Mathematical Logic (I)

Yijia Chen

In mathematics, we prove theorems by proofs. In mathematical logic, we study those proofs as mathematical objects in their own right. The following are some of the key questions we want to address in this course.

(Q1) What is a mathematical proof?

(Q2) What makes a proof correct?

(Q3) Is there a boundary of provability?

(Q4) Can computers find proofs?

Quick answers: 能证明的都对，对的都能证明

1. Proofs are built upon **first-order logic**.
   formal: 形式化的，有陈述序列
2. There are **formal proof systems** in which every true mathematical statement has a proof, and conversely every provable mathematical statement is true. This is known as **Gödel Completeness Theorem**.
   Reasonable: 你能用计算机来做
3. For any reasonable proof system, there are true mathematical statement about natural numbers $\mathbb{N}$ that have no proof in that system. This is **Gödel's First Incompleteness Theorem**.
4. Any computer program cannot decide whether an arbitrary input mathematical statement has a proof. This is **Turing's undecidability of the halting problem**.

A proof sketch of (4)

Let us fix a programming language, e.g., C++. For any C++ program $\mathbb{P}$ and its input x we write down a mathematical statement:

$$\varphi_{\mathbb{P},x} := \text{"}\mathbb{P} \text{ will eventually halt on input } x.\text{"}$$

We assume without proof that

$$\varphi_{\mathbb{P},x} \text{ has a proof} \quad \Longleftrightarrow \quad \mathbb{P} \text{ will eventually halt on input } x. \tag{1}$$

Now assume that there is a C++ program $\mathbb{T}$ such that for any given mathematical statement φ

(T1) $\mathbb{T}(\varphi)$ outputs "yes", if φ has a proof;

(T2) $\mathbb{T}(\varphi)$ outputs "no", if φ has no proof.

Now consider the following program (in pseudo-code):

这个 statement 是什么？ x halts on input x

$\mathbb{H}(x)$ // x (the code of) a C++ program
1. construct the mathematical statement $\varphi_{x,x}$
2. call the program $\mathbb{T}$ on input $\varphi_{x,x}$
3. **if** $\mathbb{T}(\varphi_{x,x}) = \text{yes}$ **then** run forever
4. **else** halt.

We analyse the behaviour of the program $\mathbb{H}$ on input (the code of) itself. Assume that $\mathbb{H}(\mathbb{H})$ halts.

$$\begin{aligned}
\mathbb{H}(\mathbb{H}) \text{ halts} &\Longrightarrow \varphi_{\mathbb{H},\mathbb{H}} \text{ has a proof,} && \big(\text{by (1)}\big)\\
&\Longrightarrow \mathbb{T}(\varphi_{\mathbb{H},\mathbb{H}}) \text{ outputs “yes”,} && \big(\text{by (T1)}\big)\\
&\Longrightarrow \mathbb{H} \text{ does not halt on input } \mathbb{H} && \big(\text{by line 3}\big).
\end{aligned}$$

Otherwise:

$$\begin{aligned}
\mathbb{H}(\mathbb{H}) \text{ does not halt} &\Longrightarrow \varphi_{\mathbb{H},\mathbb{H}} \text{ has no proof,} && \big(\text{by (1)}\big)\\
&\Longrightarrow \mathbb{T}(\varphi_{\mathbb{H},\mathbb{H}}) \text{ outputs “no,”} && \big(\text{by (T2)}\big)\\
&\Longrightarrow \mathbb{H} \text{ halts on input } \mathbb{H} && \big(\text{by line 4}\big).
\end{aligned}$$

$\square$

1 The Syntax of First-order Logic

Example 1.1 (Group Theory).

(G1) For all x, y, z we have $(x \circ y) \circ z = x \circ (y \circ z)$.

(G2) For all x we have $x \circ e = x$. 零元

(G3) For every x there is a y such that $x \circ y = e$. 逆元

A group is a triple $\mathfrak{G} = (G, \circ^{\mathfrak{G}}, e^{\mathfrak{G}})$, i.e., a structure $\mathfrak{G}$, which satisfies (G1)–(G3). $\dashv$

Example 1.2 (Equivalence Relations).

设 A 是一个集合，A 上的一个“关系” R 是 $A \times A$ 的一个子集（笛卡尔积）。
如果 $(a,b) \in R$，我们就说 a 与 b 有关系 R，记作 aRb

(E1) For all x we have $(x, x) \in R$. 自反性

(E2) For all x and y if $(x, y) \in R$ then $(y, x) \in R$. 对称性

(E3) For all x, y, z if $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$. 传递性

等价关系，例子：$R = \{(x,y) | x - y \text{ is even}\}$
(1) $\forall x, (x,x) \in R$
(2) $\forall x, y$, if $(x,y) \in R$, then $(y,x) \in R$
(3) $\forall x, y, z$, if $(x,y) \in R, (y,z) \in R$, then $(x,z) \in R$

An equivalence relation is specified by a structure $\mathfrak{A} = (A, R^{\mathfrak{A}})$ in which R^{A} satisfies (E1)–(E3). $\dashv$

1.1 Alphabets

Definition 1.3. An **alphabet** is a nonempty set of **symbols**. $\dashv$

Examples 1.4.

$$\begin{aligned}
\mathbb{A}_1 &:= \{0, 1, \dots, 9\}, && \text{i.e., the alphabet for numbers,}\\
\mathbb{A}_2 &:= \{a, b, \dots, z\}, && \text{i.e., the Latin alphabet,}\\
\mathbb{A}_3 &:= \{+, \times\},\\
\mathbb{A}_4 &:= \{c_0, c_1, \dots\}.
\end{aligned}$$

$\dashv$

Definition 1.5. Let $\mathbb{A}$ be an alphabet. Then a **word** w over $\mathbb{A}$ is a finite sequence of symbols in $\mathbb{A}$, i.e.,

$$w = w_1 w_2 \cdots w_n$$

where $n \in \mathbb{N}$ and $w_i \in \mathbb{A}$ for every $i \in [n] = \{1, \ldots, n\}$. In case $n = 0$, then w is the **empty word**, denoted by ε. The **length** $|w|$ of w is n. In particular, $|\varepsilon| = 0$.
$\mathbb{A}^*$ denotes the set of all words over $\mathbb{A}$, or equivalently

$$\mathbb{A}^* = \bigcup_{n \in \mathbb{N}} \mathbb{A}^n = \bigcup_{n \in \mathbb{N}} \left\{ w_1 \ldots w_n \;\middle|\; w_1, \ldots, w_n \in \mathbb{A} \right\}.$$

⊣

Countable sets

Later on, we will need to count the number of words over a given alphabet.

Definition 1.6. A set M is **countable** if there exists an **injective** function α from $\mathbb{N}$ **onto** M, i.e., $\alpha : \mathbb{N} \to M$ is a bijection. Thereby, we can write

$$M = \left\{ \alpha(n) \;\middle|\; n \in \mathbb{N} \right\} = \left\{ \alpha(0), \alpha(1), \ldots, \alpha(n), \ldots \right\}.$$

A set M is **at most countable** if M is either finite or countable. 至多可数 ⊣

Lemma 1.7. *Let M be a non-empty set. Then the following are equivalent.*

(a) *M is at most countable.*

(b) *There is a surjective function $f : \mathbb{N} \to M$.*

(c) *There is an injective function $f : M \to \mathbb{N}$.* ⊣

Lemma 1.8. *Let $\mathbb{A}$ be an alphabet which is at most countable. Then $\mathbb{A}^*$ is countable.* ⊣

1.2 The alphabet of a first-order language

Definition 1.9. The **alphabet of a first-order language** consists of the following symbols.

(a) $v_0, v_1, \ldots$ (variables).

(b) $\neg, \wedge, \vee, \to, \leftrightarrow$, (negation, conjunction, disjunction, implication, if and only if).

(c) $\forall, \exists$, (for all, exists).

(d) $\equiv$, (equality). 注意：这代表语法相等（你需要一个在“语法”中对应于“语义等号”的东西）

(e) $($, $)$, (parentheses).

(f) (1) For every $n \geqslant 1$ a set of n**-ary relation symbols**. 约定：P, Q, R
 (2) For every $n \geqslant 1$ a set of n**-ary function symbols**. 约定：f, g, h
 (3) A set of **constants**. 约定：$c_0, c_1, \cdots$

Note any set in (f) can be empty. ⊣

We use $\mathbb{A}$ to denote the set of symbols in (a)–(e), i.e., the set of **logic symbols**, while S is the set of remaining symbols in (f). Then a first-order language has

$$\mathbb{A}_S := \mathbb{A} \cup S$$

as its alphabet and S as its **symbol set**.

Thus every first-order language has the same set $\mathbb{A}$ of logic symbols but might have different symbol set S.

Examples 1.10. 1. For group theory we take $S_{Gr} := \{\circ, e\}$ where $\circ$ is a binary function symbol and e is a constant.

2. For equivalence relations let $S_{Eq} := \{R\}$ where R is a binary relation symbol. $\dashv$

In discussions, we often use $P, Q, R, \ldots$ to refer to relations symbols, $f, g, h, \ldots$ to function symbols, $c_0, c_1, \ldots$ to constants, and $x, y, z, \ldots$ to variables.

1.3 Terms and formulas

Throughout this section, we fix a symbol set S.

Definition 1.11. The set T^S of S-**terms** contains precisely those words in $\mathbb{A}_S^*$ which can be obtained by applying the following rules finitely many times.

(T1) Every variable is an S-term. **Term** 表达式求出来的就是 A_S 中的“值”，因为是“function”!

(T2) Every constant in S is an S-term.

（例如：$+v_7v_8$）当然为了方便也可以记为 $v_7 + v_8$

(T3) If $t_1, \ldots, t_n$ are S-terms and f is a n-ary function symbol in S, then $ft_1 \ldots t_n$ is an S-term. $\dashv$

Definition 1.12. The set L^S of S-**formulas** contains precisely those words in $\mathbb{A}_S^*$ which can be obtained by applying the following rules finitely many times.

Formula 公式求出来的就会有“真和假”，因为是“relation”!

(A1) Let t_1 and t_2 be two S-terms. Then $t_1 \equiv t_2$ is an S-formula.

(A2) Let $t_1, \ldots, t_n$ be S-terms and R an n-ary relation symbol in S. Then $Rt_1 \cdots t_n$ is also an S-formula.

(A3) If φ is an S-formula, then so is $\neg\varphi$.

(A4) If φ and ψ are S-formulas, then so is $(\varphi * \psi)$ where $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

(A5) Let φ be an S-formula and x a variable. Then $\forall x\varphi$ and $\exists x\varphi$ are S-formulas, too.

The formulas in (A1) and (A2) are **atomic**, as they don't contain any other S-formulas as subformulas.

- $\neg\varphi$ is the **negation** of φ.
- $(\varphi \wedge \psi)$ is the **conjunction** of φ and ψ.
- $(\varphi \vee \psi)$ is the **disjunction** of φ and ψ.
- $(\varphi \rightarrow \psi)$ is the **implication** from φ to ψ.
- $(\varphi \leftrightarrow \psi)$ is the **equivalence** between φ and ψ. $\dashv$

example:

For all $x, (x, x) \in R$ $\qquad \forall x Rxx$

For all x, if $(x, y) \in R$, then $(y, x) \in R$ $\qquad \forall x \forall y (Rxy \rightarrow Ryx)$

Lemma 1.13. *Let S be at most countable. Then both T^S and L^S are countable.*

Definition 1.14. Let t be an S-term. Then $\mathrm{var}(t)$ is the set of variables in t. Or inductively,

$$\begin{aligned} \mathrm{var}(x) &:= \{x\}, \\ \mathrm{var}(c) &:= \emptyset, \\ \mathrm{var}(ft_1 \ldots t_n) &:= \bigcup_{i \in [n]} \mathrm{var}(t_i). \end{aligned} \qquad \dashv$$

Definition 1.15. Let φ be an S-formula and x a variable. We say that **an occurrence of x in φ is free** if it is not in the scope of any $\forall x$ or $\exists x$. Otherwise, the occurrence is **bound**.

free(φ) is the set of variables which have free occurrences in φ. Or inductively,

How to define "scope"? e.g. $\forall x\varphi$ and x is not in φ

e.g. $(x < y \to \forall y\ y + 0 \equiv y)$ then first y is free, second is not

$$\begin{aligned}
\mathrm{free}(t_1 \equiv t_2) &:= \mathrm{var}(t_1) \cup \mathrm{var}(t_2), \\
\mathrm{free}(Rt_1 \cdots t_n) &:= \bigcup_{i\in[n]} \mathrm{var}(t_i), \\
\mathrm{free}(\neg\varphi) &:= \mathrm{free}(\varphi), \\
\mathrm{free}(\varphi * \psi) &:= \mathrm{free}(\varphi) \cup \mathrm{free}(\psi) \quad \text{with } * \in \{\wedge, \vee, \to, \leftrightarrow\}, \\
\mathrm{free}(\forall x\varphi) &:= \mathrm{free}(\varphi) \setminus \{x\}, \\
\mathrm{free}(\exists x\varphi) &:= \mathrm{free}(\varphi) \setminus \{x\}.
\end{aligned}$$ ⊣

Example 1.16. The formula below shows that a variable might have both free and bound occurrences in the same formula.

$$\begin{aligned}
\mathrm{free}((Rxy \to \forall y \neg y \equiv z)) &= \mathrm{free}(Rxy) \cup \mathrm{free}(\forall y \neg y \equiv z) \\
&= \{x, y\} \cup \big(\mathrm{free}(y \equiv z) \setminus \{y\}\big) = \{x, y, z\}.
\end{aligned}$$ ⊣

Definition 1.17. An S-formula is an S-**sentence** if free(φ) $= \emptyset$. 语句（就是数学性质！） ⊣

E.g. $\forall x \forall y(Rxy \to Ryx)$ 这个就没有 free variable，所以是语句

Recall that the **actual** variables we can use are $v_0, v_1, \ldots$.

Definition 1.18. Let $n \in \mathbb{N}$. Then

$$L_n^S := \big\{\varphi \mid \varphi \text{ an S-formula with } \mathrm{free}(\varphi) \subseteq \{v_0, \ldots, v_{n-1}\}\big\}.$$

In particular, L_0^S is the set of S-sentences. ⊣