Logic and Foundation I

Part 1. Equational system

Kazuyuki Tanaka

BIMSA

October 21, 2023

北京雁栖湖
应用数学研究院
Beijing Institute
of Mathematical
Sciences and Applications

Logic and Foundations I

- **Part 1. Equational theory**
- **Part 2. First order theory**
- **Part 3. Model theory**
- **Part 4. First order arithmetic and incompleteness theorems**

Part 1. Schedule

- Sep. 21, (1) Formal systems of equation
- Sep. 28, (2) Free algebras and Birkhoff's theorem
- Oct. 12, (3) Boolean algebras
- Oct. 19, (4) Computable functions and general recursive functions

Recap: Boolean algebra & disjunctive normal form

- The theory of **Boolean algebra** (BA) is defined as a lattice $(L, \vee, \wedge)$ with the negation operation $\neg$ and constants $0, 1$.
- $\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_n$ is also written as $\bigvee_{i=1,\ldots,n} \varphi_i$.
  We also write x^1 for x and x^0 for $\neg x$.

Theorem (Disjunctive normal form)

$$\mathrm{BA} \vdash \varphi(x_1, x_2, \ldots, x_n) = \bigvee_{f_\varphi(b_1,\ldots,b_n)=1} x_1^{b_1} \wedge x_2^{b_2} \wedge \cdots \wedge x_n^{b_n}.$$

If there is no $b_1, \ldots, b_n$ such that $f_\varphi(b_1, \ldots, b_n) = 1$, then the right-hand side is put as 0.

- To make the disjunctive normal form unique up to equality, we discard the unnecessary variables and rewrite it in the minimal number of variables. Such a form is called the strict disjunctive normal form. For instance, $(x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$ should be rewritten as x_1. If all the variables are unnecessary, it should be written as 1.

Recap: Free Boolean algebra & completeness

- Let $\mathrm{Term}(X)$ be the set of terms of Boolean algebra with variables in X.
  Let $\mathcal{T}(X)$ be the term algebra and $\equiv_E$ a congruence relation on $\mathrm{Term}(X)$ defined by $s \equiv_E t \Leftrightarrow \mathsf{BA} \vdash s = t$. Then,

  Theorem

  $\mathcal{T}(X)/{\equiv_E}$ is the free Boolean algebra.

- $\mathrm{Term}(X)/{\equiv_E}$ can be viewed as the set of strict disjunctive normal forms in X.
- From propositional logic to Boolean algebra. We eliminate the operation $\to$ in a proposition by $\varphi \to \psi := \neg\varphi \vee \psi$. Then we have

  $$\text{(prop. logic)} \models \varphi \;\Leftrightarrow\; \mathcal{T}(X)/{\equiv_E} \models [\varphi] = 1 \;\Leftrightarrow\; \mathrm{BA} \vdash \varphi = 1.$$

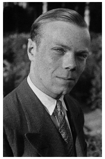
- Homework. Can you prove (prop. logic) $\vdash \varphi \;\Leftrightarrow\; \mathrm{BA} \vdash \varphi = 1$?

Today's topics

1. Recursive functions
2. Introduction to Turing machines

Introduction

- Roughly speaking, a function that can be realized as a relationship between input and output using a computer is called a **computable function**.
- Here, we will consider computable functions from (sets of) natural numbers to natural numbers (so-called number-theoretic functions).
- Research on such families of functions began with Gödel's paper on the incompleteness theorem (1931), but his lectures in 1934, Gödel (borrowing an idea from Herbrand (1931)) used equation theory to define general recursive functions.
- Later, Kleene (1936), who had attended Gödel's lectures, gave the definition of today's recursive function and proved its equivalence with general recursive functions.

Kurt Gödel

Jacques Herbrand

Stephen C. Kleene

Kleene's definition of recursive functions

Definition (Recursive function - 1/3)

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

1. **Zero function** $Z() = 0$, **Successor function** $\mathrm{S}(x) = x + 1$, **Projection** $\mathrm{P}^n_i(x_1, x_2, \cdots, x_n) = x_i$ $(1 \leq i \leq n)$ are recursive functions.

$2a$. **Composition**. If $g_i : \mathbb{N}^n \to \mathbb{N}, h : \mathbb{N}^m \to \mathbb{N}(1 \leq i \leq m)$ are recursive functions, the composed function $f = h(g_1, \cdots, g_m) : \mathbb{N}^n \to \mathbb{N}$ defined by

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

is recursive, where

$$h(g_1(x_1, \cdots, x_n), \cdots, g_m(x_1, \cdots, x_n)) = z$$

means $g_i(x_1, \cdots, x_n) = y_i$ and $h(y_1, \cdots, y_m) = z$.

Kleene's definition of a recursive function

Definition (Recursive function - 2/3)

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

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

is recursive.

Note that the functions obtained from the above condition $1, 2a, 2b$ are called **primitive recursive function**.

Kleene's definition of recursive functions

Definition (Recursive function - 3/3)

$2c$. **Minimization**.

- Let $g : \mathbb{N}^{n+1} \to \mathbb{N}$ be a recursive function.
- Suppose that for all $x_1, \ldots, x_n$, there exists y such that $g(x_1, \ldots, x_n, y) = 0$.
- Then $f : \mathbb{N}^n \to \mathbb{N}$ satisfying

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

is recursive. Here, μy means "the minimum y such that".

Example 15

Predecessor function: $\mathrm{M}(x) = x - 1$ $(x > 0)$, $\mathrm{M}(x) = 0$ $(x = 0)$ is (primitive) recursive, by the following definition.

$$\begin{cases} \mathrm{M}(0) = 0, \\ \mathrm{M}(x+1) = x = \mathrm{P}^2_1(x, \mathrm{M}(x)). \end{cases}$$

Example 16

Addition: $\mathrm{plus}(x, y) = x + y$ is (primitive) recursive.

$$\begin{cases} \mathrm{plus}(x, 0) = x, \\ \mathrm{plus}(x, y+1) = \mathrm{S}(\mathrm{plus}(x, y)). \end{cases}$$

Example 17

Subtraction: $x \dot{-} y$ is (primitive) recursive.

$$\begin{cases} x \dot{-} 0 = x, \\ x \dot{-} (y+1) = \mathrm{M}(x \dot{-} y). \end{cases}$$

Problem 9

Prove $x \cdot y$, x^y, $x!$, $\max\{x, y\}$, $\min\{x, y\}$ are primitive recursive functions.

Problem 10

Let $f(x_1, \ldots, x_n, y)$ be a primitive recursive function. Prove the following functions are also primitive recursive.

$$F(x_1, \ldots, x_n, z) = \Sigma_{y<z} f(x_1, \ldots, x_n, y),$$

$$G(x_1, \ldots, x_n, z) = \Pi_{y<z} f(x_1, \ldots, x_n, y).$$

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

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

W. Ackermann (1886 - 1962)

- Although it is easy to see from the definition that the Ackermann function is a computable function, it is not so easy to show that it is a recursive function.
- A function defined by an equation in this way is called a **general recursive function**, and we can ultimately show that it is equivalent to a recursive function.
- Ackermann function is not primitively recursive.

Problem 11 (Hard)

Let $f(x,y)$ be a Ackermann function. Show that for any primitive recursive function $g(x,y)$ there exists a c such that $g(x,y) < f(c, \max\{x,y\})$. Then show that $f(x,y)$ is not primitive recursive.

To define a general recursive function, we fix the language consisting of constant $\mathtt{0}$, the successor symbol $\mathtt{S}(x)$ and countably many function symbols $\mathtt{f}_0, \mathtt{f}_1, \ldots$. From now on, a "term" means a term in this language.

Definition (Herbrand-Gödel general recursive function)

$f : \mathbb{N}^n \to \mathbb{N}$ is said to general recursive if there is a finite set of equations E and a function symbol $\mathtt{f}(x_1, \ldots, x_n)$, and for any $a_1, \ldots, a_n, b \in \mathbb{N}$, the following holds

$$f(a_1, \ldots, a_n) = b \Leftrightarrow E \vdash \mathtt{f}(\overline{a}_1, \ldots, \overline{a}_n) = \overline{b}.$$

Here, $\overline{a} = \overbrace{\mathtt{S}(\mathtt{S}(\mathtt{S}(\cdots \mathtt{S}}^{a \text{ times}}(0) \cdots)))$.

Example 18

- $f(x,y) = x + y$ is general recursive.
- Let $E = \{\mathtt{f}(x,0) = x,\ \mathtt{f}(x,\mathtt{S}(y)) = \mathtt{S}(\mathtt{f}(x,y))\}$. Then

$$a + b = c \Leftrightarrow E \vdash \mathtt{f}(\overline{a},\overline{b}) = \overline{c}$$

- If we consider that $+$ on the left side is defined as in Example 16 in Page 10, this definition is nothing but the interpretation of E on $\mathbb{N}$.
- Thus, for each number a, b, c, the equation $\mathtt{f}(\bar{a},\bar{b}) = \bar{c}$ holds in $\mathbb{N}$ iff it is provable from E.

- A structure $\mathfrak{M} = (\mathbb{N}; 0, \mathrm{S}, f_i)_{i \in \mathbb{N}}$ is called a **standard structure** if $0 \in \mathbb{N}$ and $\mathrm{S} \colon \mathbb{N} \to \mathbb{N}$ are interpreted in the standard way, while f_i is arbitrary.
- In the standard structure $\mathfrak{M}$, the interpretation of the numeral $\bar{a}$ is

$$(\bar{a})^{\mathfrak{M}} = a.$$

- However, it is not generally true that an equation is provable in E iff it holds in the standard structure satisfying E.
- For instance, even if E is consistent, it may not have a standard model.

Theorem

All recursive functions are general recursive.

Proof. It is sufficient to show that the entire general recursive function includes $\mathrm{Z}, \mathrm{S}, \mathrm{P}_i^n$ and is closed with composition, primitive recursion, and minimization.

Zero function

Let $E = \{\mathtt{f}() = \mathtt{0}\}$. We show that for any $b \in \mathbb{N}$, $Z() = b \Longleftrightarrow E \vdash \mathtt{f}() = \overline{b}$.

$(\Rightarrow)$ If $\mathrm{Z}() = b$, then $b = 0$. Since $E \vdash \mathtt{f}() = \mathtt{0}$ and $\mathtt{0} = \overline{0}$, we have $E \vdash \mathtt{f}() = \overline{b}$.

$(\Leftarrow)$ By contraposition, let $\mathrm{Z}() \neq b$.
Then, in the standard structure $\mathfrak{M}$ that satisfies E, $\mathtt{f}^{\mathfrak{M}}() = 0 = \mathrm{Z}^{\mathfrak{M}}() \neq \overline{b}^{\mathfrak{M}} = b$.
So by the completeness (soundness) theorem of the equation theory, $E \nvdash \mathtt{f}() = \overline{b}$.

Successor function

Let $E = \{\mathtt{f}(x) = \mathtt{S}(x)\}$. For any $a, b \in \mathbb{N}$, prove $S(a) = b \Leftrightarrow E \vdash \mathtt{f}(\overline{a}) = \overline{b}$ as above.

Projection

Let $E = \{\mathtt{f}(x_1, \ldots, x_i, \ldots, x_n) = x_i\}$. For any $\vec{a} \in \mathbb{N}^n$, $b \in \mathbb{N}$, prove $\mathrm{P}_i^n(\vec{a}) = b \Leftrightarrow E \vdash \mathtt{f}(\overline{\vec{a}}) = \overline{b}$ also as above.

Composition(1)

Let $f\colon \mathbb{N}^n \to \mathbb{N}$, $g_i\colon \mathbb{N}^n \to \mathbb{N}$ $(1 \leq i \leq m)$, $h\colon \mathbb{N}^m \to \mathbb{N}$ satisfy

$$f(\vec{a}) = h(g_1(\vec{a}), \ldots, g_m(\vec{a})) \quad (\forall \vec{a} \in \mathbb{N}^n).$$

By the induction hypothesis, we may assume that for g_i $(1 \leq i \leq m)$ and h, there exist equational theories $E_{g_i}(1 \leq i \leq m)$ and E_h such that the following hold:

$$g_i(\vec{a}) = b \Leftrightarrow E_{g_i} \vdash \mathtt{g}_i(\vec{\bar{a}}) = \bar{b} \quad (\text{for } \vec{a} \in \mathbb{N}^n,\ b \in \mathbb{N}, 1 \leq i \leq m);$$
$$h(\vec{a}) = b \Leftrightarrow E_h \vdash \mathtt{h}(\vec{\bar{a}}) = \bar{b} \quad (\text{for } \vec{a} \in \mathbb{N}^m,\ b \in \mathbb{N}).$$

We may also assume that no function symbols appear in common in two or more of $E_{g_i}(1 \leq i < m)$ and E_h. Then, using a new function symbol $\mathtt{f}$, we set

$$E = E_{g_1} \cup \cdots \cup E_{g_m} \cup E_h \cup \{\mathtt{f}(\vec{x}) = \mathtt{h}(\mathtt{g}_1(\vec{x}), \ldots, \mathtt{g}_m(\vec{x}))\}$$

Here, $\vec{x}$ represents the variable sequence $x_1, \ \ldots \ , x_n$, Then we will show $f(\vec{a}) = b \Leftrightarrow E \vdash \mathtt{f}(\vec{\bar{a}}) = \bar{b}$ in the following slides.

Composition(2)

($\Rightarrow$) Take any $\vec{a} \in \mathbb{N}^n$, $b \in \mathbb{N}$ such that $f(\vec{a}) = b$. For each $1 \leq i \leq m$, by the definition of E_{g_i} and $g_i(\vec{a}) = g_i(\vec{a})$, we have

$$E_{g_i} \vdash \mathtt{g}_i(\vec{\bar{a}}) = \overline{g_i(\vec{a})} \quad (\vec{a} \in \mathbb{N}^n).$$

Also, by the definition of E_h and $f(\vec{a}) = h(g_1(\vec{a}), \ldots, g_m(\vec{a})) = b$, we have

$$E_h \vdash \mathtt{h}\left(\overline{g_1(\vec{a})}, \ldots, \overline{g_m(\vec{a})}\right) = \bar{b}.$$

Then, by substitution rule (sub),

$$E \vdash \mathtt{h}(\mathtt{g}_1(\vec{\bar{a}}), \ldots, \mathtt{g}_m(\vec{\bar{a}})) = \bar{b}.$$

Finally, by the last equation in E,

$$E \vdash \mathtt{f}(\vec{\bar{a}}) = \mathtt{h}(\mathtt{g}_1(\vec{\bar{a}}), \ldots, \mathtt{g}_m(\vec{\bar{a}})).$$

So $E \vdash \mathtt{f}(\vec{\bar{a}}) = \bar{b}$. The converse can be shown by contrapositive in the same way.

Primitive recursion (1)

Let $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$, $g\colon \mathbb{N}^n \to \mathbb{N}$, $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ satisfy

$$\begin{cases} f(\vec{a},0) = g(\vec{a}) \\ f(\vec{a},b+1) = h(\vec{a},f(\vec{a},b)) \end{cases} \qquad (\vec{a} \in \mathbb{N}^n,\ b \in \mathbb{N})$$

By induction hypothesis, there exist E_g and E_h such that:

$$g\left(\vec{a}\right) = c \Leftrightarrow E_g \vdash \mathtt{g}\left(\bar{\vec{a}}\right) = \bar{c} \quad (\text{for } \vec{a} \in \mathbb{N}^n, c \in \mathbb{N});$$
$$h\left(\vec{a},b\right) = c \Leftrightarrow E_h \vdash \mathtt{h}\left(\bar{\vec{a}},\bar{b}\right) = \bar{c} \quad (\text{for } \vec{a} \in \mathbb{N}^n,\ b,c \in \mathbb{N}).$$

By replacing appropriate symbols, we assume there are no function symbols that appear in common in two or more of E_g and E_k. Then, taking distinct variables $\vec{x} = x_1,\ldots,x_n,y$ and a new function symbol $\mathtt{f}$, we put

$$E = E_g \cup E_h \cup \{\mathtt{f}\left(\vec{x},\mathtt{0}\right) = \mathtt{g}\left(\vec{x}\right)\} \cup \{\mathtt{f}\left(\vec{x},\mathtt{S}(y)\right) = \mathtt{h}\left(\vec{x},\mathtt{f}\left(\vec{x},y\right)\right)\}$$

Then, for any $\vec{a} \in \mathbb{N}^n$, $b,c \in \mathbb{N}$, we will show

$$f\left(\vec{a},b\right) = c \Leftrightarrow E \vdash \mathtt{f}\left(\bar{\vec{a}},\bar{b}\right) = \bar{c} \qquad (*)$$

Primitive recursion (2)

- First, we show $\Rightarrow$ by induction on $b \in \mathbb{N}$.

<u>The base case: $b = 0$</u>
If $f(\vec{a}, 0) = c$, then $g(\vec{a}) = c$. So by definitions of E_g and E

$$E_g \vdash \mathtt{g}\left(\bar{\vec{a}}\right) = \bar{c}, \quad E \vdash \mathtt{f}\left(\bar{\vec{a}}, \mathtt{0}\right) = \mathtt{g}\left(\bar{\vec{a}}\right).$$

Therefore,

$$E \vdash \mathtt{f}\left(\bar{\vec{a}}, \mathtt{0}\right) = \bar{c},$$

that is, $E \vdash \mathtt{f}\left(\bar{\vec{a}}, \bar{b}\right) = \bar{c}$.
<u>Induction step</u>
Assume that $(*)$ holds with a certain $b \in \mathbb{N}$ for any $\vec{a} \in \mathbb{N}^n$, $c \in \mathbb{N}$. Also, suppose that $f(\vec{a}, b+1) = c$ holds. By the induction hypothesis and $f(\vec{a}, b) = f(\vec{a}, b)$, we have

$$E \vdash \mathtt{f}\left(\bar{\vec{a}}, \bar{b}\right) = \overline{f(\vec{a}, b)}.$$

Also, by the definition of E

$$E \vdash \mathtt{f}\left(\bar{\vec{a}}, \mathtt{S}\left(\bar{b}\right)\right) = \mathtt{h}\left(\bar{\vec{a}}, \mathtt{f}\left(\vec{a}, b\right)\right).$$

Primitive recursion (3)

In addition, by the substitution rules,

$$E \vdash \mathtt{f}(\vec{\bar{a}}, \overline{b+1}) = \mathtt{h}\left(\vec{\bar{a}}, \overline{f(\vec{a}, b)}\right).$$

Furthermore, by the definition of E_h and $h(\vec{a}, f(\vec{a}, b)) = f(\vec{a}, b+1) = c$

$$E_h \vdash \mathtt{h}\left(\vec{\bar{a}}, \overline{f\,(\vec{a}, b)}\right) = \bar{c}.$$

Therefore,

$$E \vdash \mathtt{f}(\vec{\bar{a}}, \overline{b+1}) = \bar{c}.$$

Thus we prove $(\Rightarrow)$.

- Next, $(\Leftarrow)$ will be proved by contrapositive.

Let $f(\vec{a}, b) = d \neq c$. From what was shown above, $E \vdash \mathtt{f}(\vec{\bar{a}}, \bar{b}) = \bar{d}$, so a certain standard model $\mathfrak{M}$ such that $\mathfrak{M} \models \mathtt{f}(\vec{\bar{a}}, \bar{b}) = \bar{d}$. Hence, $\mathfrak{M} \not\models \mathtt{f}(\vec{\bar{a}}, \bar{b}) = \bar{c}$. Therefore, by the the completeness (soundness) of the equation theory $E \nvdash \mathtt{f}(\vec{\bar{a}}, \bar{b}) = \bar{c}$

Minimization (1)

We first check the following operations are general recursive. $+\colon \mathbb{N}^2 \to \mathbb{N}$ (sum), $\bullet\colon \mathbb{N}^2 \to \mathbb{N}$ (product), and $T\colon \mathbb{N} \to \mathbb{N}$ (true), $F\colon \mathbb{N} \to \mathbb{N}$ (false) as defined below:

$$T(a) = \begin{cases} 0 & (a = 0) \\ 1 & (a > 0) \end{cases} \quad F(a) = \begin{cases} 1 & (a = 0) \\ 0 & (a > 0) \end{cases}$$

In fact, we can easily construct the equational theories $E_+, E_\bullet, E_T, E_F$ such that:

$$\begin{aligned} a + b = c &\Leftrightarrow E_+ \vdash +(\bar{a}, \bar{b}) = \bar{c}, \\ a \bullet b = c &\Leftrightarrow E_\bullet \vdash \bullet(\bar{a}, \bar{b}) = \bar{c}, \\ T(a) = b &\Leftrightarrow E_T \vdash \mathtt{T}(\bar{a}) = \bar{b}, \\ F(a) = b &\Leftrightarrow E_F \vdash \mathtt{F}(\bar{a}) = \bar{b}. \end{aligned}$$

The functions $+, \bullet, T, F$ are all simple primitive recursive functions. So by the same arguments as before, they are general recursive.

Minimization (2)

Now, given a general recursive function $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$, we assume for all $\vec{a} \in \mathbb{N}^n$, there exists $b \in \mathbb{N}$ such that $g(\vec{a}, b) = 0$.
By induction, we may suppose that there exists an equational theory E_g such that:

$$g(\vec{a}, b) = c \Leftrightarrow E_g \vdash \mathtt{g}(\bar{\vec{a}}, \bar{b}) = \bar{c} \quad (\text{for } \vec{a} \in \mathbb{N}^n, b, c \in \mathbb{N}).$$

Then, we want to show that $f : \mathbb{N}^n \to \mathbb{N}$ defined by $f(\vec{a}) = \mu x(g(\vec{a}, x) = 0)$, is also general recursive.
Taking $\mathtt{f}, \mathtt{h}$ as new function symbols, let

$$E = E_+ \cup E_\bullet \cup E_T \cup E_F \cup E_g \cup \{\varphi(\vec{x}, y), \mathtt{f}(\vec{x}) = \mathtt{h}(\vec{x}, \mathtt{0})\},$$

where $\varphi(\vec{x}, y)$ is the equation $\mathtt{h}(\vec{x}, y) = +\Big(\bullet\big(\mathtt{T}(\mathtt{g}(\vec{x}, y)), \mathtt{h}(\vec{x}, \mathtt{S}(y))\big), \bullet\big(\mathtt{F}(\mathtt{g}(\vec{x}, y)), y\big)\Big)$.
Since the right side is rewritten as $T(g(\vec{x}, y)) \bullet h(\vec{x}, S(y)) + F(g(\vec{x}, y)) \bullet y$,
it is $h(\vec{x}, S(y))$ if $g(\vec{x}, y) > 0$, and is y if $g(\vec{x}, y) = 0$.
Thus, we note that if $g(\vec{a}, b) > 0$ for all $b < c$, then $h(\vec{a}, b) = h(\vec{a}, c)$ for all $b < c$, that is, $E \vdash \mathtt{h}(\bar{\vec{a}}, \bar{b}) = \mathtt{h}(\bar{\vec{a}}, \bar{c})$ is also obtained by the definition of $E_+, E_\bullet, E_T, E_F, E_g$, and rules for equations.

Minimization (3)

It remains to show that for any $\vec{a} \in \mathbb{N}^n$, $c \in \mathbb{N}$

$$f\left(\vec{a}\right) = c \Leftrightarrow E \vdash \mathtt{f}\left(\vec{\bar{a}}\right) = \bar{c} \qquad (**)$$

- First, let $f(\vec{a}) = c$. Then we have $\mu x\left(g\left(\vec{a}, x\right) = 0\right) = c$. That is, for all $b < c$, $g\left(\vec{a}, b\right) > 0$ and $g\left(\vec{a}, c\right) = 0$. Hence, $E \vdash \mathtt{h}\left(\vec{\bar{a}}, \mathtt{0}\right) = \mathtt{h}\left(\vec{\bar{a}}, \bar{c}\right)$ and $E \vdash \mathtt{h}\left(\vec{\bar{a}}, \bar{c}\right) = \bar{c}$. Finally, by the last equation $\mathtt{f}(\vec{x}) = \mathtt{h}(\vec{x}, \mathtt{0})$ in E, $E \vdash \mathtt{f}(\vec{\bar{a}}) = \mathtt{h}(\vec{\bar{a}}, \mathtt{0}) = \bar{c}$.
- Conversely, let $f(\vec{a}) = d \neq c$. Since $E \vdash \mathtt{f}(\vec{\bar{a}}) = \bar{d}$, there exists a standard model $\mathfrak{M}$ such that $\mathfrak{M} \models \mathtt{f}(\vec{\bar{a}}) = \bar{d}$. Therefore, $\mathfrak{M} \not\models \mathtt{f}(\vec{\bar{a}}) = \bar{c}$. Finally, $E \nvdash \mathtt{f}(\vec{\bar{a}}) = \bar{c}$. Thus, we have shown $(**)$.

Exercise

Define a function $f(x) = k$ by $x \equiv k \pmod 3$ and $k < 3$. By constructing an equational system, show that f is general recursive.

Introduction to Turing machines

- Next, we would like to show that the general recursive function is a recursive function.
- This was conjectured by Gödel and Church, and finally proved by Kleene. Since computation models such as Turing machines had not invented yet, Kleene used very complicated arguments by coding (Gödel numbers).
- Today, we will first introduce Turing machines, then show a general recursive function can be realized by a Turing machine. Finally, by showing that a function computable by a Turing machine is a recursive function, we establish the equivalence between the general recursive functions and the recursive functions.
- A Turing machine has an infinitely extendable tape, and it defines a function by corresponding the symbol strings written on the tape at the start to the symbol strings that remains on the tape when it stops.

Alan Turing

Stephen C. Kleene

Turing regarded a "computer" as a man who calculates with pencil and paper. To make the formulation simpler, he assumed the following conditions (Turing 1936).

- I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite.
- The behaviour of the computer at any moment is determined by the symbols which he is observing and his "state of mind" at that moment.
- We will also suppose that the number of states of mind which need be taken into account is finite.
- We may suppose that in a simple operation not more than one symbol is altered.

We may now construct a machine to do the work of this computer.

Definition

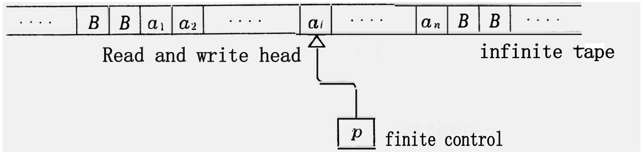
(Deterministic) Turing machine (TM) is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$,

(1) Q is a non-empty finite set of **states**.

(2) Ω is a non-empty finite set of **symbols**. The blank symbol $B \in \Omega$.

(3) $\delta : Q \times \Omega \to \Omega \times \{R, L, N\} \times Q$ is called a **transition function**.

(4) $q_0 \in Q$ is a **initial state**. (5) $F \subset Q$ is a set of **final states**.



- $\delta(p, a) = (b, x, q)$ means that at state p, if $\mathcal{M}$ reads symbol a at the head, then
  - the head write b to alter a,
  - according to $x = R$, L, N,
    the head moves to the right or the left or keep still,
    the state changes to q
- A configuration of TM, denoted $a_1 \cdots a_{i-1} p a_i \cdots a_n$, describes:
  - A string $a_1 \cdots a_n \in \Omega^*$ is written on the tape. All the symbols outside of $a_1 \cdots a_n$ on the tape are blank while the blank B may be included in the sequence,
  - the head is pointed at a_i on the tape,
  - the current state is p.

We say configuration α yields configuration α', denoted as $\alpha \triangleright \alpha'$, if there is a legal transition from configuration α to configuration α' as follows:

1) if $\delta(p, a_i) = (a'_i, L, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-2} q a_{i-1} a'_i a_{i+1} \cdots a_n \ (i > 1),$$
$$p a_1 a_2 \cdots a_n \triangleright q B a'_1 a_2 \cdots a_n.$$

2) if $\delta(p, a_i) = (a'_i, N, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-1} q a'_i a_{i+1} \cdots a_n.$$

3) if $\delta(p, a_i) = (a'_i, R, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-1} a'_i q a_{i+1} \cdots a_n \ (i \le n),$$
$$a_1 \cdots a_{n-1} a_n p \triangleright a_1 \cdots a_{n-1} a'_n B q.$$

We write the sequence of computation $\alpha_0 \triangleright \alpha_1 \triangleright \cdots \triangleright \alpha_n$ as $\alpha_0 \triangleright^* \alpha_n$ $(n \ge 0)$.

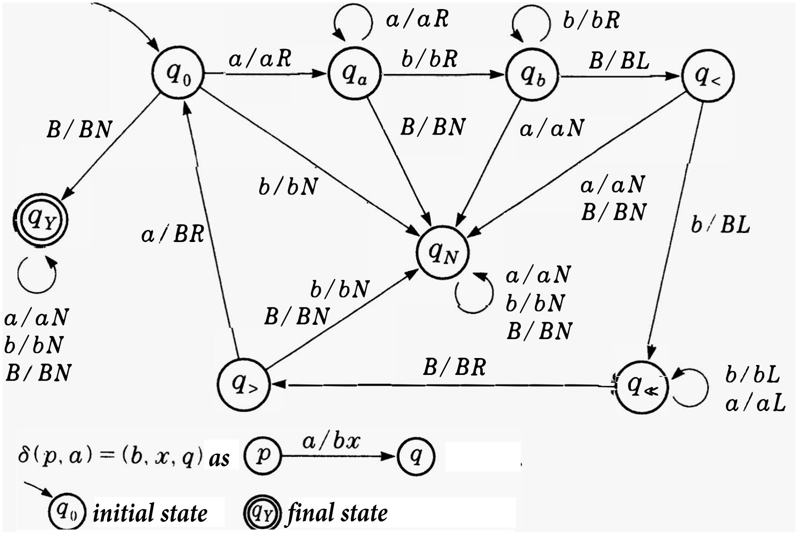
- We say $\mathcal{M}$ **accepts** $a_1 \cdots a_n \in (\Omega - \{B\})^*$ if there exists $b_1 \cdots b_m$ and $q \in F$ such that $q_0 a_1 \cdots a_n \rhd^* b_1 \cdots b_i q b_{i+1} \cdots b_m$. That is, some final state $q \in F$ is visited in the computation.
- The languages (of the strings) accepted by $\mathcal{M}$ is denoted as $L(\mathcal{M})$.

Example

$L = \{a^n b^n : n \geq 0\}$.

Example

$L = \{a^n b^n : n \geq 0\}$ is accepted by a TM.



- Up to now, the TM's we considered are devices that can decide whether an input is accepted or not.
- Notice that when the machine enter a final state, it leaves a string on the tape. If we regard such a string as an output of this TM for a given input, we can naturally define a function from strings to strings.
- This is called a (Turing) computable function.

Remark

- Such a function is partially defined, since the TM does not always terminate.
- To make the output unique, we define the output of (deterministic) TM as the string on the tape when the TM enters a final state for the first time, because it might enter a final state more than once.

- Let $\mathbb{N}$ be the set of all natural numbers. $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ is called a **number-theoretic function**.
- Turing definable function gives a mapping from strings to strings. It can be translated into a number-theoretic function.

Definition

A number-theoretic function $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ is **computable** if there is a Turing machine $\mathcal{M}$ accepts

$$1^{m_1}01^{m_2}0\cdots01^{m_k} := \underbrace{1\cdots1}_{m_1}0\underbrace{1\cdots1}_{m_2}0\cdots0\underbrace{1\cdots1}_{m_k}$$

and outputs

$$1^{f(m_1,\ldots,m_k)}.$$

We also say $\mathcal{M}$ **realizes** the function f.

However, we have

$$f \text{ is computable} \Leftrightarrow \{1^{m_1}0\cdots01^{m_k}01^{f(m_1,\ldots,m_k)} : m_1,\ldots,m_k \in \mathbb{N}\}$$
$$\text{is accepted by a TM with alphabet } \{0,1\}.$$

Theorem (Kleene's normal form theorem)

There exist a primitive recursive function $U(y)$ and a primitive recursive relation $T_n(e, x_1, \cdots, x_n, y)$ such that any computable function $f(x_1, \cdots, x_n)$ is expressed as follows: for some e,

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

$\mu y T_n(e, x_1, \cdots, x_n, y)$ is also expressed as $\mu y((1 - \chi_{T_n}(e, x_1, \cdots, x_n, y)) = 0)$, where $\chi_{T_n}(\vec{x})$ is a charactiristic function of $T_n(\vec{x})$, that is, it is 1 if $T_n(\vec{x})$ is true, and 0 otherwise.

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

Proof idea

- Our method here is different from Kleene's original proof.
- Since the internal mechanism of a Turing machine is finite, it is possible to encode it with natural number e, which can reproduce the machine.
- Giving an input sequence $(x_1, \cdots, x_n)$ to a Turing machine with code e, we record the entire transition of configurations until it stops.

- The computation may not stop in finite steps. But if it does, it is a finite sequence and can be coded as the natural number $y = (y_0, \cdots, y_k)$.
- To check whether y is a correct computation process by the Turing machine with code e on the input $(x_1, \cdots, x_n)$, it is sufficient to check whether each transition from y_i to y_{i+1} is correct or not. So, there exists a primitive recursive relation $T_n(e, x_1, \cdots, x_n, y) \Leftrightarrow$ "y is the code of the computation process of the Turing machine of code e on input $(x_1, \cdots, x_n)$."
- Furthermore, since y also contains the information about the output, there is a primitive recursive function $U(y)$ that extracts the output from y. So, $U(\mu y T_n(e, x_1, \cdots, x_n, y))$ is the output of a Turing machine with code e on input $(x_1, \cdots, x_n)$. $\square$

Fixing U and T_n constructed in the above proof, $U(\mu y T_n(e, x_1, \cdots, x_n, y))$ is called the n-ary computable (partial) function of **index** e, which is denoted as $\{e\}^n(x_1, \cdots, x_n)$ or simply $\{e\}(x_1, \cdots, x_n)$.
A function defined by a Turing machine is not defined over the whole range. Such a partial function is also written as $\{e\}(x_1, \cdots, x_n)$ for convenience.

Corollary

The family of general recursive functions and the family of recursive functions are the same.

Proof. By the theorem in Page 16, all recursive functions are general recursive. On the other hand, general recursive functions can be realized by Turing machines.
A computable function is a recursive function by Kleene's normal form theorem in Page 33.
Therefore, these function families coincide. □

For a general recursive function $f(\vec{x})$, there exists a finite equational theory E to define it. So, for each $\vec{a}$, there exists a unique c such that $E \vdash \mathtt{f}\left(\bar{\vec{a}}\right) = \bar{c}$. Assuming we know $E \vdash \mathtt{f}\left(\bar{\vec{a}}\right) = \bar{c}$ for some c, we can find out c by breadth-first search on possible proofs of E. However, it should be noted that it is undecidable whether or not a given equational theory defines a general recursive function.

Thank you for your attention!