Mathematical Logic (XI)

Yijia Chen

1. Decidability and Enumerability

1.1. Register Machines.

We fix an alphabet

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

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

$$R_0, \dots, R_m$$

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

–
$$L \ \textbf{PRINT},$$
where $L \in \mathbb{N}$. That is, output the word in R_0.

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

Definition 1.1. A *register program* (or simply *program*) is a finite sequence $\alpha_0, \dots, \alpha_k$ of instructions with the following properties.

(i) Every α_i has label $L = i$.

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

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

Definition 1.2. A program $\mathbb{P}$ *starts* with $w \in \mathcal{A}^*$ if in the beginning of the execution of $\mathbb{P}$ we have $R_0 = w$ and all other $R_i = \Box$.

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, and during the course of computation, has printed exactly one string w'. ⊣

Definition 1.3. Let $W \subseteq \mathcal{A}^*$.

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

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

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

Definition 1.4. Let $W \subseteq \mathcal{A}^*$.

(i) A program $\mathbb{P}$ *enumerates* W if started with $\Box$, $\mathbb{P}$ 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 1.5. *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 1.6. Let $F : \mathcal{A}^* \to \mathcal{B}^*$, where $\mathcal{A}$ and $\mathcal{B}$ are two alphabets.

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

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

(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 $\mathcal{A} := \{a_0, \dots, a_r\}$ 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

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

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** $R_1 = R_1 - a_0$

1 **PRINT**

2 **HALT**

is identified with the word

$$0\mathsf{LETR}1 = \mathsf{R}1 - \mathsf{a}_0 \mid 1\mathsf{PRINT} \mid 2\mathsf{HALT}.$$

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 1.7. *Π is* R*-decidable.* ⊣

Theorem 1.8. *Let $\mathcal{A}$ be a fixed alphabet.*

(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

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

is not R-decidable. ⊣

Proof: (i) Assume that there is a program $\mathbb{P}_0$ which decides Π'_{halt}. That is, for every program $\mathbb{P}$

$$\begin{aligned} &\mathbb{P}_0 : w_{\mathbb{P}} \to \Box && \text{if } \mathbb{P} : w_{\mathbb{P}} \to \text{halt}, \\ &\mathbb{P}_0 : w_{\mathbb{P}} \to w' \text{ with } w' \neq \Box && \text{if } \mathbb{P} : w_{\mathbb{P}} \to \infty. \end{aligned}$$

Assume furthermore that $\mathbb{P}_0$ has the form

0 ......

1 ......

⋮

10 **PRINT**

⋮

k **HALT**

We change $\mathbb{P}_0$ in such a way that if $\mathbb{P}_0$ prints out $\Box$, then the modified program will never halt. To that end, we replace the last k-th halt instruction by two instructions that "reverse the halting behavior", and replace every print instruction by a "jump" instruction that directly goes to the end:

$0 \;\ldots\ldots$

$1 \;\ldots\ldots$

$\vdots$

10 **IF** $\mathsf{R}_0 = \Box$ **THEN** k **ELSE** k **OR** k **OR** $\cdots$ **OR** k

i.e, goto the k-th instruction no matter what is in R_0

$\vdots$

k **IF** $\mathsf{R}_0 = \Box$ **THEN** k **ELSE** $\mathsf{k}+1$ **OR** $\mathsf{k}+1$ **OR** $\cdots$ **OR** $\mathsf{k}+1$

$\mathsf{k}+1$ **HALT**

Let $\mathbb{P}_1$ be the resulting program. It is then easy to see that for any program $\mathbb{P}$

$$\begin{aligned} &\mathbb{P}_1 : w_{\mathbb{P}} \to \infty && \text{if } \mathbb{P} : w_{\mathbb{P}} \to \text{halt},\\ &\mathbb{P}_1 : w_{\mathbb{P}} \to \text{halt} && \text{if } \mathbb{P} : w_{\mathbb{P}} \to \infty. \end{aligned}$$

As a result,

$$\begin{aligned} &\mathbb{P}_1 : w_{\mathbb{P}_1} \to \infty && \text{if } \mathbb{P}_1 : w_{\mathbb{P}_1} \to \text{halt},\\ &\mathbb{P}_1 : w_{\mathbb{P}_1} \to \text{halt} && \text{if } \mathbb{P}_1 : w_{\mathbb{P}_1} \to \infty, \end{aligned}$$

which is certainly a contradiction.

(ii) Towards a contradiction, assume that $\mathbb{P}_0$ decides Π_{halt}. That is, for every program $\mathbb{P}$

$$\begin{aligned} &\mathbb{P}_0 : w_{\mathbb{P}} \to \Box && \text{if } \mathbb{P} : \Box \to \text{halt},\\ &\mathbb{P}_0 : w_{\mathbb{P}} \to w' \text{ with } w' \neq \Box && \text{if } \mathbb{P} : \Box \to \infty. \end{aligned} \tag{1}$$

Now for every program $\mathbb{P}$ we assign in an *effective* way a program $\mathbb{P}^+$ such that

$$\mathbb{P} : w_{\mathbb{P}} \to \text{halt} \iff \mathbb{P}^+ : \Box \to \text{halt}. \tag{2}$$

Being effective means that there is a further program $\mathbb{T}$ that computes the mapping

$$w_{\mathbb{P}} \to w_{\mathbb{P}^+}.$$

The construction of $\mathbb{T}$ is tedious but not difficult.

With $\mathbb{P}_0$ and $\mathbb{T}$ we design a program which decides Π'_{halt}. On any $w \in \mathcal{A}^*$, the program first test whether $w = w_{\mathbb{P}}$ for some $\mathbb{P}$. If not, it rejects immediately[1]. Otherwise, it uses $\mathbb{T}$ to computes $w_{\mathbb{P}^+}$. Then the program calls $\mathbb{P}_0$ on input $w_{\mathbb{P}^+}$. By (2) and (1), it correctly decides whether

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

This gives us the desired contradiction to (i).

It remains to show the construction of $\mathbb{P}^+$ from any given $\mathbb{P}$ that fulfills (2). Assume that

$$w_{\mathbb{P}} = \underbrace{\mathsf{a}_0\mathsf{a}_0\ldots\mathsf{a}_0}_{n \text{ times}}.$$

Let $\mathbb{P}^+$ begin with

0 **LET** $\mathsf{R}_0 = \mathsf{R}_0 + \mathsf{a}_0$

[1] i.e., prints out some $w' \neq \Box$ and halts.

1 **LET** $R_0 = R_0 + a_0$

$\vdots$

n-1 **LET** $R_0 = R_0 + a_0$

and followed by the instructions of $\mathbb{P}$ with all labels increased by n. □

1.3. The undecidability of first-order logic.

Theorem 1.9. *The set*

$$\{\varphi \in L_0^{S_\infty} \mid \models \varphi\} \tag{3}$$

is not R-decidable.

Proof: By Theorem 1.8 (ii) for the alphabet $\mathcal{A} = \{|\}$ the problem Π_{halt} is not R-decidable. Our goal is to show that the assumed R-decidability of (3) would contradict this result. To that end, for every program $\mathbb{P}$ we will construct in an *effective* way a $\varphi_{\mathbb{P}} \in L_0^{S_\infty}$ such that

$$\mathbb{P} : \square \to \text{halt} \quad \iff \quad \models \varphi_{\mathbb{P}}.$$

Here, the effectivity means that there is a program $\mathbb{P}_1$ which computes the mapping

$$\mathbb{P} \mapsto \varphi_{\mathbb{P}}.$$

Once this is done, given an input $w \in \mathcal{A}^*$, we can first check whether $w = w_{\mathbb{P}}$, if so, extract the program $\mathbb{P}$ and compute $\varphi_{\mathbb{P}}$ using $\mathbb{P}_1$. Thus if (3) is decidable, we can apply the corresponding decision program on input $\varphi_{\mathbb{P}}$ to decide whether $\mathbb{P} : \square \to \text{halt}$. Hence, we could decide Π_{halt}.

Let $\mathbb{P}$ consist of instructions $\alpha_0, \dots, \alpha_k$, in particular every α_i has its label i. Furthermore, assume that the maximum index of the registers which $\mathbb{P}$ uses is n. Hence, the registers referred by all α_i's are among $R_0, \dots, R_n$.

Key to our construction of $\varphi_{\mathbb{P}}$ is the notion of configurations of $\mathbb{P}$. An $(n+2)$-tuple

$$(L, m_0, \dots, m_n)$$

is a *configuration of* $\mathbb{P}$ *(on input* □*) after* s *steps* if

- starting with input □ the program $\mathbb{P}$ runs at least s steps,
- after s steps, the instruction α_L is to be executed next,
- and for every $0 \leqslant i \leqslant n$ the register R_i contains the word

$$\underbrace{|\,|\cdots|}_{m_i \text{ times}}$$

  at that moment. To ease presentation, in the following we will simply say that R_i contains the number m_i.

Observe that then the execution of $\mathbb{P}$ on the $s+1$-th step is completely determined by the configuration $(L, m_0, \dots, m_n)$.

The *initial configuration*, i.e., the configuration of $\mathbb{P}$ after 0 step is

$$(0, 0, \dots, 0).$$

Recall that α_k is the last instruction of $\mathbb{P}$, i.e., the only halt instruction. Therefore

$$\begin{aligned} \mathbb{P} : \square \to \text{halt} \iff & \text{ for some } s, m_0, \dots, m_n \in \mathbb{N} \\ & \text{the tuple } (k, m_0, \dots, m_n) \text{ is the configuration of } \mathbb{P} \text{ after } s \text{ steps.} \end{aligned} \tag{4}$$

We choose four symbols from S^∞: $R := R_0^{n+2}$, $< := R_0^2$, $f := f_0^1$, and $c := c_0$, and set

$$S := \{R, <, f, c\}.$$

Then we associate with $\mathbb{P}$ an S-structure $\mathfrak{A}_\mathbb{P}$ which "describes" the execution (i.e., the behaviour) of $\mathbb{P}$ on input $\square$. We set $A_\mathbb{P} := \mathbb{N}$, $<^{\mathfrak{A}_\mathbb{P}} := \{(i,j) \mid i,j \in \mathbb{N} \text{ and } i < j\}$, $f^{\mathfrak{A}_\mathbb{P}}(i) := i+1$ for every $i \in \mathbb{N}$, $c^{\mathfrak{A}_\mathbb{P}} := 0$, and

$$R^{\mathfrak{A}_\mathbb{P}} := \{(L, m_0, \dots, m_n) \mid (L, m_0, \dots, m_n) \text{ is a reachable configuration of } \mathbb{P}\}.$$

Towards the definition of $\varphi_\mathbb{P}$ in (3), we first construct a sentence $\psi_\mathbb{P}$ which expresses the execution of $\mathbb{P}$ on $\square$. We abbreviate c, fc, ffc, … by $\bar{0}$, $\bar{1}$, $\bar{2}$, …, respectively. The desired $\psi_\mathbb{P}$ should satisfy the following two properties:

(P1) $\mathfrak{A}_\mathbb{P} \models \psi_\mathbb{P}$.

(P2) Let $\mathfrak{A}$ be an S-structure with $\mathfrak{A} \models \psi_\mathbb{P}$ and $(L, m_0, \dots, m_n)$ be a reachable configuration of $\mathbb{P}$. Then

$$\mathfrak{A} \models R\bar{L}\bar{m}_0 \cdots \bar{m}_n.$$

We set

$$\psi_\mathbb{P} := \psi_0 \wedge R\bar{0}\bar{0}\cdots\bar{0} \wedge \psi_{\alpha_0} \wedge \cdots \wedge \psi_{\alpha_{k-1}},$$

where each conjunct is defined as follows. The first

$$\psi_0 := \text{"} < \text{is an ordering"} \wedge \forall x(c < x \vee x \equiv c) \wedge \forall x(x < fx) \wedge \forall x \forall y\big(x < y \to (fx < y \vee fx \equiv y)\big),$$

i.e., $<$ is an ordering, c is the minimum element, fx is the successor of x.

For $\alpha \in \{\alpha_0, \dots, \alpha_{k-1}\}$ we define φ_α by a case analysis.

– $\alpha = L$ **LET** $R_i = R_i + \mid$. Then let

$$\psi_\alpha := \forall y_0 \cdots \forall y_n \big(R\bar{L}y_0 \cdots y_n \to R\overline{L+1}y_0 \cdots y_{i-1} f y_i y_{i+1} \cdots y_n\big).$$

– $\alpha = L$ **LET** $R_i = R_i - \mid$. Then let

$$\psi_\alpha := \forall y_0 \cdots \forall y_n \big(R\bar{L}y_0 \cdots y_n \to ((y_i \equiv \bar{0} \wedge R\overline{L+1}y_0 \cdots y_n) \vee (\neg y_i \equiv \bar{0} \wedge \exists u(fu \equiv y_i \wedge R\overline{L+1}y_0 \cdots y_{i-1} u y_{i+1} \cdots y_n)))\big).$$

– $\alpha = L$ **IF** $R_i = \square$ **THEN** L' **ELSE** L_0. Then let

$$\psi_\alpha := \forall y_0 \cdots \forall y_n \big(R\bar{L}y_0 \cdots y_n \to ((y_i \equiv \bar{0} \wedge R\overline{L'}y_0 \cdots y_n) \vee (\neg y_i \equiv \bar{0} \wedge R\overline{L_0}y_0 \cdots y_n))\big).$$

– $\alpha = L$ **PRINT**. Then let

$$\psi_\alpha := \forall y_0 \cdots \forall y_n \big(R\bar{L}y_0 \cdots y_n \to R\overline{L+1}y_0 \cdots y_n\big).$$

The verification of (P1) and (P2) is left as an exercise.

Finally let

$$\varphi_\mathbb{P} := \psi_\mathbb{P} \to \exists y_0 \cdots \exists y_n R\bar{k}y_0 \cdots y_n.$$

Now we verify that $\mathbb{P} : \Box \to \text{halt}$ if and only if $\models \varphi_{\mathbb{P}}$. First, assume $\models \varphi_{\mathbb{P}}$, in particular

$$\mathfrak{A}_{\mathbb{P}} \models \varphi_{\mathbb{P}}.$$

By (P1) we conclude

$$\mathfrak{A}_{\mathbb{P}} \models \exists y_0 \cdots \exists y_n R\bar{k} y_0 \cdots y_n.$$

Then there are some $s, m_0, \ldots, m_n \in A_{\mathbb{P}} \subseteq \mathbb{N}$ such that $(k, m_0, \ldots, m_n)$ is the configuration of $\mathbb{P}$ after s steps. Therefore, $\mathbb{P}$ reaches the last halt instruction after s steps, hence $\mathbb{P} : \Box \to \text{halt}$.

Conversely, assume $\mathbb{P} : \Box \to \text{halt}$. Let $\mathfrak{A}$ be an S-structure. We need to show that $\mathfrak{A} \models \varphi_{\mathbb{P}}$. Clearly, if $\mathfrak{A} \not\models \psi_{\mathbb{P}}$, then we are already done. Thus, assume $\mathfrak{A} \models \psi_{\mathbb{P}}$. Let $(k, m_0, \ldots, m_n)$ be the configuration of $\mathbb{P}$ when it reaches the last halt instruction α_k. Now (P2) implies that

$$\mathfrak{A} \models R\overline{s_{\mathbb{P}}}\bar{k}\bar{m}_0 \cdots \bar{m}_n.$$

Therefore

$$\mathfrak{A} \models \varphi_{\mathbb{P}}.$$

This finishes the proof. $\Box$