Mathematical Logic (IX)

Yijia Chen

1. Decidability and Enumerability

A. Procedure and Decidability.

Definition 1.1. Let $\mathcal{A}$ be an alphabet (which we always assume to be finite) and $W \subseteq \mathcal{A}^*$.

(i) Let $\mathbb{P}$ be a procedure/program (which we will make precise shortly afterwards). $\mathbb{P}$ is a *decision procedure for* W if on every input $w \in \mathcal{A}^*$ the procedure $\mathbb{P}$ will eventually halt and output some $w' \in \mathcal{A}^*$ such that

- if $w \in W$, then $w' = \square$, where $\square$ is the empty string,
- if $w \notin W$, then $w' \neq \square$.

(ii) W is *decidable* if there is a decision procedure for W. ⊣

B. Enumerability.

Definition 1.2. Let $\mathcal{A}$ be an alphabet and $W \subseteq \mathcal{A}^*$.

(i) A procedure $\mathbb{P}$ is an *enumeration procedure for* W if $\mathbb{P}$ (without any input) outputs all the words in W (in some order and possibly with repetitions).

(ii) W is *enumerable* if there is an enumeration procedure for W. ⊣

Lemma 1.3. *If there is an enumeration procedure for W, then there is an enumeration procedure for W* without repetitions. ⊣

Lemma 1.4. *Let $\mathcal{A}$ be finite. Then $\mathcal{A}^*$ is enumerable.* ⊣

Let

$$\begin{aligned} S_\infty := \{c_0, c_1, \dots\} & \qquad \text{(every } c_i \text{ is a constant)} \\ \cup \bigcup_{n \geqslant 1} \{R_0^n, R_1^n, \dots\} & \qquad \text{(every } R_i^n \text{ is an } n\text{-ary relation symbol)} \\ \cup \bigcup_{n \geqslant 1} \{f_0^n, f_1^n, \dots\} & \qquad \text{(every } f_i^n \text{ is an } n\text{-ary function symbol).} \end{aligned}$$

Lemma 1.5.

$$\left\{ \varphi \in L_0^{S_\infty} \;\middle|\; \models \varphi \right\}$$

is enumerable.

Proof: [sketch] By the Completeness Theorem

$$\left\{ \varphi \in L_0^{S_\infty} \;\middle|\; \models \varphi \right\} = \left\{ \varphi \in L_0^{S_\infty} \;\middle|\; \vdash \varphi \right\}.$$

Thus, we can enumerate all possible proofs/derivations of symbol set S^∞, thus obtain all those $\varphi \in L_0^{S_\infty}$ with $\vdash \varphi$. □

C. The Relationship between Decidability and Enumerability.

Theorem 1.6. *Every decidable set is enumerable.*

Proof: Assume that the procedure $\mathbb{P}$ decides $W \subseteq \mathcal{A}^*$. By Lemma 1.4 we can enumerate all $w \in \mathcal{A}^*$. For each w we can decide whether $w \in W$ by calling $\mathbb{P}$. If so, we output w and proceed to the next string. Otherwise, we move to the next string without outputting w. □

We will see later that the converse of Theorem 1.6 does not hold, i.e., there are enumerable sets which are not decidable. Nevertheless, we can show:

Theorem 1.7. *Let $W \subseteq \mathcal{A}^*$. Then W is decidable if and only if both W and $\mathcal{A}^* \setminus W$ are enumerable.*

Proof: The direction from left to right is straightforward by Theorem 1.6 and by observing that $\mathcal{A}^* \setminus W$ is decidable as well. For the converse, we have two procedures, $\mathbb{P}_1$ which enumerates W, and $\mathbb{P}_2$ which enumerates $\mathcal{A}^* \setminus W$.

Then given an input $w \in \mathcal{A}^*$, we simulate two procedures $\mathbb{P}_1$ and $\mathbb{P}_2$ simultaneously[1], eventually w will appear in exactly one of the outputs of $\mathbb{P}_1$ and $\mathbb{P}_2$. Then we can answer whether $w \in W$ accordingly. □

D. Computable Functions.

Definition 1.8. Let $\mathcal{A}$ and $\mathcal{B}$ be two alphabets. A procedure that for each input $w \in \mathcal{A}^*$ outputs a $w' \in \mathcal{B}^*$ determines a function $f : \mathcal{A}^* \to \mathcal{B}^*$ defined by

$$w \overset{f}{\mapsto} w'.$$

f is said to be *computable*. ⊣

1.1. Register Machines. We fix an alphabet

$$\mathcal{A} := \{a_0, \ldots, a_r\}.$$

Every *register machine* (or simply, machine) has a fixed number of registers, i.e.,

$$R_0, \ldots, R_m$$

for some fixed $m \in \mathbb{N}$, where any register R_i can contain any word in $\mathcal{A}^*$. A *program* consists of a finite number of *instructions*, each starting with a *label* $L \in \mathbb{N}$.

There are 5 types of instructions.

–

$$L \textbf{ LET } R_i = R_i + a_j,$$

where $L, i, j \in \mathbb{N}$ with $0 \leqslant i \leqslant m$ and $0 \leqslant j \leqslant r$. That is, add the letter a_j at the end of the word in R_i.

–

$$L \textbf{ LET } R_i = R_i - a_j,$$

where $L, i, j \in \mathbb{N}$ with $0 \leqslant i \leqslant m$ and $0 \leqslant j \leqslant r$. That is, if the word in R_i ends with e_j, then delete this a_j; otherwise leave the word unchanged.

–

$$L \textbf{ IF } R_i = \square \textbf{ THEN } L' \textbf{ ELSE } L_0 \textbf{ OR } L_1 \textbf{OR } \cdots \textbf{OR } L_r,$$

where $L, L', L_0, \ldots, L_r \in \mathbb{N}$. That is, if R_i contains $\square$, then go the instruction labelled L'. Otherwise, if R_i contains a word ending with the letter a_j, then go to the instruction labelled L_j.

[1] More precisely, we simulate the steps of $\mathbb{P}_1$ and $\mathbb{P}_2$ alternatively, i.e., the first step of $\mathbb{P}_1$, the first step of $\mathbb{P}_2$, the second step of $\mathbb{P}_1$, the second step of $\mathbb{P}_2$, ...

–

$$\mathtt{L}\ \textbf{PRINT},$$

where $\mathtt{L} \in \mathbb{N}$. That is, output the word in $\mathtt{R}_0$.

–

$$\mathtt{L}\ \textbf{HALT},$$

with $\mathtt{L} \in \mathbb{N}$. That is, the program halts.

Definition 1.9. A *register program* (or simply *program*) is a finite sequence α_0, α_k of instructions with the following properties.

(i) Every α_i has label $\mathtt{L} = i$.

(ii) Every jump operation refers to a label $\leqslant k$.

(iii) Only the last instruction α_k is a halt instruction. ⊣

Definition 1.10. A program $\mathbb{P}$ *starts* with $w \in \mathcal{A}^*$ if in the beginning of the execution of $\mathbb{P}$ we have $\mathtt{R}_0 = w$ and all other $\mathtt{R}_i = \square$.

If $\mathbb{P}$ starts with w and eventually reaches the last halt instruction, then we write

$$\mathbb{P} : w \to \text{halt}.$$

Otherwise,

$$\mathbb{P} : w \to \infty.$$

The notation

$$\mathbb{P} : w \to w'$$

means that if $\mathbb{P}$ starts with w, then it eventually halts with $\mathtt{R}_0 = w'$. ⊣

Definition 1.11. Let $W \subseteq \mathcal{A}^*$.

(i) A program $\mathbb{P}$ *decides* W if for all $w \in \mathcal{A}^*$

$$\begin{aligned} &\mathbb{P} : w \to \square && \text{if } w \in W, \\ &\mathbb{P} : w \to w' \text{ with } w' \neq \square && \text{if } w \notin W. \end{aligned}$$

(ii) W is *register-decidable*, or *R-decidable* for short, if there is program which decides W. ⊣

Definition 1.12. Let $W \subseteq \mathcal{A}^*$.

(i) A program $\mathbb{P}$ *enumerates* W if started with $\square$, $\mathbb{P}$ prints out exactly the words in W (in some order with possible repetitions).

(ii) W is *register-enumerable*, or *R-enumerable* for short, if there is program which enumerates W. ⊣

Proposition 1.13. *Let $W \subseteq \mathcal{A}^*$. Then W is R-decidable if and only if both W and $\mathcal{A}^* \setminus W$ are R-enumerable.*

Definition 1.14. Let $\mathrm{F} \subseteq \mathcal{A}^* \to \mathcal{B}^*$, where $\mathcal{A}$ and $\mathcal{B}$ are two alphabets.

(i) A program $\mathbb{P}$ *computes* F if for all $w \in \mathcal{A}^*$

$$\mathbb{P} : w \to \mathsf{F}(w).$$

(ii) F is *register-computable*, or *R-computable* for short, if there is program which computes F. ⊣

1.2. The halting problem for the register machines. Again let $\mathcal{A} := \{a_0, \ldots, a_r\}$ be a fixed alphabet. Our goal is to define for every program $\mathbb{P}$ over $\mathcal{A}$ a word $w_{\mathbb{P}} \in \mathcal{A}^*$. To that end, we first introduce an auxiliary alphabet

$$\mathcal{B} := \mathcal{A} \cup \{A, B, C, \ldots, Z\} \cup \{0, 1, \ldots, 9\} \cup \{=, +, -, \square, |\}.$$

As usual, we understand that the words in $\mathcal{B}^*$ are ordered *lexicographically*. Then every program can be naturally encoded as a word in $\mathcal{B}^*$. For instance

0 **LET** $R_1 = R_1 - a_0$

2 **PRINT**

3 **HALT**

is identified with the word

$$0\mathsf{LETR1} = \mathsf{R1} - a_0 \mid 1\mathsf{PRINT} \mid 3\mathsf{HALT}.$$

Note that a_0 is single letter from the alphabet $\mathcal{A} \subseteq \mathcal{B}$. Assume that this word is the n-th word in the lexicographical ordering of $\mathcal{B}^*$. Then we set

$$w_{\mathbb{P}} := \underbrace{a_0 a_0 \cdots a_0}_{n \text{ times}}.$$

Finally let

$$\Pi := \{w_{\mathbb{P}} \mid \mathbb{P} \text{ a program over } \mathcal{A}\}.$$

The mapping

$$\mathbb{P} \mapsto w_{\mathbb{P}}$$

is often called the *Gödel numbering*, and $w_{\mathbb{P}}$ is the *Gödel number* of $\mathbb{P}$.

Lemma 1.15. *Π is R-decidable.* ⊣

Theorem 1.16. *(i) The set*

$$\Pi'_{\text{halt}} := \{w_{\mathbb{P}} \mid \mathbb{P} \textit{ a program over } \mathcal{A} \textit{ and } \mathbb{P} : w_{\mathbb{P}} \to \text{halt}\}$$

is not R-decidable.

(ii) The set

$$\Pi_{\text{halt}} := \{w_{\mathbb{P}} \mid \mathbb{P} \textit{ a program over } \mathcal{A} \textit{ and } \mathbb{P} : \square \to \text{halt}\}$$

is not R-decidable. ⊣

2. Exercises

Exercise 2.1. Let $W \subseteq \mathcal{A}^*$. A program $\mathbb{P}$ *strictly enumerates* W if started with $\square$, $\mathbb{P}$ prints out all the words in W

$$w_0, w_1, \ldots$$

without repetitions such that $|w_i| \leqslant |w_{i+1}|$ for all $i \in \mathbb{N}$. Recall $|w|$ denotes the length of the word w.

W is strictly R-enumerable if there is a program which strictly enumerates W. Are the following statements correct?

– W is R-enumerable if and only W is strictly R-enumerable.

– W is R-decidable if and only W is strictly R-enumerable. ⊣

Exercise 2.2. Prove that the set

$$\left\{ w_{\mathbb{P}} \;\middle|\; \mathbb{P} \text{ a program over } \mathcal{A} \text{ and } \mathbb{P} : w \to \text{halt } \textit{for some } w \in \mathcal{A}^* \right\}$$

is not R-decidable. ⊣