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:

1. Proofs are built upon **first-order logic**.
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**.
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

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

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):

$\mathbb{H}(x)$ // 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** $\mathbb{T}(\varphi_{x,x}) =$ 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:

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

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 $x \circ e = x$.

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

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

Example 1.2 (Equivalence Relations)**.**

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

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

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

1.1 Alphabets

Definition 1.3. An **alphabet** is a nonempty set of **symbols**. ⊣

Examples 1.4.

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

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 $i \in [n] = \{1, \dots, n\}$. 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

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

$\dashv$

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

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

A set M is **at most countable** if M is either finite or countable. $\dashv$

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 $f : M \to \mathbb{N}$.* $\dashv$

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

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) $v_0, v_1, \dots$ (variables).

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

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

(d) $\equiv$, (equality).

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

(f) (1) For every $n \geqslant 1$ a set of n**-ary relation symbols**.
  (2) For every $n \geqslant 1$ a set of n**-ary function symbols**.
  (3) A set of **constants**.

Note any set in (f) can be empty. $\dashv$

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 $S_{\mathrm{Gr}} := \{\circ, e\}$ where $\circ$ is a binary function symbol and e is a constant.

2. For equivalence relations let $S_{\mathrm{Eq}} := \{R\}$ 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.

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

(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.

(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. $\dashv$

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,

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

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**.

$\text{free}(\varphi)$ is the set of variables which have free occurrences in φ. Or inductively,

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

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

$$\begin{aligned}
\text{free}((Rxy \rightarrow \forall y \neg y \equiv z)) &= \text{free}(Rxy) \cup \text{free}(\forall y \neg y \equiv z) \\
&= \{x, y\} \cup \big(\text{free}(y \equiv z) \setminus \{y\}\big) = \{x, y, z\}.
\end{aligned} \qquad \dashv$$

Definition 1.17. An S-formula is an S**-sentence** if $\text{free}(\varphi) = \emptyset$. $\dashv$

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

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

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

In particular, L_0^S is the set of S-sentences. $\dashv$