

## General theory of Flow-Charts

by

Ken HIROSE and Makoto OYA

(Received October 2, 1972)

### § 0. Introduction

In the present paper, we are concerned with the theory of computation.

Many results in the theory have been obtained by J. McCarthy, Z. Manna, E. Engeler and others (e.g. [5], [6], [7]).

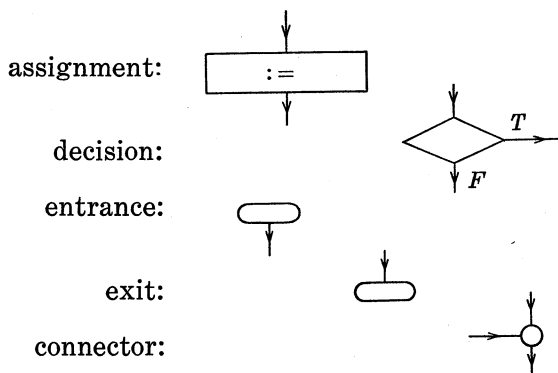
We propose a formalization of "algorithm". Thus, in §§1~4, we give a formalization of "flowchart", and "program", and prove that the class of "programs" is isomorphic to the class of relative partial recursive procedures. In §§5~8, we define some concepts and prove some results. The main result ("Normal Form Theorem") is proved in §8. Normal Form Theorem asserts that any program written by any language is reducible to an equivalent program which has at most one loop. The proof is given effectively. Furthermore, we make mention of the decidability of the equivalence between "loop-free programs".

### § 1. Flowcharts

In this section, we shall define "flowchart".

1-1 Symbols are:

- variable symbols:  $x_1, x_2, \dots, y_1, y_2, \dots$
- function symbols:  $f_1, f_2, \dots$
- predicate symbols:  $p_1, p_2, \dots$
- logical connectives:  $\vee, \neg$
- logical constants:  $T, F$
- auxiliary symbols:  $(, ), ,$
- object symbols



To each function (or predicate) symbol, there corresponds a natural number  $n$  ( $n \geq 0$ ). And in this case, it is called an  $n$ -ary function (or predicate) symbol.

0-ary function symbols are called *individual constants* (or simply *constants*). 0-ary predicate symbols are called *propositional constants*.

Function symbols and predicate symbols are called *operation symbols* and the set of operation symbols is denoted by  $Op$ .

1-2 We define *terms* as follows:

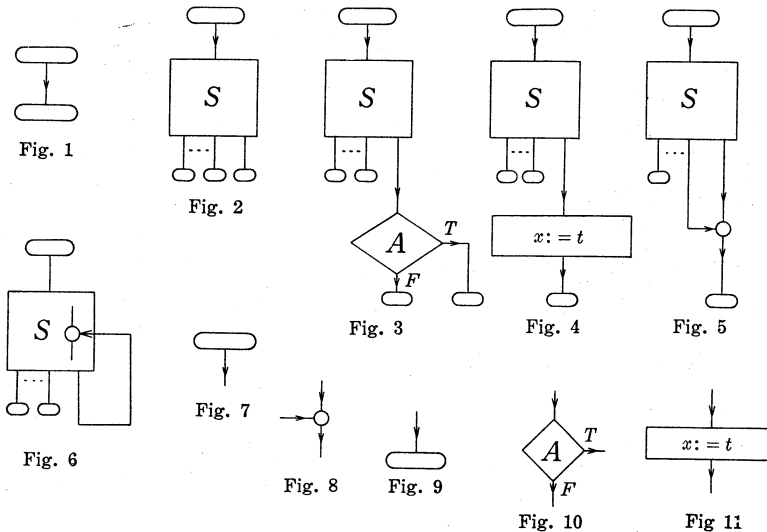
- (i) A variable symbol is a term.
- (ii) If  $f$  is an  $n$ -ary function symbol and  $t_1, \dots, t_n$  are terms ( $n \geq 0$ ), then  $f(t_1, \dots, t_n)$  is a term.
- (iii) The only terms are those given by (i) and (ii).

1-3 *Formulas* are defined by following (i)-(iv).

- (i)  $T$  and  $F$  are formulas.
- (ii) If  $p$  is an  $n$ -ary predicate symbol and  $t_1, \dots, t_n$  are terms ( $n \geq 0$ ), then  $p(t_1, \dots, t_n)$  is a formula.
- (iii) If  $A$  and  $B$  are formulas, then so are  $(A) \vee (B)$  and  $\neg(A)$ .
- (iv) The only formulas are those given by (i)-(iii).

Formulas given only by (i) and (ii) are called *atomic formulas*.

*Note:* These correspond to "open formulas" in the first order language.



1-4 Now, we shall define *flowcharts* (or *procedures*) inductively as follows:

- (i) Fig. 1 is a flowchart.
- (ii) If Fig. 2 is a flowchart, then so are Fig. 3, Fig. 4 and Fig. 5, where  $A$  is a formula,  $x$  is a variable symbol and  $t$  is a term.

(iii) If Fig. 2 is a flowchart, then so is the figure obtained by putting a connector (called the chief connector) on a line in Fig. 2 and adding an arrow from an exit to the connector. E.g. Fig. 6.

(iv) The only flowcharts are those by (i)-(iii).

Flowcharts given only by (i) and (ii) are said *loop-free flowcharts* and flowcharts given only by Fig. 1 and Fig. 4 are said *simple flowcharts*.

The figures Fig. 7, 8, 9, 10 and 11 in the flowchart are called *atoms*. If two atoms have the same figure but are located in different places, we consider they are distinct. The set of all atoms of  $S$  is denoted by  $Atom(S)$ . And the set of all exists ( $\subset Atom(S)$ ) is denoted by  $Ex(S)$ .

In the following line, to save labor, we shall use some abbreviations (For example; parenthesis, arrows,  $T, F$ , etc.).

## § 2. Interpretations

2-1 An *interpretation*  $I$  for  $OP$  consists of the following things:

(i) A nonempty set  $|I|$ .

(ii) For each  $n$ -ary function symbol  $f$ , an  $n$ -ary total function  $f_I: |I|^n \rightarrow |I|$ .

(iii) For each  $n$ -ary predicate symbol, an  $n$ -ary total predicate  $P_I: |I|^n \rightarrow \{T, F\}$ .

A 0-ary function means a fixed element of  $|I|$ , and a 0-ary predicate means value  $T$  or  $F$ .

2-2 Let  $S$  be a flowchart and  $I$  an interpretation. The pair  $(S, I)$  is called a *program* (or an *algorithm*).

2-3 If  $I$  is an interpretation for  $Op$ , the pair  $(Op, I)$  is called an *ability*.

## § 3. Semantics of programs

3-1 Let  $u$  be a term, a formula or a flowchart. By  $Var(u)$ , we mean the set of all variable symbols which occur in  $u$ . If  $Var(u) \subset \{x_1, \dots, x_n\}$ , we use  $\bar{x}_1, \dots, \bar{x}_n$  as variables through  $|I|$  corresponding to  $x_1, \dots, x_n$  respectively.

*Note:*  $x_1, \dots, x_n$  are used as syntactical variable which vary through variable symbols.

3-2 Let  $t$  be a term, and assume  $Var(t) \subset \{x_1, \dots, x_n\}$ . We define an  *$n$ -ary total function*  $I(t)$  on  $|I|$  as follows:

(i) If  $t$  is  $x_i$ , then  $I(t)(\bar{x}_1, \dots, \bar{x}_n) = x_i$ .

(ii) If  $t$  is  $f(t_1, \dots, t_m)$ , then

$$I(t)(\bar{x}_1, \dots, \bar{x}_n) = f_I(I(t_1)(\bar{x}_1, \dots, \bar{x}_n), \dots, I(t_m)(\bar{x}_1, \dots, \bar{x}_n)) .$$

3-3 Let  $A$  be a formula, and assume  $Var(A) \subset \{x_1, \dots, x_n\}$ . We define an  *$n$ -ary total predicate*  $I(A)$  on  $|I|$  as follows:

(i) If  $A$  is  $T$ , then  $I(A)(\bar{x}_1, \dots, \bar{x}_n) = T$ . If  $A$  is  $F$ , then  $I(A)$

$(\bar{x}_1, \dots, \bar{x}_n) = F$ .

(ii) If  $A$  is  $p(t_1, \dots, t_m)$ , then

$$I(A)(\bar{x}_1, \dots, \bar{x}_n) = P_I(I(t_1)(\bar{x}_1, \dots, \bar{x}_n), \dots, I(t_m)(\bar{x}_1, \dots, \bar{x}_n)) .$$

(iii) If  $A$  is  $B \vee C$ , then

$$I(A)(\bar{x}_1, \dots, \bar{x}_n) = H_{\vee}(I(B)(\bar{x}_1, \dots, \bar{x}_n), I(C)(\bar{x}_1, \dots, \bar{x}_n))$$

(where  $H_{\vee}(T, T) = H_{\vee}(T, F) = H_{\vee}(F, T) = T$  and  $H_{\vee}(F, F) = F$ ). Similarly for  $\supset B$ .

3-4 Let  $(S, I)$  be a program. Let  $x_1, \dots, x_n$  be an arbitrary set of variable symbols such that  $\text{Var}(S) \subset \{x_1, \dots, x_n\}$ . We shall define functions  $\Phi: |I|^n \rightarrow |I|^n$  and  $\varepsilon: |I|^n \rightarrow \text{Ex}(S)$  which explain the move of  $(S, I)$ .

In the first, we define  $\psi: \text{Atom}(S) \times |I|^n \rightarrow \text{Atom}(S) \times |I|^n$  as follows: for  $(\bar{x}_1, \dots, \bar{x}_n) \in I^n$  and  $a \in \text{Atom}(S)$ ;

(i) If  $a$  is as in Fig. 7 or Fig. 8,  $\psi(a, \bar{x}_1, \dots, \bar{x}_n) = (a', \bar{x}_1, \dots, \bar{x}_n)$  (where  $a'$  is the next atom along the arrow from  $a$ ).

(ii) If  $a$  is as in Fig. 9,  $\psi(a, \bar{x}_1, \dots, \bar{x}_n) = (a, \bar{x}_1, \dots, \bar{x}_n)$ .

(iii) If  $a$  is as in Fig. 10,

$$\psi(a, x_1, \dots, x_n) = \begin{cases} (a_T, \bar{x}_1, \dots, \bar{x}_n) & \text{if } I(A)(\bar{x}_1, \dots, \bar{x}_n) , \\ (a_F, \bar{x}_1, \dots, \bar{x}_n) & \text{otherwise ,} \end{cases}$$

where  $a_T$  is the next atom along the arrow from  $T$  of  $a$  and  $a_F$  is that along the arrow from  $F$  of  $a$ .

(iv) If  $a$  is as in Fig. 11 and  $x$  in Fig. 11 is  $x_i$ ,

$$\psi(a, \bar{x}_1, \dots, \bar{x}_n) = (a', \bar{x}_1, \dots, \bar{x}_{i-1}, I(t)(\bar{x}_1, \dots, \bar{x}_n), \bar{x}_{i+1}, \dots, \bar{x}_n)$$

where  $a'$  is the next atom along the arrow from  $a$ .

Next we define  $\varphi: N \times |I|^n \rightarrow \text{Atom}(S) \times |I|^n$  as follows:

$$\begin{cases} \varphi(0, \bar{x}_1, \dots, \bar{x}_n) = (a_0, \bar{x}_1, \dots, \bar{x}_n) \\ \varphi(k', \bar{x}_1, \dots, \bar{x}_n) = \psi(\varphi(k, \bar{x}_1, \dots, \bar{x}_n)) , \end{cases}$$

where  $a_0$  is an entrance in  $\text{Atom}(S)$ ,  $k'$  is the successor of  $k$  (i.e.  $k' = k + 1$ ) and  $N$  is the set of all natural numbers.

And we set  $h(\bar{x}_1, \dots, \bar{x}_n) = \mu k[\varphi(k, \bar{x}_1, \dots, \bar{x}_n)_0 \in \text{Ex}(S)]$ ; where

$$\mu k P(k) = \begin{cases} \text{the least } k \text{ such that } P(k), & \text{if } \exists k P(k) \\ \text{undefined,} & \text{otherwise ;} \end{cases}$$

and  $(V)_m = v_m$  for  $V = (v_0, v_1, \dots, v_n)$ .

Now we define  $\Phi$  and  $\varepsilon$  by

$$\psi(h(\bar{x}_1, \dots, \bar{x}_n), \bar{x}_1, \dots, \bar{x}_n) = (\varepsilon(\bar{x}_1, \dots, \bar{x}_n), \Phi(\bar{x}_1, \dots, \bar{x}_n)) .$$

Sometimes we write  $\Phi_{(S, I)}$  instead of  $\Phi$ . From this definition, we have the next lemma.

LEMMA 1. If  $|I|=N$ , then  $\varepsilon$  and  $\Phi$  is partial recursive in  $R(I)$ ; where  $R(I)$  is the set of the functions and predicates which correspond to operation symbols by  $I$ .

Note: In the case  $|I|\neq N$ , we also have Lemma 1 if there is a suitable Gödel numbering.

#### § 4. Computable functions

4-1 Let  $(S, I)$  be a program. Assume that  $e \in Ex(S)$  and  $Var(S) \subset \{x_1, \dots, x_n\}$ . Let  $\varphi$  be a  $k$ -ary partial function on  $|I|$  ( $k \leq n$ ).

We say  $(S, I)$  computes  $\varphi$  at  $e$  with  $x_i$  as the principal variable symbol and  $x_1, \dots, x_k$  as input variable symbols if for all  $(\bar{x}_1, \dots, \bar{x}_n) \in |I|^n$ ,

(i)  $\varepsilon(\bar{x}_1, \dots, \bar{x}_n) = e \iff (\bar{x}_1, \dots, \bar{x}_n) \in \text{Domain}(\varphi)$  and

(ii)  $\varepsilon(\bar{x}_1, \dots, \bar{x}_n) = e \implies (\Phi(\bar{x}_1, \dots, \bar{x}_n))_{i-1} = \varphi(\bar{x}_1, \dots, \bar{x}_k)$ .

4-2 Let  $\varphi$  be a  $k$ -ary partial function on  $D$  and  $R$  be a finite set of total functions or total predicates on  $D$ . We say  $\varphi$  is *computable* in  $R$ , if there exist a program  $(S, I)$  and a set  $\{x_1, \dots, x_n\}$  such that  $D = |I|$ ,  $R = R(I)$  and  $Var(S) \subset \{x_1, \dots, x_n\}$ , and for some  $e \in Ex(S)$  and some  $x_i$ ,  $(S, I)$  computes  $\varphi$  at  $e$  with  $x_i$  as the principal variable symbol and  $x_1, \dots, x_k$  as input variable symbols.

By Lemma 1 and the definition, we have the next theorem.

THEOREM 1. In the case  $|I|=N$ , if  $\varphi$  is computable in  $R$ , then  $\varphi$  is partial recursive in  $R$ .

And conversely:

THEOREM 2. If  $\varphi$  is partial recursive in  $R$ , then  $\varphi$  is computable in  $R \cup \{ '(successor), 0, = \}$ .

*Proof.* Each schema used to define  $\varphi$  recursively from  $R$  can be easily translated into programs.

#### § 5. Termination, correctness and equivalence

5-1 (a) We say a program  $(S, I)$  *terminates at  $e$*  ( $e \in Ex(S)$ ) for  $(\bar{x}_1, \dots, \bar{x}_n)$ , if  $\varepsilon(\bar{x}_1, \dots, \bar{x}_n) = e$ .

(b)  $(S, I)$  *terminates for*  $(\bar{x}_1, \dots, \bar{x}_n)$ , if  $(S, I)$  terminates at some  $e \in Ex(S)$ .

(c)  $(S, I)$  *terminates*, if  $(S, I)$  terminates for every  $(x_1, \dots, x_n)$ .

(d)  $S$  *terminates*, if  $(S, I)$  terminates for all interpretation  $I$ .

(a)~(d) are denoted by  $Ter(S, I, e, \bar{x}_1, \dots, \bar{x}_n)$ ,  $Ter(S, I, \bar{x}_1, \dots, \bar{x}_n)$ ,  $Ter(S, I)$  and  $Ter(S)$  respectively.

Generally, (a)~(d) are not decidable. That can be easily shown. E.g., we assume  $|I|$  contains the set of all flowcharts and elements 0, 1, and define

$$\varphi(\bar{x}, S) = \begin{cases} 0 & \text{if } Ter(S, I, \bar{x}), \\ 1 & \text{otherwise.} \end{cases}$$

(Here, we are using  $Ter(S, I, \bar{x})$  instead of  $Ter(S, I, \bar{x}, \dots, \bar{x})$  with  $n$   $\bar{x}$ 's as an abbreviation, where  $n$  is the number of  $Var(S)$ .) Then we have

**THEOREM 3.**  $\varphi$  is not computable in  $R(I)$ .

*Proof.* Assume  $\varphi$  is computable in  $R(I)$ . We set  $\psi(x) = \varphi(\bar{x}, \bar{x})$ . Clearly,  $\psi$  is computable in  $R(I)$ . Then there exists a program  $(F, I)$  which computes  $\psi$  with  $z$  as the principal variable symbol (Fig. 12). We make a flowchart  $U$  as Fig. 13.

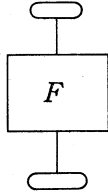


Fig. 12

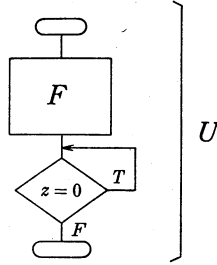


Fig. 13

Then, for any  $S$ ,

$$Ter(S, I, S) \Leftrightarrow \varphi(S, S) = 0 \Leftrightarrow \psi(S) = 0 \Leftrightarrow z = 0 \Leftrightarrow \neg Ter(U, I, S).$$

Now, put  $S = U$ ,

$$Ter(U, I, U) \Leftrightarrow \neg Ter(U, I, U).$$

We get a contradiction.

5-2 Let  $(S, I)$  be a program. Assume that  $Var(S) \subset \{x_1, \dots, x_n\}$  and  $e \in Ex(S)$ . A predicate  $P(\bar{x}_1, \dots, \bar{x}_n; \bar{y}_1, \dots, \bar{y}_n)$  is said to be *correct w.r.t.*  $(S, I)$  and  $e$ , if

$$P(\bar{x}_1, \dots, \bar{x}_n; \bar{y}_1, \dots, \bar{y}_n) \Leftrightarrow \varepsilon(x_1, \dots, x_n) = e \ \& \ \Phi(\bar{x}_1, \dots, \bar{x}_n) = (\bar{y}_1, \dots, \bar{y}_n).$$

If  $S$  has only one exit, we do not mention about  $e$ .

5-3 Let  $(S_1, I)$  and  $(S_2, I)$  be programs. Assume

$$Var(S_1) \subset \{x_1, \dots, x_k, y_1, \dots, y_n\}, \quad Var(S_2) \subset \{x_1, \dots, x_k, z_1, \dots, z_m\}$$

and  $\{y_1, \dots, y_n\} \cap \{z_1, \dots, z_m\} = \emptyset$  (empty)

(a) We say  $(S_1, I)$  and  $(S_2, I)$  are *equivalent* (denoted by  $(S_1, I) \sim (S_2, I)$ ) if there exists one to one correspondence  $M$  between  $Ex(S_1)$  and  $Ex(S_2)$ , and for every  $\bar{x}_1, \dots, \bar{x}_k, \bar{y}_1, \dots, \bar{y}_n, \bar{z}_1, \dots, \bar{z}_m$

$$\varepsilon(\bar{x}_1, \dots, \bar{x}_k, \bar{y}_1, \dots, \bar{y}_n) \xleftrightarrow{M} \varepsilon(\bar{x}_1, \dots, \bar{x}_k, \bar{z}_1, \dots, \bar{z}_m)$$

and

$$(\Phi_{(S_1, I)}(\bar{x}_1, \dots, \bar{x}_k, \bar{y}_1, \dots, \bar{y}_n))_i = (\Phi_{(S_2, I)}(\bar{x}_1, \dots, \bar{x}_k, \bar{z}_1, \dots, \bar{z}_m))_i$$

(for all  $0 \leq i < k$ ).

(b) If  $(S_1, I) \sim (S_2, I)$  for every interpretation  $I$ , we say  $S_1$  and  $S_2$  are *equivalent* (denoted by  $S_1 \sim S_2$ ).

*Note:* Equivalence does not depend on a choice of

$$\{x_1, \dots, x_k, y_1, \dots, y_n\} \quad \text{or} \quad \{x_1, \dots, x_k, z_1, \dots, z_m\}.$$

5-4 If  $P_1^i$  is correct w.r.t.  $(S_1, I)$  and  $e_i$ , and if  $P_2^i$  is correct w.r.t.  $(S_2, I)$  and  $M(e_i)$ , then  $(S_1, I) \sim (S_2, I)$  if and only if  $P_1^i \Leftrightarrow P_2^i$  for all  $i$ .

### § 6. Loop-free programs

6-1 Let  $u$  be a term, a formula or a flowchart. Assume  $Var(u) \subset \{x_1, \dots, x_n\}$ . We use  $u_{x_1 \dots x_n}(t_1, \dots, t_n)$  to designate the expression obtained from  $u$  by replacing each occurrence of  $x_i$  by  $t_i$  for  $i=1, \dots, n$ . And we use  $t$  to designate an  $n$ -tuple of terms  $(t_1, \dots, t_n)$ .

6-2 Let  $L$  be a loop-free flowchart. Assume  $Var(L) \subset \{x_1, \dots, x_n\}$ . To each  $e \in Ex(L)$ , we shall define inductively the representing set of  $e$ ,  $\{(A_1, t_1), \dots, (A_m, t_m)\}$  (where  $A_1, \dots, A_m$  are formulas and  $t_1, \dots, t_m$  are  $n$ -tuples of terms);

(i) For Fig. 14,  $\{T, (x_1, \dots, x_n)\}$ .

(ii) For Fig. 15, if the representing set of  $e$  is  $\{(A_1, t_1), \dots, (A_m, t_m)\}$  and that of  $f$  is  $\{(B_1, s_1), \dots, (B_k, s_k)\}$ , then;

(a) For Fig. 16, the representing set of  $e'$  is  $\{(A_1, t'_1), \dots, (A_m, t'_m)\}$  where  $t'_i$  is the  $n$ -tuple of terms obtained from  $t_i$  by replacing  $(t_i)_{j-1}$  by  $r_{x_1 \dots x_n}(t_i)$ ;

(b) For Fig. 17, the representing set of  $e_1$  is

$$\{(A_1 \wedge C_{x_1 \dots x_n}(t_1), t_1), \dots, (A_m \wedge C_{x_1 \dots x_n}(t_m), t_m)\}$$

and the representing set of  $e_2$  is

$$\{(A_1 \wedge \neg C_{x_1 \dots x_n}(t_1), t_1), \dots, (A_m \wedge \neg C_{x_1 \dots x_n}(t_m), t_m)\};$$

(c) For Fig. 18, the representing set of  $e'$  is

$$\{(A_1, t_1), \dots, (A_m, t_m), (B_1, s_1), \dots, (B_k, s_k)\}.$$

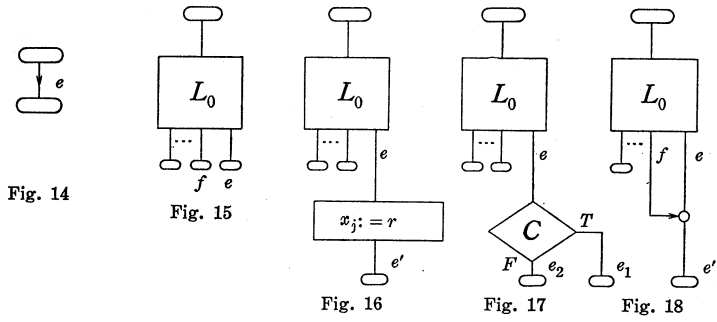
6-3 Let  $(L, I)$  be a loop-free program. Assume  $Var(L) \subset \{x_1, \dots, x_n\}$  and  $e \in Ex(L)$ .

If the representing set of  $e$  is  $\{(A_1, t_1), \dots, (A_m, t_m)\}$ , we set

$$P_{(L, I)}(\xi; \eta) \Rightarrow \begin{cases} \text{(i) } I(A_1 \vee \dots \vee A_m)(\xi) \text{ and} \\ \text{(ii) } I(A_i)(\xi) \Rightarrow \eta = I(t_i)(\xi) \end{cases}$$

(for all  $i=1, \dots, m$ );

where  $\xi = (\bar{x}_1, \dots, \bar{x}_n)$ ,  $\eta = (\bar{y}_1, \dots, \bar{y}_n)$ .



Note:  $(A_1, t_1), \dots, (A_m, t_m)$  correspond to paths from the entrance to  $e$ . Let  $p_1, \dots, p_m$  be these paths respectively. Then, (i) means  $\varepsilon(\xi) = e$  and (ii) means that the output from  $e$  is  $\eta$  for input  $\xi$  if the control passes  $p_i$ .

THEOREM 4.  $P_{(L,I)}$  is correct w.r.t.  $(L, I)$  and  $e$ .

Proof. By the similarity of the definition of  $\varepsilon$  and  $\Phi$ , and that of the representing set.

6-4 From 6-3, every loop-free flowchart with exits  $e_1, \dots, e_r$  is equivalent to a flowchart in the following form (Fig. 19). Where  $\square$  is a simple flowchart.

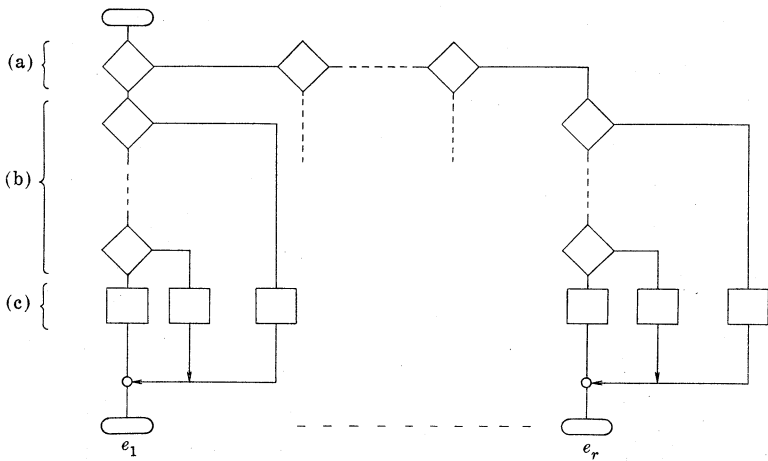


Fig. 19

Note: (a) corresponds to  $A_1 \vee \dots \vee A_m$  in 6-3, (b) to  $A_1, \dots, A_m$ , and (c)  $t_1, \dots, t_m$ .

Note: The order of  $e_1, \dots, e_r$  is immaterial.

6-5 A loop-free program seems very simple. However note that the class of loop-free programs is determined by given ability.



For example, consider a number theoretic function  $\varphi(x, y) = x \cdot y$ . If given ability has function  $\cdot$ , there exists a loop-free program which computes  $\varphi$ . But, if given ability has only function  $+$  and predicate  $=$ , there is no loop-free program which computes  $\varphi$ .

§ 7. The class FC

7-1 Let  $S$  be a flowchart. Assume  $a \in \text{Atom}(S)$ . The sub-flowchart beginning at  $a$  is the sub-flowchart of  $S$  consist of atoms which we can reach from  $a$  passing arrows one after another (independently of the contents of atoms), and of arrows between them. (In the strict sense, it is the flowchart obtained by adding an entrance to that figure and

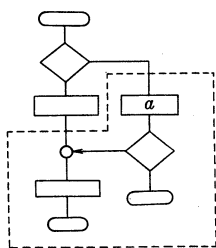


Fig. 20

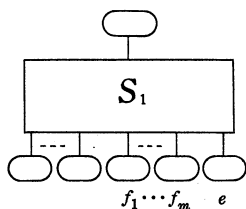


Fig. 21

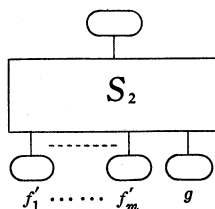


Fig. 22

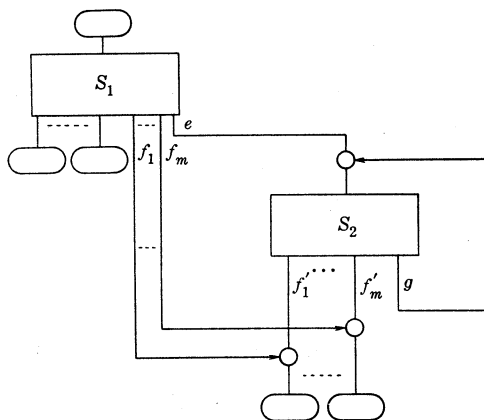



Fig. 23

taking away some useless connectors.)

For example, in Fig. 20,  is the sub-flowchart beginning at  $a$ .

7-2 We shall define inductively the class of flowcharts FC as follows:

- (i) Every loop-free flowchart belongs to FC.
- (ii) If Fig. 21 and Fig. 22 belong to FC, then Fig. 23 belongs to FC.
- (iii) The only flowcharts in FC are those given by (i) and (ii).

**THEOREM 5 (Enumeration Theorem).** *For every flowchart  $S$ , there exists a flowchart  $S^*$  such that:*

- (1)  $S^* \in FC$  and
- (2)  $S \sim S^*$ .

*Proof.* To examine the definition of flowcharts in 1-4, it is easily shown that every flowchart can be generated by applying (i) and (ii) at first, and by applying (iii) afterwards. So, the proof completes if we show that the rule (iii) in 1-4 is modified into the rule (iii) in the above definition of FC.

First, regard  $S_1$  just as  $S$  in 1-4 (Fig. 2).

In Fig. 6, let  $c$  be the chief connector and let  $a$  be the next atom to  $c$ . We assume  $a$  is not  $c$ , since the case  $a=c$  is trivial one (Consider  $S_2$  is empty or Fig. 1.). Then  $a$  is in  $S$ . Now, in Fig. 2, take the sub-flowchart beginning at  $a$  (Strictly, the flowchart of the same figure as it; i.e. its copy.). Regard this as  $S_2$ . Then Fig. 23 with a suitable connection is equivalent to Fig. 6.

$S^*$  ( $\in FC$ ) may be definable in various ways of applications of (i) and (ii). The least number of times of applications of (ii) enough to define  $S^*$  is said the *rank* of  $S^*$  (denoted by  $rank(S^*)$ ).

## § 8. Normal form

8-1 First, we prepare Lemma 2. We often use this lemma without mentioning in the proof of Theorem 6.

**LEMMA 2.** *Let  $S$  be a flowchart and  $a$  be an arbitrary decision in  $Atom(S)$ . Then  $S$  is equivalent to  $S'$  which is in the form obtained from the form of  $S$  by exchanging  $T$  and  $F$  of  $a$ .*

*Proof.* Assume that  $a$  is Fig. 24. Then we get  $S'$  from  $S$  by

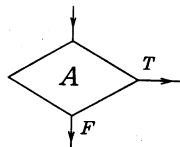


Fig. 24

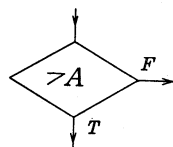


Fig. 25

replacing  $a$  by Fig. 25.

8-2 Let  $L, L_1, L_2, \dots$  be loop-free flowcharts.

We say  $S$  is in normal form if  $S$  is in the form Fig. 26.

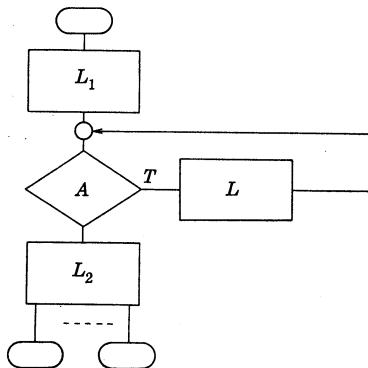


Fig. 26

**THEOREM 6 (Normal Form Theorem).** *For every flowchart  $S$ , there exists a flowchart in normal form which is equivalent to  $S$ .*

*Proof.* By Theorem 5 it is sufficient to consider  $S \in FC$ . We prove by induction on the rank of  $S$ . In the case  $rank(S)=0$ , then  $S$  is a loop-free flowchart and the theorem is trivial. So we assume that  $rank(S)=r>0$  and the theorem is true for each flowchart whose  $rank < r$ .

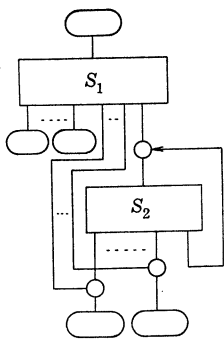


Fig. 27

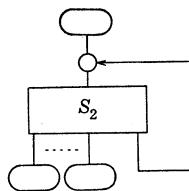


Fig. 28

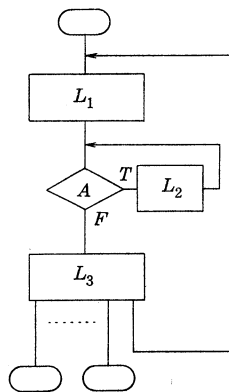


Fig. 29

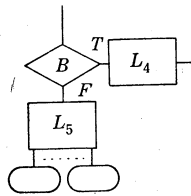


Fig. 30

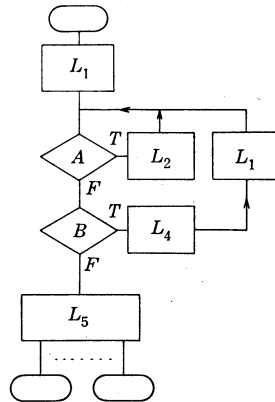


Fig. 31

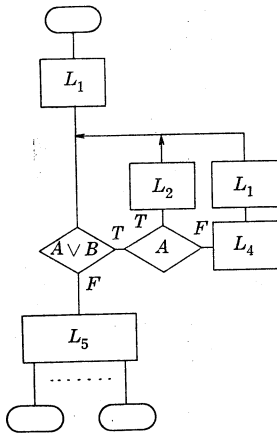


Fig. 32

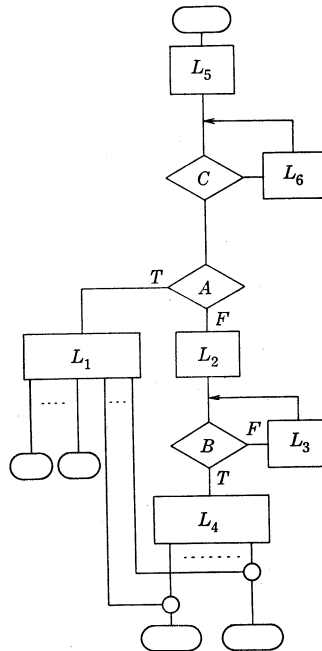


Fig. 33

By Theorem 5,  $S$  is equivalent to Fig. 27 where  $rank(S_1) < r$  and  $rank(S_2) < r$ .

In the first, we change Fig. 28 into normal form. By the assumption of our induction, Fig. 28 is equivalent to Fig. 29. By 6-4, we can consider that  $L_5$  is in the form Fig. 30. So Fig. 29 is equivalent to Fig. 31. And we have Fig. 32 using the disjunction of  $A$  and  $B$ . Fig. 32 is in normal form.

Next, by the assumption of our induction, from Fig. 27, Fig. 32 and

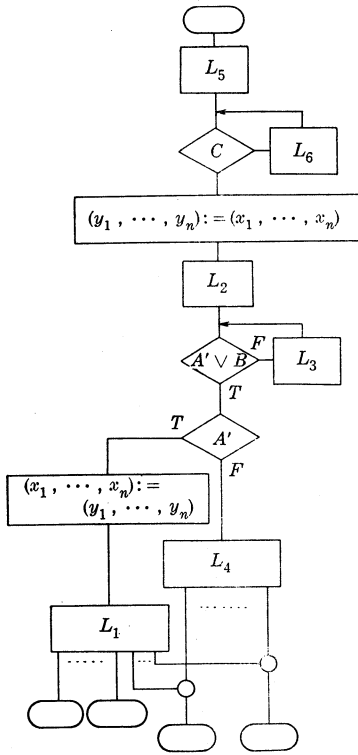


Fig. 34

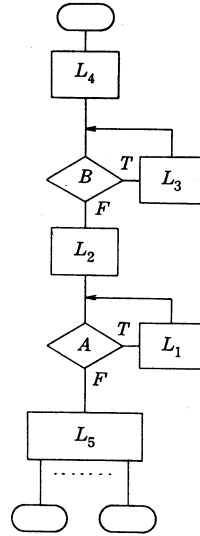


Fig. 35

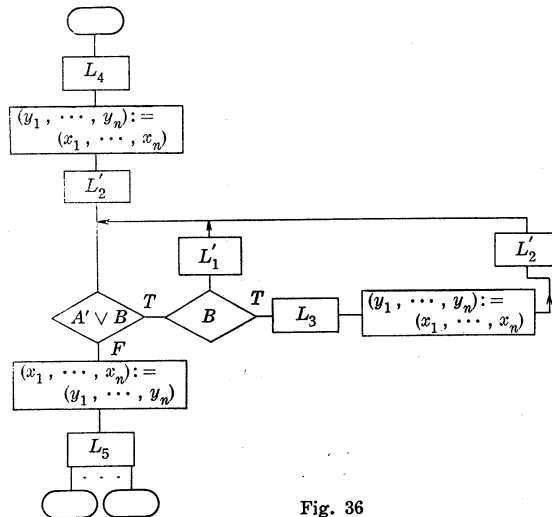


Fig. 36

6-4,  $S$  is equivalent to Fig. 33, where  $A, B, C, L_1, \dots, L_6$  are used as new names. Let  $Var(A) = \{x_1, \dots, x_n\}$  and  $y_1, \dots, y_n$  be variable symbols not occurring in Fig. 33. And let  $A'$  be  $A_{x_1 \dots x_n}(y_1, \dots, y_n)$ . Then Fig. 33 is equivalent to Fig. 34 and Fig. 34 can be written in the form Fig. 35,

where  $A, B, L_1, \dots, L_s$  are used as new names.

Let  $\text{Var}(A) \cup \text{Var}(L_1) \cup \text{Var}(L_2) = \{x_1, \dots, x_n\}$ . Let  $y_1, \dots, y_n$  be variable symbols not occurring in Fig. 35, and let  $A', L'_1, L'_2$  be  $A_{x_1 \dots x_n}(y_1, \dots, y_n)$ ,  $L_{x_1 \dots x_n}(y_1, \dots, y_n)$ ,  $L_{x_1 \dots x_n}(y_1, \dots, y_n)$  respectively. Then Fig. 35 is equivalent to Fig. 36. Fig. 36 is in normal form.

### § 9. Application of Normal Form Theorem

9-1 Let  $(F, I)$  be a program. By Theorem 5,  $F$  is equivalent to a flowchart  $S$  in the form Fig. 26.

Assume  $\text{Var}(S) \subset \{x_1, \dots, x_n\}$  and  $e \in \text{Ex}(S)$ . We define  $P_{(F, I)}$  as follows:

$$P_{(F, I)}(\xi; \eta) \Leftrightarrow (\exists \eta_0, \eta_1, \dots, \eta_k) [P_{(L_1, I)}(\xi; \eta_0) \ \& \ (\forall i)_{i < k} \{I(A)(\eta_i) \ \& \ P_{(L, I)}(\eta_i; \eta_{i+1})\} \ \& \ \supset I(A)(\eta_k) \ \& \ P_{(L_2, I)}(\eta_k; \eta)] \quad (\text{cf. 6-3}).$$

THEOREM 7.  $P_{(F, I)}$  is correct w.r.t.  $(F, I)$  and  $e$ .

*Proof.* By Theorems 4, 6 and the definition of correctness.

*Note:* The above gives another proof of Theorem 1.

*Note:* Similarly we can easily get a predicate correct w.r.t. a program by Theorem 5.

*Note:* By the definition of  $P_{(F, I)}$  and Theorem 2, we obtain the following proposition: For every recursively enumerable predicate  $R(a_1, \dots, a_n)$ , there exists a diophantine predicate  $D(k, a_1, \dots, a_n, y)$  such that

$$R(a_1, \dots, a_n) \Leftrightarrow (\exists y)(\forall k)_{k < y} D(k, a_1, \dots, a_n, y).$$

This is well known result by M. Davis [1].

### § 10. On the equivalence of loop-free programs

10-1 Let  $A$  be a formula and  $I$  be an interpretation. Assume  $\text{Var}(A) \subset \{x_1, \dots, x_n\}$ . We define:

$A$  is valid in  $I$  if  $I(A)(\bar{x}_1, \dots, \bar{x}_n)$  for all  $\bar{x}_1, \dots, \bar{x}_n \in |I|$ .

$A$  is valid if  $A$  is valid in every interpretation.

An interpretation by which a predicate symbol = corresponds to “=” on  $|I|$  is said a *structure*.

$A$  is *s-valid* if  $A$  is valid in every structure. Flowcharts  $S_1$  and  $S_2$  are *s-equivalent* (denoted by  $S_1 \sim_s S_2$ ) if  $(S_1, I) \sim (S_2, I)$  for every structure  $I$ .

10-2 If  $L$  is a loop-free flowchart and  $I$  is a structure, we can rewrite  $P_{(L, I)}$  as follows:

$$(*) \quad P_{(L, I)}(\xi; \eta) \Leftrightarrow I[(A_1 \vee \dots \vee A_m) \wedge \bigwedge_{i=1}^m (A_i \Rightarrow Y = t_i)](\xi; \eta),$$

where  $Y = (y_1, \dots, y_n)$  (cf. 6-3).

So we have,

**THEOREM 8.** *If  $I$  is a structure, for loop-free programs  $(L_1, I)$  and  $(L_2, I)$ , it can be constructed a formula  $A$  satisfying*

$$(L_1, I) \sim (L_2, I) \Leftrightarrow [A \text{ is valid in } I] .$$

*Proof.* By 5-4 and (\*), we can easily construct such  $A$ .

**COROLLARY 8-1.** *For loop-free flowcharts  $L_1$  and  $L_2$ , it can be constructed a formula  $A$  satisfying*

$$L_1 \underset{s}{\sim} L_2 \Leftrightarrow [A \text{ is } s\text{-valid}] .$$

**COROLLARY 8-2.** *For loop-free flowcharts  $L_1$  and  $L_2$ ,*

(1)  $L_1 \underset{s}{\sim} L_2$  *is decidable.*

(2)  $L_1 \sim L_2$  *is decidable.*

*Proof.* (1) “ $A$  is  $s$ -valid” is decidable (see [2]).

(2) Immediately from (1).

10-3 We introduce concept of “axiom” to extend these results.

Let  $\Gamma$  be a set of formulas called *axioms*. A structure  $I$  is said  $\Gamma$ -*model* if every axiom of  $\Gamma$  is valid in  $I$ . A formula  $B$  is  $\Gamma$ -*valid* if  $B$  is valid in every  $\Gamma$ -model.

Flowcharts  $S_1$  and  $S_2$  are  $\Gamma$ -*equivalent* (denoted by  $S_1 \underset{\Gamma}{\sim} S_2$ ) if  $(S_1, I) \sim (S_2, I)$  for every  $\Gamma$ -model  $I$ .

Now, we have

**COROLLARY 8-3.** *For loop-free flowcharts  $L_1$  and  $L_2$ ,*

$$L_1 \underset{\Gamma}{\sim} L_2 \Leftrightarrow [A \text{ is } \Gamma\text{-valid}] \Leftrightarrow [A \text{ is provable in } \Gamma] .$$

*Proof.* From Theorem 7 and Gödel’s completeness theorem [2].

Hence, we can get some results from model theory or proof theory, for example, from Herbrand’s theorem [2].

10-4 Conversely to Theorem 8,

**THEOREM 9.** *For a formula  $A$ , there exist loop-free flowcharts  $L_1$  and  $L_2$  satisfying*

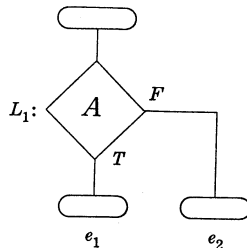


Fig. 37

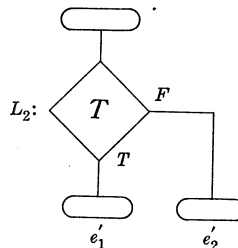


Fig. 38

$$(A \text{ is valid in } I) \Leftrightarrow (L_1, I) \sim (L_2, I) .$$

*Proof.* Set  $L_1$  and  $L_2$  as follows:

10-5 From Theorems 8 and 9, we can say the validity of open formulas is equivalent to the equivalence between loop-free programs.

Following applications of this result are about undecidability.

**THEOREM 10.** *For some  $\Gamma$ , it is undecidable whether  $L_1 \sim_{\Gamma} L_2$  or not for given loop-free flowcharts  $L_1$  and  $L_2$ .*

*Proof.* By Theorem 9, for any formula  $A$ , there exist loop-free flowcharts  $L_1$  and  $L_2$  satisfying

$$[A \text{ is provable in } \Gamma] \Leftrightarrow L_1 \sim_{\Gamma} L_2 .$$

However, the left half is undecidable [2].

**THEOREM 11.** *Assume an ability  $(Op, I)$  as follows:  $Op = \{0, 1, +, \cdot, =\}$ ,  $|I| = \mathbb{Z}$  (the set of all integers), and interpretation  $I$  gives natural meaning to each element of  $Op$ . Then, there is no algorithm that determines whether  $(L_1, I) \sim (L_2, I)$  or not for given loop-free programs  $(L_1, I)$  and  $(L_2, I)$ .*

*Proof.* By negative solution of Hilbert's 10th problem [4], there is a Diophantine equation  $D=0$  having no algorithm that determines whether any solution  $D=0$  exists or not for given coefficients of  $D$ . Consider  $D \neq 0$  as  $A$  in Theorem 9. Then,

$$(D \neq 0 \text{ is identically true}) \Leftrightarrow (L_1, I) \sim (L_2, I)$$

(where  $L_1$  and  $L_2$  are loop-free flowcharts determined by coefficients of  $D$ ). Hence,

$$(D=0 \text{ has some solution}) \Leftrightarrow \text{not } [(L_1, I) \sim (L_2, I)] .$$

So, if whether  $(L_1, I) \sim (L_2, I)$  or not is decidable, then whether  $D=0$  has any solution or not is decidable. That contradicts to the assumption of  $D$ .

10-6 The operation  $\cdot$  (product) in Theorem 11 is very important. If given ability has only 0, + and =, then equivalence between loop-free programs is decidable. In fact, it is well known that it is decidable whether given formula that has only + and = as operation symbols is valid or not.

*Acknowledgement.* The authors should like to express their cordial thanks to Professor T. Simauti for his useful advices and discussions.

### References

- [1] DAVIS, M.; *Computability and Unsolvability*. McGraw-Hill.
- [2] SHOENFIELD, J.; *Mathematical Logic*. Addison-Wesley Publ. Co.
- [3] KLEENE, S. C.; *Introduction to Metamathematics*. Van Nostrand.



- [ 4 ] MATIJASEVIC, Ju. V.; Enumerable sets are Diophantine. *Soviet Math. Dokl.*, **11** (1970).
- [ 5 ] MCCARTHY, J.; Towards a mathematical science of computation. *Proc. of IFIP* **62** (1963).
- [ 6 ] MANNA, Z.; Properties of Programs and the First-Order Predicate Calculus. *J. of A. C. M.*, 1969.
- [ 7 ] ENGELER, E.; Structure and Meaning of Elementary Programs. *Symposium on Semantics of Algorithmic Languages*, Springer-Verlag, 1970.
- [ 8 ] HIROSE, K. and OYA, M.; Some Results in General Theory of Flow-Charts. *Proceedings of 1st USA-Japan Computer Conference*, 1972.