type of logic called **predicate logic**.

其实不是严格意义上的 proposition
statements involving (uncertain) variables are neither true or false when the value of the variables are not specified. We discuss the ways that propositions can be produced from such statements

"x is greater than 3" → the predicate — refers to a property that the subject of the statement can have

x: variable — the subject of the statement

denoted by $P(x)$: the value of the propositional function P at x (代入 assigned x)

可理解为把不确定(元)真值的 statement map 到 have a truth value 的 propositions.

→ function waiting to put in specified variables

> we can also have statements involving more than one variable (subject). denoted by $P(x_1, x_2, \dots, x_n)$ — the value of the propositional function P at the n-tuple $(x_1, x_2, \dots, x_n)$ or n-place (n-ary) predicate

quantifiers

quantification are introduced to express the extent to which a predicate is true over the elements under consideration.

Such as the words "all, some, none ..."

The area of logic that deals with prediates and quantifiers is called the **predicate calculus**

We will focus on two types of quantification here:

Def1 **universal quantification** of $P(x)$ is the statement "$P(x)$ for all values of x in the domain" (domain of discourse or the universe of discourse) ⇒ specifies the possible values of the variable x

asserts that $P(x)$ is true for ...

$\forall x\, P(x)$ "for all (every) x $P(x)$" (It's best to avoid using "for any x" because it is often ambiguous as to whether "any" means "every" or "some" (unambiguous in negatives))

$\forall$: universal quantifier

⇓ the universal quantification of $P(x)$ changes when we change the domain

So the domain must always be specified when a u.q. is used (without it, u.q. is not defined)

An element for which $P(x)$ is false is called a counterexample to $\forall x\, P(x)$

Remark: Generally, an implicit assumption is made that all domains of discourse for quantifiers are nonempty. If empty, $\forall x\, P(x)$ is true, because no elements x in the domain for which $P(x)$ is false.

Def2 **existential quantification** of $P(x)$ is the proposition "There exists element x in the domain such that $P(x)$."

"for at least one" "there is" "for some"

$\exists x\, P(x)$

$\exists$: existential quantifier

$\exists x\, P(x)$ is false when $P(x)$ is false for every x. (domain empty, $\exists x P(x)$ false.)

uniqueness quantifier

quantifier that is most often seen among all other quantifiers describing extents (we can define a lot actually...) is the uniqueness quantifier $\exists! x\, P(x)$ ($\exists_1$) "There exists a unique x such that $P(x)$ is true." ("exactly one")

(can use quantifiers and propositional logic to express uniqueness so it can be avoided)

> quantifiers over finite domains

$\forall x\, P(x) \equiv P(x_1) \wedge P(x_2) \cdots \wedge P(x_n)$

$\exists x\, P(x) \equiv P(x_1) \vee P(x_2) \cdots \vee P(x_n)$

By the way, it's sometimes helpful to think in terms of looping and searching when determining the truth value of a quantification.

Quantifiers (with restricted domains)

as an abbreviated notion

即可视作 domain. [condition can means just the specification of domain.]

$\forall x<0\,(x^2>0) \equiv \forall x\,(x<0 \rightarrow x^2>0)$ 而非 ∧. 若为∧则指二者对 ∀x 均同时成立.

$\exists z>0\,(z^2=2) \equiv \exists z\,(z>0 \wedge z^2=2)$ 而非 "→". 若为 "→", 不管前为T时如何, 只要有一个 x 使前为 F, "→" 就是 T 了.

precedence of quantifiers: the quantifiers $\exists$ and $\forall$ have higher precedence than all logical operators from propositional calculus $\forall x P(x) \vee Q(x) \equiv (\forall x P(x)) \vee Q(x)$

⇓ will just bind the variable after it without parentheses

So we may often need to add parentheses for quantifier to express its true bigger scope (just below)

Binding Variables.

When a quantifier is used on the variable x, we say that this occurrence of the variable is **bound**.

An occurrence of a variable that are not bound by a quantifier or set equal to a particular value is said to be **free**

All the variables that occur in a propositional function must be bound or set equal to a particular value to turn it into a proposition

This can be done using a combination of universal quantifiers, existential quantifiers and value assignments

The part of a logical expression to which a quantifier is applied is called the **scope** of this quantifier.

→ Consequently, a variable is free if it is outside the scope of all quantifiers in the formula that specify this variable.

[same or different letters can be used to represent variables bound by different quantifiers with scopes that do not overlap]

scope 的缩放: V. 人不不改变其 bind 到 variables 的前提下直接缩放

→ 且 quantifier 在前 则需改 quantifier: $\forall x\, P(x) \rightarrow S \equiv \exists x\,(P(x) \rightarrow S)$; $\exists x\, P(x) \rightarrow S \equiv \forall x\,(P(x) \rightarrow S)$ ★ 原因是 S 无 quantifier

在后则不用改. $S \rightarrow \forall x P(x) \equiv \forall x (S \rightarrow P(x))$; $S \rightarrow \exists x P(x) \equiv \exists x (S \rightarrow P(x))$

Logic Equivalences Involving Quantifiers

We extend the notion of logical equivalences to expressions involving predicates and quantifiers

Def 3. Statements involving predicates and quantifiers are logically equivalent if and only if they have the same truth value no matter which predicates are substituted into these statements and which domain of discourse is used for the variables in these propositional functions.

$S \equiv T$ proof by showing { if S is true, then T is true (false involved) (but common)
if T is true, then S is true (is true) if and only if T (is true)
in other words : S (is true) if and only if T (is true)

Negating Quantified Expressions De Morgan's laws for quantifiers

↳ is called this because matching quantifiers over finite domains and then matching the previous De Morgan's laws

$\neg\forall x P(x) \equiv \exists x \neg P(x)$

if and only if

$\forall x P(x)$ is true iff $\forall x P(x)$ is false iff there's an element x in the domain for which $P(x)$ is false

iff $\exists x \neg P(x)$ is true. □

$\neg\exists x P(x) \equiv \forall x \neg P(x)$

Nested Quantifiers

In formal section we actually avoid nested quantifiers, where one quantifier is within the scope of another.

Note that: the everything within the scope of a quantifier can be thought of as a propositional function

Understanding Statements Involving Nested Quantifiers :

(We can think of quantification as loops (finite cases) using loop x loop and "hit" and "for every" :)

It's important to note that the order of the quantifiers is important.

- the order of nested universal (or existential) quantifiers in a statement without other quantifiers can be changed without changing the meaning of the quantified statement.
- Be careful with the order of existential and universal quantifiers

$\forall x \exists y$: y possibly depend on x → so the choice of y is limited more and the proposition will be true more likely ("extent" get smaller)

$\exists y \forall x$: y is a constant independent of x

$\exists y \forall x P(x,y) \rightarrow \forall x \exists y P(x,y)$ [inverse isn't necessary to be true.

So far we can use nested quantifiers to express "$\exists!$" quantifier

$\forall x \exists! y B(x,y) \equiv \forall x \exists y (B(x,y) \wedge \forall z((z \neq y) \rightarrow \neg B(x,z)))$

前束范式
把 quantifiers 都移到最前面.
可利用 "换名"
和 "quantifier 的分配律"
↳ ∀对∧才有⇔
∃对∨才有⇔

Negating Nested Quantifiers

Just be negated by successively applying the rules for negating statements involving a single quantifier :)

Rules of Inference

Later in this chapter we will study proofs. Proofs in mathematics are valid arguments that establish the truth of mathematical statements.

an argument: a sequence of statements (propositions) that end with a final conclusion

mean that the conclusion, or final statement of the argument must follow from the truth of the preceding statements or premises of the argument

all but the final p. in the argument

An argument is valid if and only if (it is impossible for: all the premises to be true and the conclusion to be false) → the truth of all its premises implies that the conclusion is true

Firstly we focus on valid arguments in propositional logic>

an argument is valid ← replace propositions by propositional variables ← argument form: a sequence of compound propositions involving propositional variables

is valid if no matter which particular propositions are substituted for the propositional variables in its premises, the conclusion is true whenever the premises are all true.

☆ essentially, it means an argument form with premises $p_1, p_2, \ldots, p_n$ and conclusion q is valid exactly when $(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \rightarrow q$ is a tautology

a valid argument must likely lead to an incorrect conclusion if one or more of its premises is false.

To deduce new statements from statements we already have, we use rules of inference which are "templates" for constructing valid arguments

basic tools for establishing the truth of statements

Rules of Inference for propositional logic:

truth table is not useful...

Instead, we can first establish the validity of some relatively simple argument forms, called rules of inference and use them to build blocks to construct more complicated valid argument forms

- Modus ponens (law of detachment) $(p \wedge (p \rightarrow q)) \rightarrow q$ (tautology) (leading to the valid argument form $\frac{p \quad p \rightarrow q}{\therefore q}$)
- Modus tollens $(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$ $\frac{\neg q \quad p \rightarrow q}{\therefore \neg p}$
- Hypothetical syllogism $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$ $\frac{p \rightarrow q \quad q \rightarrow r}{\therefore p \rightarrow r}$

Proof: the conclusion proposition is a condition statement, so to test its truth value we can bring in conditional proof assumption

CP规则 [Condition Proof] ($H_1 \cdots H_m P \rightarrow Q \equiv H_1 \cdots H_m \rightarrow (P \rightarrow Q)$)

(只要 conclusion 为 conclusion 不是 false, conclusion 就一定为 true)

- Disjunctive syllogism $(p \vee q) \wedge \neg q) \rightarrow p$ $\frac{p \vee q \quad \neg p}{\therefore q}$

- Addition $\quad p \to (p \lor q) \quad \frac{p}{\therefore p \lor q}$
- Simplification $\quad p \land q \to p \quad \frac{p \land q}{\therefore p}$
- Conjunction $\quad ((p) \land (q)) \to p \land q \quad \frac{p \quad q}{\therefore p \land q}$
- Resolution $\quad ((p \lor q) \land (\neg p \lor r)) \to (q \lor r) \quad \frac{p \lor q \quad \neg p \lor r}{q \lor r}$

Proof: Using hypothetical syllogism

$\neg p \to q. \quad p \to r. \quad \to \quad \neg q \to r \equiv (q \lor r)$

$\neg q \to p.$

use 置换 p. equivalent :)

Resolution

Computer programs have been developed to automate the task of reasoning and proving theorems. Many of these programs make use of a rule of inference known as resolution.

based on the tautology: $(p \lor q) \land (\neg p \lor r) \to (q \lor r)$ — $(q \lor r)$ is called the resolvent

let $r = F$. we get $(p \lor q) \land \neg p \to q$ [disjunctive syllogism]

$q = r$. $(p \lor q) \land (\neg p \lor q) \to q$

EXAMPLE 6 Show that the premises "It is not sunny this afternoon and it is colder than yesterday," "We will go swimming only if it is sunny," "If we do not go swimming, then we will take a canoe trip," and "If we take a canoe trip, then we will be home by sunset" lead to the conclusion "We will be home by sunset."

Extra Examples

Solution: Let p be the proposition "It is sunny this afternoon," q the proposition "It is colder than yesterday," r the proposition "We will go swimming," s the proposition "We will take a canoe trip," and t the proposition "We will be home by sunset." Then the premises become $\neg p \land q$, $r \to p$, $\neg r \to s$, and $s \to t$. The conclusion is simply t. We need to give a valid argument with premises $\neg p \land q$, $r \to p$, $\neg r \to s$, and $s \to t$ and conclusion t.

We construct an argument to show that our premises lead to the desired conclusion as follows.

Step	Reason
1. $\neg p \land q$	Premise
2. $\neg p$	Simplification using (1)
3. $r \to p$	Premise
4. $\neg r$	Modus tollens using (2) and (3)
5. $\neg r \to s$	Premise
6. s	Modus ponens using (4) and (5)
7. $s \to t$	Premise
8. t	Modus ponens using (6) and (7)

+ equivalents' using

Note that we could have used a truth table to show that whenever each of the four hypotheses is true, the conclusion is also true. However, because we are working with five propositional variables, p, q, r, s, and t, such a truth table would have 32 rows.

Rules of Inference for Quantified Statements

TABLE 2 Rules of Inference for Quantified Statements.

Rule of Inference	*Name*
$\forall x P(x)$ / $\therefore P(c)$	Universal instantiation
$P(c)$ for an arbitrary c / $\therefore \forall x P(x)$	Universal generalization
$\exists x P(x)$ / $\therefore P(c)$ for some element c	Existential instantiation
$P(c)$ for some element c / $\therefore \exists x P(x)$	Existential generalization

(Combining Rules of Inference for Propositions and Quantified Statements
eg. universal modus ponens and universal modus tollens)

间接 proof, 即反证法

proof ways: ① 证后为false时前必为false
即证 $\neg q \to \neg p$.
② 用equivalent 和已知 "⇒" 来推.

(if "$p \to q$" is a tautology. we can notion it as $p \Rightarrow q$).

也即 inference 的 rules in 各种打包

such as:

$\forall x A(x) \lor \forall x B(x) \Rightarrow \forall x (A(x) \lor B(x))$

$\exists x (A(x) \land B(x)) \Rightarrow \exists x A(x) \land \exists x B(x)$

即 ∀与∨一起 或 ∃与∧一起时 仅有这两个永真式而无等价式.

$\forall x (A(x) \to B(x)) \Rightarrow \forall x A(x) \to \forall x B(x)$

$\exists x (A(x) \to B(x)) \Rightarrow \forall x A(x) \to \exists x B(x)$

nested quantifiers

1. $\forall x \forall y A(x,y) \Rightarrow \forall x A(x, x)$
2. $\exists x A(x, x) \Rightarrow \exists x \exists y A(x,y)$
3. $\forall x \forall y A(x,y) \Rightarrow \exists y \forall x A(x,y)$
4. $\forall y \forall x A(x,y) \Rightarrow \exists x \forall y A(x,y)$
5. $\exists y \forall x A(x,y) \Rightarrow \forall x \exists y A(x,y)$
6. $\forall x \exists y A(x,y) \Rightarrow \exists y \exists x A(x,y)$
7. $\exists x \forall y A(x,y) \Rightarrow \forall y \exists x A(x,y)$
8. $\forall y \exists x A(x,y) \Rightarrow \exists x \exists y A(x,y)$