Logic and Computation I

Chapter 1 Introduction to theory of computation

Kazuyuki Tanaka

BIMSA

September 26, 2024

Logic and Computation I

- **Part 1. Introduction to Theory of Computation**
- **Part 2. Propositional Logic and Computational Complexity**
- **Part 3. First Order Logic and Decision Problems**
- **Part 4. Modal logic**

Part 1. Schedule

- Sep.10, (1) Automata and monoids
- Sep.12, (2) Turing machines
- Sep.19, (3) Computable functions and primitive recursive functions
- Sep.24, (4) Computability and incomputability
- Sep.26, (5) Partial recursive functions and computably enumerable sets
- Oct. 8, (6) Rice's theorem and many-one reducibility

Recap

- A TPL program $\mathcal{P}$ is a sequence of instructions with codes $c_0, c_1, \ldots, c_l$. So, it is represented by a sequence $1^{c_0}0\cdots01^{c_l}$ on $\{0,1\}^*$, and hence by a single number
$$p(0)^{c_0+1} \cdot p(1)^{c_1+1} \cdot \cdots \cdot p(l)^{c_l+1}.$$
This number is called the Gödel number of program $\mathcal{P}$, denoted $\ulcorner\mathcal{P}\urcorner$.

- Any Turing machine $\mathcal{M}$ can be emulated by a TPL program $\mathcal{P}_\mathcal{M}$ on a universal Turing machine. So, the index (or code) of TM $\mathcal{M}$ is defined to be the Gödel number $\ulcorner\mathcal{P}_\mathcal{M}\urcorner$.

- $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ is a partial computable function if $\{1^{m_1}0\cdots01^{m_k}01^{f(m_1,\ldots,m_k)} : m_1, \ldots, m_k \in \mathbb{N}\}$ is a type-0 language.

  If it is realized by a TM $\mathcal{M}$ with index e, f is denoted by $\{e\}^k$ (or simply $\{e\}$). When e is not an index of TM, $\{e\}$ is regarded as a partial function with empty domain.

- Enumeration theorem: For any $n \geq 0$, there exists a natural number e_n such that for any d, $x_1, \ldots, x_n$,

$$\{e_n\}^{n+1}(d, x_1, \ldots, x_n) \sim \{d\}^n(x_1, \ldots, x_n).$$

  $f(x_1, \ldots, x_n) \sim g(x_1, \ldots, x_n)$ means either both sides are not defined or they are defined with the same value.

- A set $X \subset \mathbb{N}^n$ is said to be computably enumerable or CE if $\{1^{x_1}0 \cdots 01^{x_n} : (x_1, \ldots, x_n) \in X\}$ is type-0, i.e., the domain of a partial recursive function.

- X is said to be computable if both X and X^c are CE.

- If a function $f(x)$ has a finite value at $x = n$, we write $f(n)\downarrow$. Then we define a halting program K as follows.

$$\mathrm{K} = \{e : \{e\}(e)\downarrow\}.$$

  K is CE but not computable (by a diagonal argument).

§1.5. Partial recursive functions and CE sets

Definition 1.36 (Partial recursive functions)

The **partial recursive functions** are defined as follows.

1. Constant 0, Successor function $\mathrm{S}(x)$, Projections $\mathrm{P}_i^n(x_1, x_2, \cdots, x_n)$.
2. **Composition**. If $g_i : \mathbb{N}^n \to \mathbb{N}, h : \mathbb{N}^m \to \mathbb{N} (1 \leq i \leq m)$ are partial recursive functions, the composed function $f = h(g_1, \cdots, g_m) : \mathbb{N}^n \to \mathbb{N}$ defined by

$$f(x_1, \cdots, x_n) \sim h(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n))$$

is partial recursive, where $h(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n)) = z$ if each $g_i(x_1, \cdots, x_n) = y_i$ is defined and $h(y_1, \cdots, y_m) = z$.

Note: By $f(x_1, \cdots, x_n) \sim g(x_1, \cdots, x_n)$, we mean that either both functions are undefined or defined with the same value.

Definition 1.36 (Partial recursive functions, continued)

3. **Primitive recursion**. If $g : \mathbb{N}^n \to \mathbb{N}, h : \mathbb{N}^{n+2} \to \mathbb{N}$ are partial recursive, so is the function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ defined by

$$\begin{aligned} f(x_1, \cdots, x_n, 0) &\sim g(x_1, \cdots, x_n) \\ f(x_1, \cdots, x_n, y+1) &\sim h(x_1, \cdots, x_n, y, f(x_1, \cdots, x_n, y)). \end{aligned}$$

4. **Minimization**. If $g : \mathbb{N}^{n+1} \to \mathbb{N}$ is a partial recursive function, so is the function $f : \mathbb{N}^n \to \mathbb{N}$ defined by

$$f(x_1, \cdots, x_n) \sim \mu y(g(x_1, \cdots, x_n, y) = 0).$$

Here, $\mu y(g(x_1, \cdots, x_n, y) = 0) = c$ if "$g(x_1, \cdots, x_n, c) = 0$, and for $z < c$, $g(x_1, \cdots, x_n, z)$ is defined with non-zero values"; if there is no such c, then $\mu y(g(x_1, \cdots, x_n, y) = 0)$ is undefined.

Theorem 1.37 (Kleene normal form theorem)

There are a primitive recursive function $U(y)$ and a primitive recursive relation $T_n(e, x_1, \cdots, x_n, y)$ such that any partial computable function $f(x_1, \cdots, x_n)$, there exists e such that

$$f(x_1, \cdots, x_n) \sim U(\mu y T_n(e, x_1, \cdots, x_n, y)),$$

where $\mu y T_n(e, x_1, \cdots, x_n, y)$ takes the smallest value y satisfying $T_n(e, x_1, \cdots, x_n, y)$; if there is no such y, it is undefined.

In other words, every partial computable function can be expressed by fixed primitive recursive function U and relation T_n by applying μ-operator to T_n, and thus it is a partial recursive function.

Proof.

- We define a relation $T_n(e, x_1, \cdots, x_n, y)$ as follows:
  "y is the Gödel number (code) of the whole computation process γ of a universal TM with input $(e, x_1, \cdots, x_n)$"
  Note that γ essentially includes the computation process of TM with index e on input $(x_1, \cdots, x_n)$, which is actually emulated by a universal TM.
- γ is a sequence of configurations $\alpha_0 \triangleright \alpha_1 \triangleright \cdots \triangleright \alpha_n$ with the initial α_0 and a final α_n, which can regarded as a word over $\Omega \cup Q \cup \{\triangleright\}$.
- In general, we cannot decide if such a computation process γ exists or not. But for a given γ, we can easily check that for each $i < n$, $\alpha_i \triangleright \alpha_{i+1}$ is a valid transition, as well as α_0 and α_n are the first and last configurations. Thus, it is primitive recursive to check that y is a code for such a γ.
- Finally, let $U(y)$ be a primitive recursive function that extracts the output information from the last configuration α_n of γ coded by y. Then, $U(\mu y T_n(e, x_1, \cdots, x_n, y))$ is nothing but the result of a TM with index e on input $(x_1, \cdots, x_n)$. □

Corollary 1.38

Every partial computable function is a partial recursive function.
Every computable function is recursive.

Corollary 1.39

There is a primitive recursive function $T_n(e, x_1, \ldots, x_n, y)$ such that for any CE set $X \subset \mathbb{N}^n$, there exists e such that

$$(x_1, \ldots, x_n) \in X \Leftrightarrow \exists y\, T_n(e, x_1, \ldots, x_n, y).$$

Theorem 1.40 (Parameter theorem [1])

For any $m, n \geq 1$, there exists a primitive recursive function $S_n^m : \mathbb{N}^{m+1} \to \mathbb{N}$ such that

$$\{e\}^{m+n}(x_1, \cdots, x_n, y_1, \cdots, y_m) \sim \{S_n^m(e, y_1, \cdots, y_m)\}^n(x_1, \cdots, x_n).$$

Remark [2]

- This theorem says that input data $(y_1, \cdots, y_m)$ may be treated as parameters in a TPL program.
- Roughly speaking, the Enumeration and Parameter theorems are inverses to one another: the former pushes an index out to an input variable while the latter pulls input variables into a index (as parameters).

[1] Also known as "S-m-n theorem", which comes from the symbol S_n^m in the statememt of the theorem.

[2] cf. *Recursively Enumerable Sets and Degrees* by Robert I. Soare

Proof.

- Suppose $\mathcal{P}$ is a TPL program with Gödel number $\{e\}^{m+n}$ and a sequence $(y_1, \ldots, y_m)$ is given.
- First construct a TPL program $\mathcal{P}'$ that rewrites an input string $1^{x_1}0\cdots01^{x_n}$ to $1^{x_1}0\cdots01^{x_n}01^{y_1}0\cdots01^{y_m}$. The Gödel number of $\mathcal{P}'$ can be obtained by the primitive recursive function of $(y_1, \ldots, y_m)$.
- We then define a program $\mathcal{P}''$ by connecting $\mathcal{P}'$ with $\mathcal{P}$ with the following two modifications: letting k be the number of the last line in $\mathcal{P}'$,
  - Rewrite all "**halt**" in $\mathcal{P}'$ as "**goto** $k+1$".
  - In $\mathcal{P}$, change l in "**goto** l" and "**if** ? **then goto** l" to $k+1+l$.
- Thus, the Gödel number of $\mathcal{P}''$ are obtained as a primitive recursive function of e and $(y_1, \cdots, y_m)$. Denote it as S_n^m. This completes the proof. □

Theorem 1.41 (Recursion theorem)

Let $f(x_1, \cdots, x_n, y)$ be a partial recursive function. There exists e such that

$$\{e\}^n(x_1, \cdots, x_n) \sim f(x_1, \cdots, x_n, e).$$

Proof.
By parameter theorem, there is a primitive recursive function $g(y)$ such that $\{g(y)\}(x_1, \cdots, x_n) \sim \{y\}(x_1, \cdots, x_n, y)$.

Let d be the index of a partial computable function $f(x_1, \cdots, x_n, g(y))$.
Then let $e = g(d)$. We have

$$\{g(d)\}(x_1, \cdots, x_n) \sim \{d\}(x_1, \cdots, x_n, d) \sim f(x_1, \cdots, x_n, g(d))$$

□

There are infinitely many e satisfying the theorem, because there are infinitely many functions S_n^m for the parameter theorem.

From the recursion theorem, we can derive the following theorem, which is also often called the recursion theorem.

Theorem 1.42 (Fixed point theorem)

Let $\sigma(x)$ be a computable function. Then, there exists an e such that

$$\{e\}^n(x_1, \cdots, x_n) \sim \{\sigma(e)\}^n(x_1, \cdots, x_n).$$

Such an e is called a fixed point of σ.

- The fixed point theorem immediately follows from the recursion theorem, since the right-hand side of the former can be viewed as a special case of the right-hand side of the latter.
- Conversely, the fixed point theorem also leads to the recursion theorem with the help of the parameter theorem.

- In order to see that a fixed point e can be arbitrarily large, we take any N. Let $\{e\}$ be a partial recursive function different from any of $\{0\}, \{1\}, \cdots, \{N\}$. For a given computable function $\sigma(x)$, define

$$\sigma'(x) = \begin{cases} e & \text{if} \quad x \leq N \\ \sigma(x) & \text{if} \quad x > N \end{cases}$$

  Any fixed point of $\sigma'(x)$ is greater than N and is also a fixed point of $\sigma(x)$.

Exercise 1.5.1

Let $\sigma(x, y)$ be a computable function. Prove the existence of computable function g such that

$$\{g(y)\}^n(x_1, \cdots, x_n) \sim \{\sigma(g(y), y)\}^n(x_1, \cdots, x_n).$$

Hint: Consider a computable function $h(x)$ such that $\{\{x\}(x)\} \sim \{h(x)\}$ and then $\sigma(h(x), y)$ is expressed as $\{S(y)\}(x)$ by the parameter theorem.

Example 18. Application of the Recursion Theorem

- Choose any recursive function f whose range is an infinite set. Then, the set $A = \{x : \forall y < x f(x) \neq f(y)\}$ is also infinite.
- We want to make a recursive function g that lists the elements of A from the smallest in the usual order. By primitive recursion, g might be defined as

$$\begin{aligned} g(0) &= 0 \\ g(x) &= \mu w \forall y \leq g(x \dot{-} 1) f(w) \neq f(y). \end{aligned}$$

  But, this definition uses primitive recursion and minimization at the same time, which does not fit our definition of recursive functions.
- However, by the recursion theorem, we have a partial recursive function g s.t.

$$g(x) \sim \begin{cases} 0 & \text{if } x = 0 \\ \mu w \forall y \leq g(x \dot{-} 1) f(w) \neq f(y) & \text{otherwise} \end{cases}$$

  Since it is easy to prove by induction that this g is total, g is the desired recursive function.

Self-Study: Ackermann function

The following function f is called the **Ackermann function**.

$$\begin{cases} f(0,y) &= y+1, \\ f(x+1,0) &= f(x,1), \\ f(x+1,y+1) &= f(x,f(x+1,y)) \end{cases}$$

Wilhelm Ackermann
(1896 - 1962)

- The Ackermann function is a total computable function that is not primitive recursive.
- The Ackermann function grows rapidly, that is, faster than any primitive recursive function.

We show Ackermann function is a total recursive function.

To this end, we defined the following primitive recursive function.

$$c(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{if } x \neq 0 \end{cases}, \quad \bar{c}(x) = 1 - c(x)$$

Then, by the recursion theorem, there is an index e such that

$$\begin{aligned} \{e\}(x,y) \quad \sim \quad & (y+1)\bar{c}(x) + \{e\}(x \dot{-} 1, y)c(x)\bar{c}(y) \\ & + \{e\}(x \dot{-} 1, \{e\}(x, y \dot{-} 1))c(x)c(y). \end{aligned}$$

By induction, we can verify that the function $\{e\}$ satisfies the conditions of the Ackermann function. Finally, by Corollary 1.38, this function is recursive.

Exercise 1.5.2

Show that the Ackermann function is not primitive recursive.
Hint: For any primitive recursive function $g(x,y)$ there exists a c such that

$$g(x,y) < f(c, \max\{x,y\}).$$

Recursive relations

Definition 1.43

A relation $R \subset \mathbb{N}^n$ is **recursive** if its characteristic function χ_R is recursive.

Lemma 1.44

If n-ary relations A, B are recursive, also are the followings:

$$\neg A, A \wedge B, A \vee B, \forall y < z A, \exists y < z A.$$

Lemma 1.45

Given two recursive n-ary functions g, h and a recursive n-ary relation R, f defined below is also recursive.

$$f(x_1, \cdots, x_n) = \begin{cases} g(x_1, \cdots, x_n), & \text{if } R(x_1, \cdots, x_n) \\ h(x_1, \cdots, x_n), & \text{otherwise} \end{cases}$$

Lemma 1.46

The graph of a recursive function is recursive.

Lemma 1.47

A total function with a recursive graph is a recursive function.

Proof. A function with graph $F(x, y)$ is represented by $\mu y F(x, y)$, or more precisely, $\mu y\ (1 - \chi_F(x, y) = 0)$. So a function with a recursive graph is partial recursive. If it is total, by Corollary 1.38 (or Kleene's normal form theorem) it is a recursive function. □

Exercise 1.5.3 (Challenging)

Show that the graph of the Ackermann function is a primitive recursive set.

Definition 1.48

An n-ary relation $R \subset \mathbb{N}^n$ is **recursively enumerable**, RE for short, if the following (partial) function f is partial recursive:

$$f(x_1, \cdots, x_n) = \begin{cases} 1 & \text{if } R(x_1, \cdots, x_n) \text{ holds,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- Obviously, the domain of any partial recursive function is RE. By the next lemma, we will see that the range of any (partial) recursive function is also RE, and the name "recursively enumerable" actually describes this property.
- As previously shown, "recursive" and "computable" are interchangeable, so "recursively enumerable (RE)" and "computably enumerable (CE)" are also interchangeable.
- In this course, we will use CE more frequently than RE.

Major conditions equivalent to RE are summarized in the next lemma.

Lemma 1.49

For a relation $R \subset \mathbb{N}^n$, the following conditions are equivalent.

(1) R is recursively enumerable.

(2) R is an empty set or the range of some primitive recursive function.

(3) R is a finite set or the range of some recursive injection (1-to-1 function).

(4) R is an empty set or the range of some recursive function.

(5) R is the range of some partial recursive function.

(6) There exists a primitive recursive relation S such that

$$R(x_1, \cdots, x_n) \Leftrightarrow \exists y S(x_1, \cdots, x_n, y).$$

(7) There exists a recursive relation S such that

$$R(x_1, \cdots, x_n) \Leftrightarrow \exists y S(x_1, \cdots, x_n, y).$$

Proof. To show $(1) \Rightarrow (6) \Rightarrow (7) \Rightarrow (1)$.

- $(1) \Rightarrow (6)$. Consider a partial recursive function f such that

$$f(x_1, \cdots, x_n) = \begin{cases} 1 & \text{if } R(x_1, \cdots, x_n), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

  By Kleene's normal form theorem, there exist primitive recursive U and T s.t.

$$f(x_1, \cdots, x_n) \sim U(\mu y T(x_1, \cdots, x_n, y)).$$

  Thus, $R(x_1, \cdots, x_n) \Leftrightarrow \exists y T(x_1, \cdots, x_n, y)$.

- $(6) \Rightarrow (7)$ is obvious.

- $(7) \Rightarrow (1)$. Let S be a recursive relation such that

$$R(x_1, \cdots, x_n) \Leftrightarrow \exists y S(x_1, \cdots, x_n, y).$$

  If U is defined to be a primitive recursive function such that $U(y) = 1$ for all y, $f(x_1, \cdots, x_n) \sim U(\mu y S(x_1, \cdots, x_n, y))$ satisfies Definition 1.48, so (1) holds.

To show $(2) \Rightarrow (3) \Rightarrow (4) \Rightarrow (5) \Rightarrow (6) \Rightarrow (2)$.

- $(2) \Rightarrow (3)$ The recursive function g in Example 18 (p.16) will be used to make a given function injective. • $(3) \Rightarrow (4) \Rightarrow (5)$ is obvious.
- $(5) \Rightarrow (6)$. Suppose R is the range of a partial recursive function f. By Kleene's normal form theorem, there exist primitive recursive U and T such that $f(x) \sim U(\mu y T(x, y))$.

$$\begin{aligned} z \in \mathrm{range}(f) &\Leftrightarrow \exists x \exists y (T(x, y) \wedge U(y) = z) \\ &\Leftrightarrow \exists x (T(\mathrm{c}(x, 0), \mathrm{c}(x, 1)) \wedge U(\mathrm{c}(x, 1)) = z), \end{aligned}$$

  where x codes a pair (x_0, x_1) and x_i is given by primitive recursive $\mathrm{c}(x, i)$.
- $(6) \Rightarrow (2)$. Suppose that S is primitive recursive and $R(x) \Leftrightarrow \exists y S(x, y)$. We may assume that R is non-empty and choose any $d \in R$. Then define a primitive recursive function g as follows:

$$g(x) = \begin{cases} \mathrm{c}(x, 0) & \text{if } S(\mathrm{c}(x, 0), \mathrm{c}(x, 1)), \\ d & \text{otherwise.} \end{cases}$$

  Then, $\mathrm{range}(g)$ is the same as R.

Lemma 1.50

If n-ary relations A, B are CE, also are the following relations.

$$A \wedge B,\ A \vee B,\ \forall y < z A,\ \exists y < z A,\ \text{and } \exists y A.$$

Lemma 1.51

The graph of a partial recursive function is CE.

Lemma 1.52

A function whose graph is CE is a partial recursive.

Exercise 1.5.4 (Homework for everybody)

Show that any infinite CE set contains an infinite computable subset.

Summary

- The set of all **partial recursive functions**: the smallest class that contains the constant 0, successor function, projection, and closed under composition, primitive recursion and minimalization.
- **Kleene normal form theorem**: every partial recursive function can be obtained from two fixed primitive recursive functions by applying μ-operator to one of them.
- **Parameter theorem**: input can be seen as parameters in the TPL program.
- **Recursion theorem**: A function to define can be used in the definition.
- Definition of **recursively enumerable relation** and its equivalent statements.

Further readings

N. Cutland. *Computability: An Introduction to Recursive Function Theory*, Cambridge University Press, 1st edition, 1980.

Thank you for your attention!