

# MATHEMATICAL LOGIC CLASS NOTE

## 1. PROPOSITIONAL LOGIC

A *propositional language* consists of

- (1) Connective symbols:  $\neg, \rightarrow$
- (2) Punctuation symbols:  $(, )$
- (3) non empty set  $\mathcal{L}$ .

Elements of  $\mathcal{L}$  are called *sentence symbols*. (Sentence symbols will be denoted by  $p, q, r, \dots$ ) An  $\mathcal{L}$ -*expression* is any finite sequence of symbols from  $\mathcal{L}$  together with connective and punctuation symbols. For example  $\neg(pqq) \rightarrow$  or  $((\neg(p \rightarrow q)) \rightarrow (\neg q))$  are both  $\mathcal{L}$ -expressions where  $p, q \in \mathcal{L}$ . (An expression will be denoted by a lower-case Greek letter.  $\text{Exp}(\mathcal{L})$  denotes the set of all  $\mathcal{L}$ -expressions. When underlying  $\mathcal{L}$  is clear, we often omit it.) Now let us single out a set of sentences among that of expressions.

**Definition 1.1.** An  $\mathcal{L}$ -expression  $\phi$  is called  $\mathcal{L}$ -sentence if there is a formation sequence of  $\phi$ . Namely, there is a finite sequence of expressions  $\phi_0, \dots, \phi_n$  with  $\phi_n = \phi$  such that for all  $i \leq n$ , either

- (1)  $\phi_i = p \in \mathcal{L}$ , or
- (2)  $\phi_i = (\neg\phi_j)$  for  $j < i$ , or
- (3)  $\phi_i = (\phi_j \rightarrow \phi_k)$  for  $j, k < i$ .

For example, in above the former one is not a sentence while the latter one is. From now on,  $\text{Sent}(\mathcal{L})$  denotes the set of all sentences of  $\mathcal{L}$ .

**Fact 1.2.** Suppose that  $\mathcal{S}'$  is a set of expressions containing  $\mathcal{L}$  and closed under connectives, i.e.

- (1)  $\mathcal{L} \subseteq \mathcal{S}'$ ,
- (2) if  $\varphi, \psi \in \mathcal{S}'$ , then both  $(\neg\varphi)$  and  $(\varphi \rightarrow \psi)$  are in  $\mathcal{S}'$ .

Then  $\text{Sent}(\mathcal{L}) \subseteq \mathcal{S}'$ , and  $\text{Sent}(\mathcal{L})$  also satisfies (1) and (2), i.e.  $\text{Sent}(\mathcal{L})$  is the smallest set among all the sets of expressions satisfying (1) and (2).

---

This note is basically a summary of Chapter 1,2 of "A mathematical Introduction to Logic" (2nd Edn., Academic Press, 2001) by Herbert B. Enderton.

The fact is frequently used when we prove that a certain property holds for all sentences, by showing that the set of expressions satisfying the property contains  $\mathcal{L}$  and closed under connectives.

### Exercise

- (1) Show that no sentence begins with  $\neg$ .
- (2) For a sentence  $\varphi$ , show that  $l(\varphi) = r(\varphi)$  where  $l(\varphi)$  ( $r(\varphi)$ , resp.) is the number of left (right, resp.) parenthesis in  $\varphi$ .
- (3) Let  $\varphi$  be a sentence of length  $n$ . Show that for  $1 \leq k < n$ ,  $r(\varphi, k) < l(\varphi, k)$ , where  $l(\varphi, k)$  ( $r(\varphi, k)$ , resp.) is the number of left (right, resp.) parenthesis among the first  $k$  symbols of  $\varphi$ .

### • Unique Readability

Each sentence  $\chi$  (of  $\mathcal{L}$ ) has one and only one of following forms:

- (1)  $p$  for  $p \in \mathcal{L}$ ,
- (2)  $(\neg\varphi)$  for unique sentence  $\varphi$ ,
- (3)  $(\phi \rightarrow \psi)$  for unique sentences  $\phi, \psi$ .

(Use previous Exercise to verify this.)

Unique Readability allows us to *recursively* define ‘unique’ function  $f$  on  $\text{Sent}(\mathcal{L})$  from the following information:

- (1) The rule of assigning  $f(p)$  for  $p \in \mathcal{L}$  is given.
- (2) The rule of assigning  $f(\neg\psi)$  from  $f(\psi)$  is given.
- (3) The rule of assigning  $f(\phi \rightarrow \varphi)$  from  $f(\phi), f(\varphi)$  is given.

For example, there is a unique rank function  $rk : \text{Sent}(\mathcal{L}) \rightarrow \omega$  satisfying the following:

- (1)  $rk(p) = 0$  for  $p \in \mathcal{L}$ ,
- (2)  $rk(\neg\psi) = 1 + rk(\psi)$ ,
- (3)  $rk(\phi \rightarrow \varphi) = 1 + \max\{rk(\phi), rk(\varphi)\}$ .

### • Convention of Abbreviation

Drop outermost pair of parentheses.

Drop further by accepting ‘linking’ conventions:  $\neg$  links more strongly than  $\rightarrow, \vee, \wedge, \leftrightarrow$ , and  $\vee, \wedge$  link more strongly than  $\rightarrow, \leftrightarrow$ , where

$$\alpha \vee \beta := \neg\alpha \rightarrow \beta$$

$$\alpha \wedge \beta := \neg(\alpha \rightarrow \neg\beta)$$

$$\alpha \leftrightarrow \beta := (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha).$$

For example,  $\neg\varphi \vee \psi \rightarrow \chi$  is  $((\neg\varphi) \vee \psi) \rightarrow \chi$ .

Additionally, when  $\rightarrow$  is used repeatedly, grouping is to the right:  $p \rightarrow q \rightarrow r \rightarrow s$  is  $p \rightarrow (q \rightarrow (r \rightarrow s))$ .

- Truth Assignment

A *truth assignment*  $\nu$  for a language  $\mathcal{L}$  is a function  $\nu : \mathcal{L} \rightarrow \{T, F\}$ . By above, there is a unique extension  $\bar{\nu}$  of  $\nu$  on  $\text{Sent}(\mathcal{L})$  such that

$$\begin{aligned}\bar{\nu}(\neg\varphi) &= T \text{ iff } \bar{\nu}(\varphi) = F, \\ \bar{\nu}(\varphi \rightarrow \psi) &= F \text{ iff } \bar{\nu}(\varphi) = T \text{ and } \bar{\nu}(\psi) = F.\end{aligned}$$

A sentence  $\alpha$  is said to be *tautology* if  $\bar{\nu}(\alpha) = T$  for every truth assignment  $\nu$ . A set  $\Sigma$  of sentences (or a single sentence  $\alpha$ ) is said to be *satisfiable* if for some truth assignment  $\nu$ ,  $\bar{\nu}(\alpha) = T$  for all  $\alpha \in \Sigma$ . Sentences  $\alpha, \beta$  are called *tautologically equivalent* if  $\bar{\nu}(\alpha) = \bar{\nu}(\beta)$  for every truth assignment  $\nu$ .

**Remark 1.3.** (1)  $\alpha$  is not satisfiable iff  $\neg\alpha$  is tautology. (But a satisfiable sentence need not be tautology.)  
(2)  $\alpha, \beta$  are tautologically equivalent iff  $\alpha \leftrightarrow \beta$  is a tautology.  
(3) De Morgan's law:  $\neg(\varphi \wedge \psi) \leftrightarrow \neg\varphi \vee \neg\psi$ ;  $\neg(\varphi \vee \psi) \leftrightarrow \neg\varphi \wedge \neg\psi$  are both tautology.

A *literal* is a sentence which is a sentence symbol, or the negation of a sentence symbol. We say a sentence  $\sigma$  is in *disjunctive normal form* (d.n.f.) if  $\sigma$  is a disjunction of conjunction of literals, e.g.  $(p \wedge \neg q \wedge r) \vee \neg r \vee (s \wedge p)$ .

**Theorem 1.4.** For any sentence  $\varphi$ , there is a sentence  $\psi$  in d.n.f. such that  $\varphi, \psi$  are tautologically equivalent.

A *Boolean function* is an operation on  $\{T, F\}$ , i.e. a mapping from  $\{T, F\}^n$  to  $\{T, F\}$  for some  $n$ . We say a Boolean function  $h : \{T, F\}^n \rightarrow \{T, F\}$  is *realized* by a sentence  $\varphi$ , if  $h(\alpha_1, \dots, \alpha_n) = \bar{\nu}_{\alpha_1 \dots \alpha_n}(\varphi)$ , where  $\nu_{\alpha_1 \dots \alpha_n}(A_i) = \alpha_i$  ( $\alpha_i \in \{T, F\}$ ,  $A_i$  is the  $i$ th sentence symbol in  $\varphi$ ). A sentence realizes a Boolean function. Conversely, by the proof of previous theorem, every Boolean function is realized by some sentence in d.n.f.

A set  $H$  of connectives is called *complete* if every Boolean function is realized by some sentence with all connectives in  $H$ , equivalently if every sentence is tautologically equivalent to a sentence with all connectives in  $H$ .

**Corollary 1.5.**  $\{\neg, \wedge, \vee\}$ ,  $\{\neg, \wedge\}$ ,  $\{\neg, \vee\}$  are all complete.

The connective symbol  $|$  is called *nand*, and  $\varphi | \psi$  means  $\neg(\varphi \wedge \psi)$ . The symbol  $\downarrow$  is called *nor*, and  $\varphi \downarrow \psi$  means  $\neg(\varphi \vee \psi)$ .

**Corollary 1.6.** *The connectives  $\{\downarrow\}$ ,  $\{\mid\}$  are all complete.*

- Compactness

Recall a set  $\Sigma$  of sentences is said to be *satisfiable* if for some truth assignment  $\nu$ ,  $\bar{\nu}(\alpha) = T$  for all  $\alpha \in \Sigma$ . The set  $\Sigma$  is *finitely satisfiable* if every finite subset of  $\Sigma$  is satisfiable.

**Theorem 1.7. (Compactness)** *For any set  $\Sigma$  of sentences,  $\Sigma$  is finitely satisfiable iff  $\Sigma$  is satisfiable.*

It suffices to show if  $\Sigma$  is finitely satisfiable, then  $\Sigma$  is satisfiable. The proof takes the following steps.

- (1) If  $\Gamma$  is a finitely satisfiable set of sentences and  $\sigma$  is a given sentence, then  $\Gamma \cup \{\sigma\}$  or  $\Gamma \cup \{\neg\sigma\}$  is finitely satisfiable.
- (2) Given finitely satisfiable  $\Sigma$ , there is finitely satisfiable  $\Delta (\supseteq \Sigma)$  such that for every sentence symbol  $p \in \mathcal{L}$ , either  $p \in \Delta$  or  $\neg p \in \Delta$ .
- (3) Define  $\nu : \mathcal{L} \rightarrow \{T, F\}$  such that,

$$\nu(p) = \begin{cases} T, & \text{if } p \in \Delta \\ F, & \text{if } \neg p \in \Delta. \end{cases}$$

Show that  $\bar{\nu}$  satisfies  $\Delta$ .

- Soundness and completeness

**Definition 1.8.** *Let  $\Sigma$  be a set of sentences, and  $\alpha$  be a sentence.*

- (1) *The notation  $\Sigma \models \alpha$  (called,  $\alpha$  is a tautological consequence of  $\Sigma$  (a semantic notion)) means that if  $\Sigma$  is satisfied by some truth assignment  $\nu$ , then the  $\nu$  satisfies  $\alpha$  too.*
- (2) *A deduction (or proof) from  $\Sigma$  is a finite sequence  $\langle \alpha_0, \dots, \alpha_m \rangle$  of sentences such that, for each  $i \leq m$ ,  $\alpha_i \in \Sigma$ , or  $\alpha_i$  is tautology, or  $\alpha_i$  follows from  $\alpha_j, \alpha_k$  for some  $j, k < i$  by Modus Ponens (i.e.  $\alpha_k$  is  $\alpha_j \rightarrow \alpha_i$ ).*
- (3) *The notation  $\Sigma \vdash \alpha$  (called,  $\alpha$  is provable from  $\Sigma$  (a syntactic notion)) means that there is a deduction  $\langle \alpha_0, \dots, \alpha_m \rangle$  from  $\Sigma$  with  $\alpha_m = \alpha$ .*

Similarly to Fact 1.2, one can see that  $\{\alpha \mid \Sigma \vdash \alpha\}$  is the smallest set among sets containing  $\Sigma$  with all tautologies and closed under Modus Ponens.

**Theorem 1.9. (Soundness and Completeness)**  $\Sigma \vdash \alpha$  iff  $\Sigma \models \alpha$ .

## 2. FIRST-ORDER LOGIC

A 1st-order language  $\mathcal{L}$  consists of

- (1) Logical symbols
  - (a) Punctuation symbols:  $(, )$
  - (b) Connectives:  $\neg, \rightarrow$
  - (c) Quantifier:  $\forall$
  - (d) Equality symbol:  $=$  (can be considered as binary predicate symbol.)
  - (e) Variables:  $v_0, v_1, v_2, \dots$
- (2) Nonlogical symbols
  - (a) For each  $n \geq 1$ , a set (possibly empty) of  $n$ -ary function symbols.
  - (b) A set (possibly empty) of constant symbols.
  - (c) For each  $n \geq 1$ , a set (possibly empty) of  $n$ -ary predicate symbols.

Every 1st-order language has the fixed logical symbols. Hence to specify  $\mathcal{L}$ , we may identify  $\mathcal{L}$  to be only the collection of nonlogical symbols. While that  $\mathcal{L}$  is finite (or finite language  $\mathcal{L}$ ) means  $\mathcal{L}$  has finitely many *nonlogical* symbols,  $Card(\mathcal{L})$  denotes the cardinality of  $\mathcal{L}$  including logical symbols, hence always infinite. We often omit to indicate  $\mathcal{L}$  when the language  $\mathcal{L}$  is fixed.

We define recursively *terms, atomic formulas and formulas* (of  $\mathcal{L}$ ).

Term:

- (1) Variable  $v_i$  and constant symbol  $c$  are terms.
- (2) If  $f$  is an  $n$ -ary function symbol, and  $\tau_1, \dots, \tau_n$  are terms, then  $f\tau_1\dots\tau_n$  is a term.
- (3) Nothing else is a term unless it can be obtained by finitely many applications of (1) and (2).

Atomic formula: Suppose that  $R$  is an  $n$ -ary predicate symbol (possibly the binary predicate  $=$ ), and  $\tau_1, \dots, \tau_n$  are terms. Then  $R\tau_1\dots\tau_n$  is an atomic formula.

Formula:

- (1) Every atomic formula is a formula.
- (2) If  $\alpha, \beta$  are formulas, then so are  $(\neg\alpha)$ ,  $(\alpha \rightarrow \beta)$  and  $\forall v_j\alpha$  ( $j \in \omega$ ).
- (3) Nothing else is a formula unless it can be obtained by finitely many applications of (1) and (2).

In the following, upper-case Greek letters denote sets of formulas of  $\mathcal{L}$ , lower-case Greek letters denote formulas of  $\mathcal{L}$ ;  $x, y, ..$  denote arbitrary variables,  $c, d, ..$  constant and  $\tau$ , or  $t, u, ..$  terms.

- Unique readability and abbreviation

Each term  $\tau$  (of  $\mathcal{L}$ ) has one and only one of following forms:

- (1) a unique constant  $c$ ;
- (2) variable  $v_i$  for unique  $i$ ;
- (3)  $f\tau_1\dots\tau_m$  for unique  $m$  and unique  $m$ -ary function symbol  $f$ , and unique terms  $\tau_1, \dots, \tau_m$ .

Each formula  $\chi$  (of  $\mathcal{L}$ ) has one and only one of following forms:

- (1) an atomic formula  $P\tau_1\dots\tau_n$  for unique  $n$ , and unique  $n$ -ary predicate  $P$ , and unique terms  $\tau_1, \dots, \tau_n$ ;
- (2)  $(\neg\varphi)$  for unique formula  $\varphi$ ;
- (3)  $(\phi \rightarrow \psi)$  for unique formulas  $\phi, \psi$ ;
- (4)  $\forall v_i\varphi$  for unique  $i$  and  $\varphi$ .

Unique readability enables us to define recursively unique function on the set of all terms, or formulas (of  $\mathcal{L}$ ).

Given formula  $\varphi$ , assign a set  $fv(\varphi)$  (can be  $\emptyset$ ) of variables as follows:

- (1) If a formula  $\alpha$  is atomic, then  $fv(\alpha)$  is a set of all variables in  $\varphi$ .
- (2)  $fv((\neg\beta)) = fv(\beta)$
- (3)  $fv((\alpha \rightarrow \beta)) = fv(\alpha) \cup fv(\beta)$
- (4)  $fv(\forall v_i\alpha) = fv(\alpha) \setminus \{v_i\}$ .

If  $x \in fv(\alpha)$ , then we say that the variable  $x$  *occurs free* (or *is free*) in  $\alpha$ . A *sentence* is a formula having no free variables. We often write  $\varphi(x_1, \dots, x_n)$  to represent  $fr(\varphi) = \{x_1, \dots, x_n\}$  in order.

For abbreviation of notation, first follow the rule described in Propositional Logic.

Write  $\tau = \mu$  (or  $\tau \neq \mu$ ) instead of  $= \tau\mu$  (or  $\neg = \tau\mu$ ), and also for other familiar binary predicate or function symbols such as  $<, \times, \dots$

$\exists x\varphi$  stands for  $\neg\forall x(\neg\varphi)$ .

$\forall x, \exists x$  link more strongly than binary connectives. For example,  $\forall x\neg\varphi \rightarrow \psi$  is  $(\forall x(\neg\varphi) \rightarrow \psi)$ , not  $\forall x((\neg\varphi) \rightarrow \psi)$ .

$\forall x_1\dots x_n\varphi$  stands for  $\forall x_1\dots\forall x_n\varphi$  (similarly to  $\exists x_1\dots x_n\varphi$ ).

### 3. STRUCTURE (OR MODEL) AND SATISFACTION

Let  $\mathcal{L}$  be a first-order language. An  $\mathcal{L}$ -*structure* (or *model*)  $M$  is a nonempty set (denoted by  $|M|$ ), called the *universe* of the model  $M$ , equipped with the following interpretations for non-logical symbols.

- (1) For each constant symbol  $c$ , there corresponds an element  $c^M \in |M|$ .
- (2) For each  $n$ -ary function symbol  $f$ , there corresponds an  $n$ -ary function  $f^M : |M|^n \rightarrow |M|$ .

(3) For each  $n$ -ary predicate symbol  $P$ , there corresponds an  $n$ -ary relation  $P^M \subseteq |M|^n$ .

For equality symbol  $=$  in  $\mathcal{L}$ ,  $=^M$  must be an equality relation in  $|M|$ , i.e.  $=^M = \{(a, b) \in |M|^2 : a = b\}$ . For  $n$ -ary predicate  $P$  and  $a_1, \dots, a_n \in |M|$ ,  $P^M(a_1, \dots, a_n)$  means  $(a_1, \dots, a_n) \in P^M$ .

For example, for the language of ordered field  $\mathcal{L}_{orf} = \{+, \times, 0, 1, <\}$  with binary function symbols  $+$ ,  $\times$ , two constant symbols  $0$ ,  $1$ , and binary predicate  $<$ , the real ordered field  $\mathbb{R}$  with canonical interpretations is an  $\mathcal{L}_{orf}$ -model. In fact  $M = (\{a, b\}, +^M, \times^M, 0^M, 1^M, <^M)$  with some functions  $+^M, \times^M : \{a, b\}^2 \rightarrow \{a, b\}$ , some subset  $<^M \subseteq \{a, b\}^2$  and say  $0^M = a, 1^M = b$ , is an  $\mathcal{L}_{orf}$ -model too.

Now we shall define the notation  $M \models \varphi[s]$  ( $\varphi$  is realized (or satisfied) by  $s$  in  $M$ , or  $M$  satisfies  $\varphi$  with  $s$ ) for ( $\mathcal{L}$ -)structure  $M$ , formula  $\varphi$ , and  $s : \mathcal{V} \rightarrow |M|$  where  $\mathcal{V} = \{v_0, v_1, \dots\}$ , the set of all variables.

First, let us extend the function  $s$  to  $\bar{s}$  on  $\mathcal{T}$ , the set of all terms:

- (1)  $\bar{s}(v_i) = s(v_i)$ .
- (2)  $\bar{s}(c) = c^M$ .
- (3)  $\bar{s}(f\tau_1\dots\tau_n) = f^M(\bar{s}(\tau_1), \dots, \bar{s}(\tau_n))$ .

Then define  $M \models \varphi[s]$  by recursion on formulas.

- (1) If  $\alpha$  is atomic, i.e.  $P\tau_1\dots\tau_n$ , then  $M \models \alpha[s]$  iff  $P^M(\bar{s}(\tau_1), \dots, \bar{s}(\tau_n))$
- (2)  $M \models \neg\beta[s]$  iff not  $M \models \beta[s]$ .
- (3)  $M \models (\alpha \rightarrow \beta)[s]$  iff (not  $M \models \alpha[s]$ ) or ( $M \models \beta[s]$ ).
- (4)  $M \models \forall v_i\varphi[s]$  iff for every  $d \in |M|$ ,  $M \models \varphi[s(v_i|d)]$  where  $s(v_i|d) : \mathcal{V} \rightarrow |M|$  such that  $s, s(v_i|d)$  agree on  $\mathcal{V} \setminus \{v_i\}$  and  $s(v_i|d)(v_i) = d$ .

**Proposition 3.1.** *If  $s_1, s_2$  are maps from  $\mathcal{V}$  to a model  $M$ , and agree on all free variables in a formula  $\varphi$ , then  $M \models \varphi[s_1]$  iff  $M \models \varphi[s_2]$ .*

The proposition is proved using routine induction on formulas. By the proposition, without ambiguity we often write  $M \models \varphi[a_1, \dots, a_n]$  instead of  $M \models \varphi[s]$  where  $\varphi = \varphi(x_1, \dots, x_n)$  and  $s(x_i) = a_i$ . (Then this is more natural notation since  $M \models \varphi[a_1, \dots, a_n]$  means informally that  $\varphi$  holds in  $M$  when each free variable is replaced by the element in  $|M|$ .) For example the real field  $\mathbb{R} \models \exists x(v_0 + x \cdot x = v_1)[-1, 2]$ , but  $\mathbb{R} \not\models \exists x(v_0 + x \cdot x = v_1)[2, -1]$ .

A formula  $\varphi$  is said to be *valid* if for every model and function  $s$ ,  $M \models \varphi[s]$ . We say a set  $\Gamma$  of formulas is *satisfiable* if for some model  $M$  and  $s$ ,  $M$  satisfies (every member of)  $\Gamma$  with  $s$ . Now for a sentence  $\sigma$  and a model  $M$ , we can write  $M \models \sigma$  when  $M \models \sigma[s]$  for any (some)  $s$ . Hence for any model  $M$  and sentence  $\sigma$ , either  $M \models \sigma$  ( $\sigma$  is true in  $M$ ) or  $M \models \neg\sigma$ . When  $\Sigma$  is a set of *sentences*, we can say  $M$  is a model of  $\Sigma$  if  $M$  satisfies every member of  $\Sigma$ .

**Definition 3.2.** Let  $\Gamma$  be a set of ( $\mathcal{L}$ -)formulas, and  $\psi$  a formula. Then  $\Gamma \models \psi$  ( $\psi$  is a logical consequence of  $\Gamma$  (semantic notion)) means that for every ( $\mathcal{L}$ -)model  $M$  and every  $s : \mathcal{V} \rightarrow |M|$ , whenever  $M \models \phi[s]$  for all  $\phi \in \Gamma$ , then  $M \models \psi[s]$ . (In other words, if  $M$  satisfies  $\Gamma$  with  $s$  then so does  $\psi$ .)

Given a set of  $\mathcal{L}$ -sentence  $\Sigma$ ,  $Mod(\Sigma)$  denotes the class of all  $\mathcal{L}$ -models of  $\Sigma$ . A class of  $\mathcal{L}$ -structures  $\mathcal{M}$  is called an *elementary class* (EC) if  $\mathcal{M} = Mod(\sigma)$  for some sentence  $\sigma$ .  $\mathcal{M}$  is called an *elementary class in the wider sense* ( $EC_\Delta$ ) if  $\mathcal{M} = Mod(\Sigma)$  for some set of sentences  $\Sigma$ . The class of groups, rings, fields (in the standard languages) are all EC. The class of infinite groups, fields of characteristic 0, algebraically closed fields are  $EC_\Delta$ . On the other hand, we shall see that the class of finite groups, countable groups, or well-orderings are *not*  $EC_\Delta$ .

- Embedding

We first introduce very important notion of a *definable set* in a model. Fix a language  $\mathcal{L}$  for the definition.

**Definition 3.3.** We say a subset  $D \subseteq |M|^k$  is definable in the model  $M$  if there is a formula  $\varphi(x_1, \dots, x_k)$  such that  $D = \{(a_1, \dots, a_k) \in |M|^k : M \models \varphi[a_1, \dots, a_k]\}$ .

For example, in the language of field  $\mathcal{L}_{field} = \{+, \cdot, 0, 1\}$ , the order relation  $<$  in the real field is defined by  $\exists x(v_0 + x \cdot x = v_1)$ .

Now we study morphisms between two  $\mathcal{L}$ -models  $M$  and  $N$ .

**Definition 3.4.** A function  $h : |M| \rightarrow |N|$  is said to be embedding (or homomorphism) if  $h$  preserves all non-logical symbols, i.e.

- (1)  $h(c^M) = c^N$  for any constant  $c$ ,
- (2) for any predicate symbol  $P$  (including  $=$ ) and elements  $a_1, \dots, a_n \in |M|$ ,  $P^M(a_1, \dots, a_n)$  iff  $P^N(h(a_1), \dots, h(a_n))$ ,
- (3) for any function symbol  $f$  and elements  $a_1, \dots, a_n \in |M|$ ,  $h(f^M(a_1, \dots, a_n)) = f^N(h(a_1), \dots, h(a_n))$ .

Every embedding is injective ( $a =^M b$  iff  $h(a) =^N h(b)$ ). Embedding  $h$  is called an *isomorphism* between  $M$  and  $N$  if  $h$  is onto; then  $M, N$  are called isomorphic, denoted by  $M \cong N$ . If  $h : |M| \rightarrow |M|$  is isomorphism,  $h$  is called an *automorphism* of  $M$ .

For  $\mathcal{L}$ -structures  $M, N$ , we say  $M$  is a *substructure* of  $N$  (denoted by  $M \subseteq N$ ) if  $|M| \subseteq |N|$  as sets, and the inclusion map is embedding. By the definition of embedding,  $M \subseteq N$  if and only if  $|M| \subseteq |N|$ ; for any constant  $c$ ,  $c^M = c^N$ ; for any  $n$ -ary predicate  $P$ ,  $P^M = P^N \cap |M|^n$ ; for any  $n$ -ary function symbol  $f$ ,  $f^M = f^N \upharpoonright |M|^n$ .

**Theorem 3.5.** (1)  $h : |M| \rightarrow |N|$  is embedding if and only if for any quantifier free  $\varphi(x_1, \dots, x_n)$  and  $a_1, \dots, a_n \in |M|$ ,  $M \models \varphi[a_1, \dots, a_n]$  iff  $N \models \varphi[h(a_1), \dots, h(a_n)]$ .  
(2) (Isomorphism Theorem)  $h : |M| \rightarrow |N|$  is an isomorphism, then for any formula  $\varphi(x_1, \dots, x_n)$  and  $a_1, \dots, a_n \in |M|$ ,  $M \models \varphi[a_1, \dots, a_n]$  iff  $N \models \varphi[h(a_1), \dots, h(a_n)]$ .

**Corollary 3.6.** (1)  $M \cong N$  then  $M \models \varphi$  iff  $N \models \varphi$ .  
(2) If  $D \subseteq |M|^k$  is definable set, and  $h : |M| \rightarrow |M|$  is automorphism, then  $h(D) = D$  (i.e.  $h$  preserves  $D$  set-wise).

- Reduct of a model

Let a language  $\mathcal{L}$  be a subset of a language  $\mathcal{L}'$ . If  $M$  is an  $\mathcal{L}'$ -structure, then *the reduct of  $M$  to  $\mathcal{L}$*  (or  $\mathcal{L}$ -reduct of  $M$ ), denoted by  $M[\mathcal{L}]$ , is simply the structure with the same universe as  $M$  forgetting interpretations of symbols in  $\mathcal{L}' \setminus \mathcal{L}$ . Hence the reduct  $M[\mathcal{L}]$  is an  $\mathcal{L}$ -structure. Conversely, we call  $M$   $\mathcal{L}'$ -expansion of  $M[\mathcal{L}]$ . For  $s : \mathcal{V} \rightarrow |M| (= |M[\mathcal{L}]|)$ , and  $\mathcal{L}$ -formula  $\varphi$ , clearly  $M \models \varphi$  (in  $\mathcal{L}'$ ) iff  $M[\mathcal{L}] \models \varphi$  (in  $\mathcal{L}$ ).

#### 4. DEDUCTION (OR PROOF)

Fix a first order language  $\mathcal{L}$ .

The set  $\Lambda$  of *logical axioms* (of  $\mathcal{L}$ ) is the set of all generalizations ( $\forall x_1 \dots \forall x_n \varphi$  for some  $n \geq 0$ ) is called a *generalization* of  $\varphi$  of the following forms:

- (1) Tautologies: In propositional logic, if we take 1st-order  $\mathcal{L}$ -prime formulas (= either atomic formulas or of the form  $\forall v_i \varphi$ ) as sentence symbols, then every  $\mathcal{L}$ -formula  $\varphi$  can be considered as a propositional sentence with the sentence symbols. The formula  $\varphi$  is said to be a *tautology* if it is so as the propositional sentence.
- (2)  $\forall x \alpha \rightarrow \alpha_t^x$  where the term  $t$  is substitutable for  $x$  in  $\alpha$ .

Terminology:

- (a)  $\alpha_t^x$  is the result of replacing each free occurrence of  $x$  in  $\alpha$  by the term  $t$ . More formally, first define  $u_t^x$  for a term  $u$ :

- (i) For constant  $c$ ,  $c_t^x = c$ ; for variable  $y$ ,  $y_t^x = \begin{cases} t & \text{if } x = y, \\ y & \text{otherwise.} \end{cases}$
- (ii) For  $f u_1 \dots u_n$ ,  $(f u_1 \dots u_n)_t^x = f(u_1)_t^x \dots (u_n)_t^x$ .

Now, define  $\varphi_t^x$  recursively:

- (i) For an atomic formula  $P t_1 \dots t_n$ ,  $(P t_1 \dots t_n)_t^x = P(t_1)_t^x \dots (t_n)_t^x$ .
- (ii)  $(\neg \alpha)_t^x = (\neg \alpha_t^x)$ ;  $(\alpha \rightarrow \beta)_t^x = (\alpha_t^x \rightarrow \beta_t^x)$ .
- (iii)  $(\forall y \alpha)_t^x = \begin{cases} \forall y \alpha & \text{if } x = y, \\ \forall y (\alpha_t^x) & \text{if } x \neq y. \end{cases}$

- (b) Informally  $t$  is *not* substitutable for  $x$  in  $\alpha$  if  $x$  is free in  $\alpha$  and there is some variable  $y$  in  $t$  which is captured by quantifier  $\forall y$  in  $\alpha_t^x$ . Formally,
- (i) for atomic  $\alpha$ ,  $t$  is always substitutable for  $x$  in  $\alpha$ .
  - (ii)  $t$  is substitutable for  $x$  in  $\neg\alpha$  iff  $t$  is substitutable for  $x$  in  $\alpha$ ;  $t$  is substitutable for  $x$  in  $\alpha \rightarrow \beta$  iff  $t$  is substitutable for  $x$  both in  $\alpha, \beta$ .
  - (iii)  $t$  is substitutable for  $x$  in  $\forall y\alpha$  iff either
    - (A)  $x$  does not occur free in  $\forall y\alpha$ , or
    - (B)  $y$  is not in  $t$  and  $t$  is substitutable for  $x$  in  $\alpha$ .
- (3)  $\forall x(\alpha \rightarrow \beta) \rightarrow (\forall x\alpha \rightarrow \forall x\beta)$ .
- (4)  $\alpha \rightarrow \forall x\alpha$  where  $x$  not free in  $\alpha$ .
- (5)  $x = x$ .
- (6)  $x = y \rightarrow (\alpha \rightarrow \alpha')$  where  $\alpha$  is an atomic formula of the form

$$Pz_1 \cdots z_k \text{ or } fz_1 \cdots z_k = fw_1 \cdots w_k$$

( $z_i, w_i$  are variables), and  $\alpha'$  is obtained from  $\alpha$  by replacing some occurrences of  $x$  (from  $\{z_1, \dots, z_k, w_1, \dots, w_k\}$ ) by  $y$ .

The following is obviously expected.

**Theorem 4.1.** *Every logical axiom is valid.*

It is not difficult to show the theorem except for axioms of form (2). For those, we need the following lemma.

**Lemma 4.2. (Substitution lemma)** *If  $t$  is substitutable for  $x$  in  $\varphi$ , then for any model  $M$  and  $s : \mathcal{V} \rightarrow |M|$ ,*

$$M \models \varphi_t^x[s] \text{ iff } M \models \varphi[s(x|\bar{s}(t))].$$

Now we are ready to define *deduction*.

**Definition 4.3.** *We say  $\varphi$  is deducible from (or provable from, or a theorem of)  $\Gamma$ , denoted by  $\Gamma \vdash \varphi$ , if there is a sequence of formulas  $\langle \varphi_0, \dots, \varphi_n \rangle$ , called deduction (or proof) of  $\varphi$  from  $\Gamma$ , such that  $\varphi_n$  is  $\varphi$  and for each  $i \leq n$ , either  $\varphi_i \in \Gamma \cup \Lambda$  or for some  $j, k < i$ ,  $\varphi_i$  is obtained by modus ponens (MP) from  $\varphi_j, \varphi_k$  (i.e.  $\varphi_k$  is  $\varphi_j \rightarrow \varphi_i$ ).*

Induction on Theorems: If  $\Gamma \cup \Lambda \subseteq \Sigma$  and  $\Sigma$  is closed under MP (i.e.  $\alpha \in \Sigma$  and  $(\alpha \rightarrow \beta) \in \Sigma$  implies  $\beta \in \Sigma$ ), then  $\{\alpha : \Gamma \vdash \alpha\} \subseteq \Sigma$ .

- Metatheorems on Deduction

Rule T: If  $\Gamma \vdash \alpha_0, \dots, \alpha_n$  and  $\alpha$  is *tautological consequence* of  $\alpha_0, \dots, \alpha_n$ , (i.e. any truth assignment satisfying  $\alpha_0, \dots, \alpha_n$  also satisfies  $\alpha$ ) then  $\Gamma \vdash \alpha$ .

Corollary of Rule T:  $\Gamma \vdash \alpha$  iff  $\alpha$  is a tautological consequence of some finite subset of  $\Gamma \cup \Lambda$ .

Deduction Theorem and its converse:  $\Gamma \cup \{\alpha\} \vdash \beta$  iff  $\Gamma \vdash \alpha \rightarrow \beta$ .

Contraposition:  $\Gamma \cup \{\alpha\} \vdash \beta$  iff  $\Gamma \cup \{\neg\beta\} \vdash \neg\alpha$ .

Reductio ad Absurdum:  $\Gamma \vdash \alpha$  iff  $\Gamma \cup \{\neg\alpha\}$  is inconsistent. (A set of formula is *inconsistent* if some formula and its negation are both deducible from the set.)

Generalization Theorem: If  $\Gamma \vdash \alpha$  and  $x$  not free in  $\Gamma$ , then  $\Gamma \vdash \forall x\alpha$ .

Specialization Theorem: If  $\Gamma \vdash \forall x\alpha$  and  $t$  is substitutable for  $x$  in  $\alpha$ , then  $\Gamma \vdash \alpha_t^x$ .

$\exists$ -elimination: If  $\Gamma \cup \{\exists x\alpha\} \vdash \beta$ , then  $\Gamma \cup \{\alpha\} \vdash \beta$ .

If  $\Gamma \cup \{\alpha\} \vdash \beta$  and  $x$  not free in  $\Gamma \cup \{\beta\}$ , then  $\Gamma \cup \{\exists x\alpha\} \vdash \beta$ .

$\exists$ -introduction: If  $\Gamma \vdash \alpha_t^x$  and  $t$  is substitutable for  $x$  in  $\alpha$ , then  $\Gamma \vdash \exists x\alpha$ .

Generalization on constants (Form 1): Suppose  $\Gamma \vdash \alpha$  and  $c$  constant not in  $\Gamma$ . Let  $\langle \alpha_0, \dots, \alpha_n \rangle$  be a deduction of  $\alpha (= \alpha_n)$  from  $\Gamma$ , and let  $y$  be a variable not in any of  $\alpha_i$ . Then  $\langle (\alpha_0)_y^c, \dots, (\alpha_n)_y^c \rangle$  is a deduction. Hence  $\Gamma \vdash \forall y\alpha_y^c$  and there is a deduction of  $\forall y\alpha_y^c$  from  $\Gamma$  where  $c$  does not occur.

Generalization on constants (Form 2): Suppose  $\Gamma \vdash \alpha_c^x$  where  $c$  is not in  $\Gamma \cup \{\alpha\}$ . Then  $\Gamma \vdash \forall x\alpha$  and there is a deduction of  $\forall x\alpha$  from  $\Gamma$  where  $c$  does not occur.

Rule of EI (Dual of Form 2): Assume that  $c$  is not in  $\Gamma \cup \{\alpha, \beta\}$ , and that  $\Gamma \cup \{\alpha_c^x\} \vdash \beta$ . Then  $\Gamma \cup \{\exists x\alpha\} \vdash \beta$  via a deduction in which  $c$  does not occur.

Equality Theorem:

- (1)  $\vdash x = x$ .
- (2)  $\vdash x = y \rightarrow y = x$ .
- (3)  $\vdash x = y \rightarrow y = z \rightarrow x = z$ .
- (4)  $\vdash x_0 = y_0 \rightarrow x_1 = y_1 \rightarrow \dots \rightarrow x_n = y_n \rightarrow \alpha \rightarrow \alpha'$  where  $\alpha$  is an atomic formula of the form  $Pz_1 \cdots z_k$  or  $fz_1 \cdots z_k = fw_1 \cdots w_k$ , and  $\alpha'$  is obtained from  $\alpha$  by replacing some occurrences of  $x_i$  by  $y_i$

Corollary of Equality Theorem: Let some set of formulas  $\Delta$  be given. For terms  $t, u$ , let  $t \simeq u$  mean that  $\Delta \vdash t = u$ . Then  $\simeq$  is an equivalence relation on the set of all terms. Moreover if  $t_i \simeq u_i$  ( $i = 1, \dots, n$ ), then for  $n$ -ary function symbol  $f$ ,  $ft_1 \dots t_n \simeq fu_1 \dots u_n$ , and for  $n$ -ary predicate symbol  $P$ ,  $\Delta \vdash Pt_1 \dots t_n \leftrightarrow Pu_1 \dots u_n$ .

## 5. COMPLETENESS THEOREM

The goal of this section is to show that  $\Gamma \models \varphi$  iff  $\Gamma \vdash \varphi$ .

**Theorem 5.1. (Soundness)** *If  $\Gamma \vdash \varphi$  then  $\Gamma \models \varphi$ .*

Easy consequence of Theorem 4.1.

**Corollary 5.2.** *If  $\Gamma$  is satisfiable, then  $\Gamma$  is consistent.*

**Theorem 5.3. (Gödel's Completeness Theorem)** *If  $\Gamma \models \varphi$  then  $\Gamma \vdash \varphi$ .*

Completeness Theorem is equivalent to the following.

**Theorem 5.4.** *If  $\Gamma$  is consistent, then  $\Gamma$  is satisfiable.*

For Henkin's proof of Theorem 5.4, we need the following preparatory lemmas.

**Lemma 5.5. (Lindenbaum's Lemma)** *Suppose that  $\Sigma$  is a consistent set of arbitrary  $\mathcal{L}$ -formulas. Then  $\Sigma$  can be extended to a maximal consistent set  $\Delta$  of  $\mathcal{L}$ -formulas (i.e.  $\Delta$  is consistent, and for any  $\mathcal{L}$ -formula  $\varphi$ , either  $\varphi \in \Delta$ , or  $\neg\varphi \in \Delta$ ).*

If  $\Sigma$  is maximal consistent in  $\mathcal{L}$ , then  $\Delta \vdash_{\mathcal{L}} \psi$  iff  $\psi \in \Delta$ .

**Lemma 5.6.** *Let  $\Sigma$  be consistent set of  $\mathcal{L}$ -formulas. Let  $\mathcal{L}'$  be  $\mathcal{L}$  together with some set of new constant symbols. Then  $\Sigma$  remains consistent in  $\mathcal{L}'$ .*

**Lemma 5.7.** *Let  $\Sigma$  be consistent set of  $\mathcal{L}$ -formulas. Then for a variable  $x$ , and a constant  $d \in \mathcal{L}$ ,  $\Sigma_d^x = \{\varphi_d^x \mid \varphi \in \Sigma\}$  is consistent, too.*

**Lemma 5.8. (Henkin's Lemma)** *Suppose that  $\Sigma$  is a consistent set of  $\mathcal{L}$ -formulas. Let  $\mathcal{L}' = \mathcal{L} \cup \mathcal{C}_0$  where  $\mathcal{C}_0$  is some infinite set of new constant symbols of cardinality  $\kappa = \text{Card}(\mathcal{L})$ . Then there is consistent set  $\Delta^*$  of  $\mathcal{L}'$ -formulas containing  $\Sigma$  having the Henkin property, namely for each  $\mathcal{L}'$ -formula  $\varphi$  and each variable  $x$ , there is a constant  $c \in \mathcal{L}'$  such that  $\exists x\varphi \rightarrow \varphi_c^x \in \Delta^*$ .*

We are ready to show 5.4. Suppose that a consistent set  $\Gamma$  of  $\mathcal{L}$ -formulas is given. By 5.6 and 5.7,  $\Gamma'$  obtained from  $\Gamma$  by replacing each free occurrence of  $v_i$  in any formula in  $\Gamma$  by a new constant  $d_i$  is still consistent in  $\mathcal{L}_1 = \mathcal{L} \cup \{d_i \mid i \in \omega\}$ . Each element in  $\Gamma'$  is then an  $\mathcal{L}_1$ -sentence. Now by Lindenbaum and Henkin, there is maximal consistent  $\Delta(\supseteq \Gamma')$  of  $\mathcal{L}'$ -sentences having the Henkin property for  $\mathcal{L}'$ -sentences (i.e. the Henkin property holds for sentences of the form  $\exists x\varphi$ ).

We construct a model  $M$  of  $\Delta$ . The universe of  $M$  is  $\mathcal{T}/\simeq$ , where  $\mathcal{T}$  is the set of all closed  $\mathcal{L}'$ -terms (terms having no variable), and  $\simeq$  is defined as in Corollary of Equality Theorem, previously. Now for a constant  $c \in \mathcal{L}'$ , let  $c^M = c/\simeq$ ; for a function symbol  $f$

and closed terms  $t_i$ ,  $f^M(t_1/\simeq, \dots, t_n/\simeq) = ft_1\dots t_n/\simeq$ ; for predicate  $P$ ,  $P^M(t_1/\simeq, \dots, t_n/\simeq)$  iff  $Pt_1\dots t_n \in \Delta$ . By Equality theorem the interpretations are well-defined, and  $M$  is an  $\mathcal{L}'$ -model.

**Lemma 5.9.** *For an  $\mathcal{L}'$ -formula  $\varphi$  having free variables among  $\{x_1, \dots, x_n\}$ , and closed terms  $t_1, \dots, t_n$ ,*

$$\varphi_{t_1\dots t_n}^{x_1\dots x_n} \in \Delta \text{ iff } M \models \varphi[s] \text{ where } s(x_i) = t_i/\simeq.$$

Then  $M[\mathcal{L}]$  satisfies  $\Gamma$  with  $s : \mathcal{V} \rightarrow |M|$  such that  $s(v_i) = d_i^M$ . Hence the proof of Completeness Theorem is completed.

## 6. COMPACTNESS AND ITS CONSEQUENCES

**Theorem 6.1. (Compactness Theorem)** *If a set of formulas  $\Gamma$  is finitely satisfiable, then  $\Gamma$  is satisfiable.*

**Theorem 6.2.** *Let infinite  $\kappa = \text{Card}(\mathcal{L})$ .*

- (1) (**Löwenheim-Skolem: Downward**) *Let  $\Sigma$  be a set of  $\mathcal{L}$ -formulas satisfied by some model. Then  $\Sigma$  has a model of cardinality  $\leq \kappa$ .*
- (2) (**Tarski: Upward**) *Let  $\Sigma$  be a set of  $\mathcal{L}$ -formulas satisfied by an infinite model. Then for any  $\lambda \geq \kappa$ ,  $\Sigma$  has a model of cardinality  $\lambda$ .*

In particular, for a countable (i.e. finite or countably infinite) language  $\mathcal{L}$ , every consistent set  $\Sigma$  of formulas has a countable model. If  $\Sigma$  has an infinite model, then for every infinite cardinal  $\lambda$ ,  $\Sigma$  has a model of cardinality  $\lambda$ .

A *theory* (in  $\mathcal{L}$ ) is a set of ( $\mathcal{L}$ -)sentences. If  $T$  is the theory, then  $Cn(T) = \{\sigma : \sigma \text{ is } \mathcal{L}\text{-sentence and } T \vdash \sigma\}$ . The theory  $T$  is said to be *complete* (in  $\mathcal{L}$ ) if  $Cn(T)$  is maximal consistent in  $\mathcal{L}$  (i.e.  $T$  is consistent and for any sentence  $\sigma$ , either  $T \vdash \sigma$  or  $T \vdash \neg\sigma$ .) Given two ( $\mathcal{L}$ -)theories  $S, T$ , we say  $S$  *axiomatizes*  $T$  (write  $S \vdash T$ ) if  $Cn(S) = Cn(T)$ .

For a class of  $\mathcal{L}$ -structures  $\mathcal{M}$ ,  $Th(\mathcal{M})$  denotes the set of all sentences true in every model of  $\mathcal{M}$ . For a model  $M$ ,  $Th(M)(= Th(\{M\}))$  is complete. Two  $\mathcal{L}$ -models  $M, N$  are said to be *elementarily equivalent* (write  $M \equiv N$ ) if  $Th(M) = Th(N)$ . If two models  $M, N$  are isomorphic, then clearly they are elementarily equivalent. Note that a theory  $T$  is complete iff  $T$  has a model and any two models of  $T$  are elementarily equivalent iff  $T$  has a model  $M$  such that  $Cn(T) = Th(M)$ .

**Definition 6.3.** *Let  $\kappa$  be a cardinal. A theory  $T$  is said to be  $\kappa$ -categorical (or categorical in  $\kappa$ ) if any two models of  $T$  of cardinality  $\kappa$  are isomorphic.*

**Theorem 6.4. (Łoś-Vaught Test)** *Let  $T$  be a theory in  $\mathcal{L}$ . Assume that every model of  $T$  is infinite, and  $T$  is  $\kappa$ -categorical for some infinite cardinal  $\kappa \geq \text{Card}(\mathcal{L})$ . Then  $T$  is complete.*

**Examples:**

- (1)  $T_\infty = \{\lambda_n \mid 2 \leq n\}$  where  $\lambda_n = \exists x_1 \dots x_n \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j$ ;
- (2)  $T_{DLO} = \{\forall x \neg(x < x); \forall xyz(x < y \rightarrow y < z \rightarrow x < z); \forall xy(x < y \vee x = y \vee y < x); \forall xy(x < y \rightarrow \exists z(x < z \wedge z < y)); \forall x \exists y(x < y) \wedge \forall x \exists y(y < x)\}$  in  $\mathcal{L} = \{<\}$ ;
- (3) For a prime number  $p$  or  $p = 0$ ,

$$T_{ACF_p} = \{\text{field axioms}\}$$

$$\cup \{\forall x_0 \dots x_n (x_n \neq 0 \rightarrow \exists y x_0 + x_1 \cdot y + x_2 \cdot y \cdot y + \dots + x_n \cdot y^n = 0) \mid 2 \leq n\}$$

$$\cup \begin{cases} \{p(= \underbrace{1 + \dots + 1}_p) = 0\} & \text{if } p \text{ is a prime number,} \\ \{p \neq 0 \mid p \text{ prime}\} & \text{if } p = 0 \end{cases}$$

$$\text{in } \mathcal{L}_{\text{field}} = \{+, \cdot, 0, 1\};$$

- (4) Nonstandard models of number theory;

**Theorem 6.5. (Morley's Categoricity Theorem)** *Let a theory  $T$  be complete in a countable language. If  $T$  is categorical in some uncountable cardinal  $\kappa$ , then  $T$  is categorical in every uncountable cardinal.*

**Vaught's Conjecture:** For each complete theory  $T$  in a countable language, there are either countably many or  $2^\omega$ -many non-isomorphic countable models.

## 7. TURING MACHINE AND CHURCH'S THESIS

- Turing machine

**Definition 7.1.** *Turing machine is a function  $M$  such that for some  $m, n (\geq 0)$ ,  $\text{dom}(M) \subseteq S \times A$  and  $\text{ran}(M) \subseteq S \times A \times \{R, L\}$ , where  $S = \{q_0, \dots, q_m\}$  (set of states) and  $A = \{b, 1, X_0, \dots, X_n\}$  (set of alphabets).*

We can interpret the Turing machine as a partial function  $f$  from  $\omega$  to  $\omega$  (i.e.  $\text{dom}(f) \subseteq \omega$ ) as follows: The input of the machine is a tape partitioned into (infinitely many) squares with a starting square singled out. Represent  $n$  on the tape as a string of  $n$  1's beginning from the starting square while the rest of the tape blank. The machine always starts to read the tape from the starting square with state  $q_0$ . Hence the first move of the tape depends on  $M(q_0, 1)$  or  $M(q_0, b)$  ( $b$  represents blank). Now if the machine reads some square of the tape  $x$  ( $x \in A$ ) with the state  $q_i$ , and value of  $M(q_i, x) = (q_j, x', y)$  ( $x' \in A$ , and  $y = R$  or  $L$ ), then the machine changes the state to  $q_j$ , writes  $x'$  on the square (while erasing  $x$ , of course) and

makes a movement to the right or left according to  $y$ . The machine halts when undefined (i.e.  $(q_i, x) \notin \text{dom}(M)$ ). When the machine stops with input  $n$ , the output number  $f(n)$  is the number of 1's from the starting square up to the first blank or  $X_i$ 's. The machine will not halt with the input  $n$  iff  $n \notin \text{dom}(f)$ . (According to this interpretation, if  $M = \emptyset$  then trivially  $M$  represents an identity function.)

For given  $k \in \omega$ , the above interpretation also interprets the machine  $M$  as partial function from  $\omega^k$  ( $\omega^0 = \{0\}$ ) to  $\omega$  with input string, beginning from the starting square, of  $n_1$  1's;  $X_1$ ;  $n_2$  1's; ...;  $X_{k-1}$ ;  $n_k$  1's, (representing  $(n_1, \dots, n_k) \in \omega^k$ ) while the rest of the tape blank.

With this interpretation, any Turing machine corresponds, for each  $k \in \omega$ , to a partial function from  $\omega^k$  to  $\omega$ . Clearly two Turing machines may represent the same function. If the domain of the function is entire, then the function is called a total function.

Example) Turing machine  $M$  computing the function  $f(n) = n + 1$ . Let  $A = \{b, 1\}$ ,  $S = \{q_0, q_1\}$ .  $M(q_0, 1) = (q_0, 1, R)$ ,  $M(q_0, b) = (q_1, 1, L)$ .

$M$  moves right until it comes to the first  $b$ , then changes this to 1 and goes left, then halts, as the stage  $q_1$  is undefined.

Example) Turing machine  $M$  computing the function  $f(n) = 2n$ . Let  $A = \{b, 1, X\}$ ,  $S = \{q_0, q_1, q_2\}$ .  $M$  consists of the following quintuples:  $(q_0, 1, q_0, X, R)$ ,  $(q_0, b, q_1, b, L)$ ,  $(q_1, 1, q_1, 1, L)$ ,  $(q_1, X, q_2, 1, R)$ ,  $(q_2, 1, q_2, 1, R)$ ,  $(q_2, b, q_1, 1, L)$ .

$M$  changes 1's to  $X$ 's, then writing additional 1 for each  $X$  while erasing the  $X$  to 1.<sup>1</sup>

Example) Turing machine  $M$  computing the function  $f(m, n) = m + n$ .  $A = \{b, 1, X\}$ ,  $S = \{q_0, \dots, q_3\}$ .  $M$  consists of the following quintuples:  $(q_0, 1, q_0, 1, R)$ ,  $(q_0, X, q_1, 1, R)$ ,  $(q_1, 1, q_1, 1, R)$ ,  $(q_1, b, q_2, b, L)$ ,  $(q_2, 1, q_3, b, L)$ .

Example) Turing machine  $M$  which halts if a blank tape is given, but does not halt if the tape starts with 1.  $A = \{b, 1\}$ ,  $S = \{q_0, q_1, q_2\}$ .  $M$  consists of the following:  $(q_0, b, q_2, b, R)$ ,  $(q_0, 1, q_1, 1, R)$ ,  $(q_1, 1, q_1, 1, R)$  and  $(q_1, b, q_1, 1, R)$ .

**Definition 7.2. (Turing)**  $f : \omega^k \rightarrow \omega$  is Turing computable if there is a Turing machine such that given input  $\bar{a} \in \omega^k$ , the machine halts after finitely many steps with output  $f(\bar{a})$ .

(As in general recursion theory, one can also call a partial function  $f$  ( $\text{dom}(f) \subseteq \omega^k$ ) Turing computable if there exists corresponding Turing Machine. But here we pay our attention only to the function whose domain is total ( $\text{dom}(f) = \omega^k$ )).

**Example: (Rado, 1962)** Let  $\mathcal{C}_n$  ( $n \geq 0$ ) be the collection of the Turing machine  $M$  such that

- (1) the set of alphabets  $A$  is contained in  $\{b, 1, X_0, \dots, X_n\}$ ,
- (2)  $S \subseteq \{q_0, \dots, q_n\}$ ,
- (3)  $M$  halts when input  $n$  is given.

---

<sup>1</sup>This idea suggested by Dale Lee, an undergraduate student in 2014 fall semester class, reduced the previous argument requiring 14 quintuples.

Then clearly  $\mathcal{C}_n$  is finite but non-empty. Now, let  $G_n = \{M(n) + 1 \mid M \in \mathcal{C}_n\}$ . Define the Busy beaver function  $Bb : \omega \rightarrow \omega$  such that  $Bb(n) = \text{maximum of } G_n$ . Then the Busy beaver function is *not* Turing computable.

- Recursive functions and sets

**Definition 7.3.** For  $R \subseteq \omega^n$  a relation,  $\chi_R : \omega^n \rightarrow \omega$ , the characteristic function on  $R$ , is given by

$$\chi_R(\bar{a}) = \begin{cases} 1 & \text{if } \neg R(\bar{a}), \\ 0 & \text{if } R(\bar{a}). \end{cases}$$

**Definition 7.4.** A function from  $\omega^m$  to  $\omega$  ( $m \geq 0$ ) is called recursive (or computable) if it is obtained by finitely many applications of the following rules:

- R1. •  $I_i^n : \omega^n \rightarrow \omega$ ,  $1 \leq i \leq n$ , defined by  $(x_1, \dots, x_n) \mapsto x_i$  is recursive;  
•  $+$  :  $\omega \times \omega \rightarrow \omega$  and  $\cdot$  :  $\omega \times \omega \rightarrow \omega$  are recursive;  
•  $\chi_{<} : \omega \times \omega \rightarrow \omega$  is recursive.
- R2. (Composition) For recursive functions  $G, H_1, \dots, H_k$  such that  $H_i : \omega^n \rightarrow \omega$  and  $G : \omega^k \rightarrow \omega$ ,  $F : \omega^n \rightarrow \omega$ , defined by

$$F(\bar{a}) = G(H_1(\bar{a}), \dots, H_k(\bar{a})).$$

is recursive.

- R3. (Minimization) For  $G : \omega^{n+1} \rightarrow \omega$  recursive, such that for all  $\bar{a} \in \omega^n$  there exists some  $x \in \omega$  such that  $G(\bar{a}, x) = 0$ ,  $F : \omega^n \rightarrow \omega$ , defined by

$$F(\bar{a}) = \mu x (G(\bar{a}, x) = 0)$$

is recursive. (Recall that  $\mu x P(x)$  for a relation  $P$  is the minimal  $x \in \omega$  such that  $x \in P$  obtains.)

**Definition 7.5.**  $R(\subseteq \omega^k)$  is called recursive, or computable ( $R$  is a recursive relation) if  $\chi_R$  is a recursive function.

**Theorem 7.6.** A function  $f : \omega^k \rightarrow \omega$  is recursive iff the function  $f$  is Turing computable.

Summary of proof that every recursive function is Turing computable: Firstly show that the basic recursive functions in R1 are Turing computable by constructing Turing machines computing each of those. Secondly, suppose that  $f$  is obtained by composition of  $H_1, \dots, H_k$  and  $G$ , with corresponding Turing machines  $M_{H_i}$  and  $M_G$ . Then show that the machines can be combined to produce a machine  $M_f$  which computes  $f$ . Thirdly, show similarly that a machine  $M_G$  can be altered into  $M_f$  which computes  $f$  obtained by minimization of  $G$ .

Summary of proof that every Turing computable function is recursive: Suppose that a Turing machine  $M$  computes a function  $f$  on  $\omega^k$ . We only concern the case  $k \geq 1$ . Moreover, without loss of generality we can assume that  $k = 1$ , since there is a bijective recursive function from  $\omega^k$  to  $\omega$ .

Now at  $s$  step after the machine began to run, we can describe the current configuration of the input tape representing  $x \in \omega$  as a tuple  $(\bar{a}, b, \bar{c})$ , where  $\bar{a}$  is the sequence of symbols from the starting square to the square just before the scanned square,  $b$  is the symbol in the scanned square, and  $\bar{c}$  is the sequence of symbols to the right of the scanned square (up to the last 1 before the blank). By the encoding techniques, which will be described in the proof of Gödel's incompleteness theorems, we can obtain a single number coding the configuration  $(\bar{a}, b, \bar{c})$ . Now there is a recursive function  $H_M$  on  $\omega^2$  such that  $H_M(x, s)$  is the code for the configuration of, at  $s$  step, the input tape representing  $x$ . Clearly  $H_M(x, s) = H_M(x, s_0)$  for all  $s > s_0$ , where  $s_0$  is the number at which step the machine  $M$  halts with the input  $x$ . Now if we define  $G(x) :=$  the code of the sequence  $(H_M(x, 0), \dots, H_M(x, s_0))$ , then the function  $G$  is again recursive. Moreover  $x < G(x)$  and  $f(x) < G(x)$ . Define another recursive function  $F_M$  such that

$$F_M(c) = \begin{cases} f(x) & \text{if there is } x < c \text{ such that } G(x) = c, \\ 0 & \text{otherwise.} \end{cases}$$

Then  $f(x) = F_M(G(x))$ . Hence  $f$  is recursive.

- Church's Thesis

**Definition\* 1** A countable set  $S(\subseteq \text{countable } Ob)$  is called *computable\** if there is an algorithm (finitely described mechanical decision procedure (e.g. a computer program)) determining the membership of  $S$ , i.e. for given input  $a \in Ob$ , if  $a \in S$  then after finitely many steps the algorithm produces an output 'yes'; if  $a \notin S$  then the output is 'no'.

A function  $f : \omega^k \rightarrow \omega$  is *computable\** if there is an algorithm which effectively produces  $f(\bar{a})$  for given  $\bar{a} \in \omega^k$ .

**Church's Thesis** The intuitive concepts of computable\* (total) functions, and sets (in  $\omega^k$ ), coincide with Turing computable (equivalently recursive) functions, and sets.

**Proposition\* 2** In reasonable\*(=computable\*) countable language  $\mathcal{L}$ , the set of all logical axioms (in  $\text{Exp}(\mathcal{L})$  or  $\text{Sent}(\mathcal{L})$ ) are computable\*.

**Definition\* 3**

- (1) A theory  $T$  in a countable language  $\mathcal{L}$  is called *axiomatizable\** if there is a computable\*  $S$  that axiomatizes  $T$ .
- (2) The theory  $T$  is said to be *decidable\** if  $Cn(T)$  is computable\*.

Now we introduce the notion weaker than computability\*.

**Definition\* 4** A countable set  $S$  is called *effectively enumerable\** if there is an algorithm which effectively lists all the members of  $S$  in some order.

**Proposition\* 5** A countable set  $S$  of objects is given. The following are equivalent.

- (1)  $S$  is effectively enumerable\*.

- (2) There is an algorithm such that given input pair  $(a, n)$  of an object  $a$  and  $n \in \omega$  the algorithm always halts with ‘yes’ or ‘no’. Moreover  $a \in S$  iff given input  $(a, n)$  for some  $n \in \omega$ , the algorithm halts with ‘yes’.

In the spirit of above Proposition\* 5, we can define the formal counterpart of effective enumerability\*.

**Definition 7.7.**  $P(\subseteq \omega^n)$  is called recursively enumerable (equivalently called computably enumerable) if there is a recursive relation  $R \subseteq \omega^{n+1}$  such that  $P(\bar{a})$  iff  $\exists x R(\bar{a}x)$ .

**Church’s Thesis for effective enumerability** The intuitive concept of effective enumerability\* coincides with that of recursive enumerability.

**Proposition\* 6** For a computable\* set of formulas  $\Gamma$  in a reasonable\*  $\mathcal{L}$ , the set of formulas  $\{\varphi \mid \Gamma \models \varphi\}$  is effectively enumerable\*. In particular, the set of all valid formulas is effectively enumerable\*.

**Corollary\* 7** If a theory  $T$  in a reasonable\*  $\mathcal{L}$  is axiomatizable\*, then  $Cn(T)$  is effectively enumerable\*.

**Corollary\* 8** If a theory  $T$  in a reasonable\*  $\mathcal{L}$  is axiomatizable\* and complete, then  $T$  is decidable\*.

We shall see the formal counterparts of above starred propositions.