A generalization of algorithms and one of its applications (I)

By Nobuo ZAMA

(Received November 15, 1966)

Introduction

After Turing had invented his famous Turing machine, some mathematicians have given definitions of various sorts of algorithms, i.e. effective procedures which transform a word to another. The equivalency between these algorithms are shown in application of abstract flow-charts given by Kaluznin.

On the other hand, von Neumann gave two sorts of models of the self-reproducing machine, which have powers to construct the machine itself just like those given at first.

In this paper, the author will show that the investigation of self-reproducing machines will become more easier by the adoption of generalised algorithms which transform a finite set to another.

The concept of operaters is invented to confine the information about a flow-chart to a finite set. We can prove that this concept enables us to perform an algorithm within a finite set itself.

1. The definition of algorithms

Let $\boldsymbol{M}$ be a set of arbitrary subjects, $\boldsymbol{P}$ be a set of predicates $P(X)$ whose arguments range over $\boldsymbol{M}$, and $\boldsymbol{F}$ be a set of functions $F(X)$ whose variables range over $\boldsymbol{M}$. A mathematical structure $\mathscr{F}$ is the union of such $\boldsymbol{M}$, $\boldsymbol{F}$ and $\boldsymbol{P}$ as above. $\mathscr{F}$ is written $\langle \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M} \rangle$.

A $FC(\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M})$ is, roughly speaking, a flow-chart constructed from elements of $\boldsymbol{F}$ and $\boldsymbol{P}$. The details of $FC(\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M})$ are given in [3]. The set of functions and predicates defined by $FC(\mathscr{F})$ is written $\langle\langle \mathscr{F} \rangle\rangle$ or $\langle\langle \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M} \rangle\rangle$. An algorithm over $\mathscr{F}$ is a function defined by $FC(\mathscr{F})$, and the set of algorithms over $\mathscr{F}$ is written $Alg(\mathscr{F})$.

The theorem 1, which plays the fundamental role in this paper, is proved in [3]. So we shall apply it without any proof here.

Theorem 1. *Let* $\mathscr{F} = \langle \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M} \rangle$ *and* $\mathscr{G} = \langle \boldsymbol{G}, \boldsymbol{Q}, \boldsymbol{M} \rangle$ *be two mathematical structures. Then*

$$\langle\langle \mathscr{F} \rangle\rangle \supseteq \langle\langle \mathscr{G} \rangle\rangle \Leftrightarrow \boldsymbol{G} \subseteq \langle\langle \mathscr{F} \rangle\rangle \quad \text{and} \quad \boldsymbol{F} \subseteq F\langle\langle \mathscr{F} \rangle\rangle$$

2. The definition of s-algorithms

2.1. The definition of finite sets

Let us consider finite numbers of arbitrary subjects $a, b, c, \cdots$. Let **A** be the set of these subjects, and we call it an alphabet.

A word on an alphabet **A** is a finite sequence of elements in **A**, admitting the repetition of the same symbols, and we write them $a_1a_2\cdots$. Every subject in a word W is called the component of W. If a word W_2 is a consecutive part of W_1, we call W_2 is a subword of W_1. Let **W(A)** be the set of all words on **A**.

If we define the predicate $\sim$ in **W(A)** by the rule that

$W_1 \sim W_2 \Leftrightarrow$ *all components* W_1 *are contained in* W_2 *and all components of* W_2 *are contained in* W_1,

$\sim$ is an equivalence-relation in **W(A)**. Therefore, we can devide **W(A)** into equivalence-classes by $\sim$. **W(A)**/$\sim$, the set of so-gained quivalence-classes, is written **M(A)**. Every element of **M(A)** is called a finite set on **A**, and **M(A)** is called the family of finite sets of **A**. When a class in $\mathbf{M(A)}=\mathbf{W(A)}/\sim$ contains a word W, we call W is the representative of the class. When S is contained in **M(A)**, we write $S \in \mathbf{M(A)}$.

Let $S_1, S_2 \in \mathbf{M(A)}$. If every representative V of a class S_2 in **W(A)**/$\sim$ is a subword of a suitable representative U of S_1, S_2 is called a subset of S_1, and write $S_1 \subset S_2$.

When $V \in \mathbf{M(A)}$, every component of the representative of V is called an element of V. If a is an element of V, we write $a \in V$. The finite set $V \in \mathbf{M(A)}$, whose elements are $a_1, \cdots, a_n$, is written $\{a_1, a_2, \cdots\}$.

2.2. The definition of s-algorithm over **A**

When X is an arbitrary set in **M(A)**, 《a particular finite set P is contained in X》 is a predicate whose argument is X. We write it $P?(X)$.

When X is the same as above, the transformation 《Construct a new set in **M(A)** by the change of a particular subset P of X to a particular finite set $Q \in \mathbf{M(A)}$, and if X doesn't contain P, do nothing》 is a function whose variable is X. We write it $[P \rightarrow Q](X)$.

Let $\boldsymbol{F}$ be the set of some functions with the types $[P \rightarrow Q](X)$, where $P, Q \in \mathbf{M(A)}$. Similarly, let $\boldsymbol{P}$ be the set of some predicates with the type $P?(X)$ where $P \in \mathbf{M(A)}$. Then we have a mathematical structure $\mathscr{F} = \langle \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M}(\mathbf{A}) \rangle$. We call every element in $Alg(\mathscr{F})$ is a *s*-algorithm over $\mathscr{F}$.

We write $F(R)=S$ or $R \xrightarrow{F} S$ when a *s*-algorithm F transforms R to S.

When $M \in \mathbf{M(A)}$, we shall understand that $\bar{M}$ is a representative of M as a class in **W(A)**/$\sim$. Let $M, N \in \mathbf{M(A)}$, the notation $M+N$ means the set whose representative is the word connected $\bar{M}$ to $\bar{N}$. Similarly, $M-N$ means the set whose representative is the word rejected $\bar{N}$ from

$\bar{M}$ (where $M \subset N$).

Let $a, b \in \mathbf{A}$. We understand $a?(X)$ means the predicate 《$X \in \mathbf{M(A)}$ contains a》, $[a \rightarrow b](X)$ means the function 《Transform a in X to b; but do nothing, if X doesn't contain a.》. $[\rightarrow b](X)$ means 《Add b to X.》, $[a \rightarrow](X)$ means 《Erase a in X》.

Theorem 2. *Let $Alg(\mathscr{F})$ be s-algorithms over $\mathscr{F} = \langle \boldsymbol{F}, \boldsymbol{P}, \mathbf{M(A)} \rangle$. Then, we can choose a mathematical structure $\mathscr{G} = \langle \boldsymbol{G}, \boldsymbol{Q}, \mathbf{M(B)} \rangle$, with the next condition, where $\mathbf{B}$ is an extention of $\mathbf{A}$,*[*)]

1) *$\boldsymbol{G}$ is the set of some functions with the type $[a \rightarrow b](X)$ where $a, b \in \mathbf{B}$,*
2) *$\boldsymbol{Q}$ is the set of some predicates with the type $a?(X)$, where $a \in \mathbf{B}$;*
3) *$Alg(\mathscr{G}) \subset Alg(\mathscr{G})$.*

PROOF: It is sufficient that $\boldsymbol{F} \subset$《$\mathscr{G}$》 and $\boldsymbol{P} \subset$《$\mathscr{G}$》. Therefore, we must show that $P?(X)$ and $[P \rightarrow Q](X)$ are written in the form of $FC(\mathscr{G})$.

(a) $P?(X)$. Let $P = \{a_1, \cdots, a_n\}$. If these a_i's are all different from each other, we have only to make certain whether X contains a_i $(i = 1, \cdots, n)$, one after another. If X contains some numbers of the same a_i, these a_i's must be transformed to t after we make certain each $a_i (t \notin \mathbf{A})$.

After the operation, all of t's are transformed to a_i's.

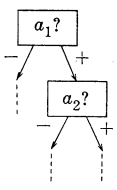

Fig. 1.[**)]

(b) $[P \rightarrow Q](X)$. Let $P = \{p_1, \cdots, p_n\}$, $Q = \{q_1, \cdots, q_n\}$.

If $m = n$, $[P \rightarrow Q](X)$ can be done by the repetitions of $[p_1 \rightarrow q_1](X)$, $[p_2 \rightarrow q_2](X)$, $\cdots$, $[p_n \rightarrow q_n](X)$.

If $m > n$ or $m < n$, we must add the function of the type $[p_i \rightarrow](X)$ or $[\rightarrow q_j](X)$. q.e.d.

3. Operaters and machines

3.1. Operaters and their operations

Every s-algorithm must be defined by a FC. We shall condiser whether the informations about every FC can be contained in a finite set. This plan can be realised by introducing the concept of operaters.

3.1.1. The elements of $\mathbf{P}$ must be classified to two distinct classes in the following:

the one is the class of simple element: $a, b, c, \cdots$,

the other is the class of operaters: $\alpha(\ ), \beta(\ ), \gamma(\ ), \cdots$.

3.1.2. The operation of each operaters.

To every operater $\alpha(\ )$ in $\mathbf{P}$, some sets consisting of at most two

Note: *) An alphabet B is called an extention of A when B contains A.

**) In this Fig., + means 《Yes》, − means 《No》.

Note: The symbol ( ) is considered as one symbol involved empty parenthesses. If no confusion is expected, we shall omitt the parentheses.

elements in **P** are corresponded. Let them be $\{a_1, b_1\}, \cdots, \{a_k, b_k\}, c_1, \cdots, c_p$. These elements a_i, b_i, c_i's can be either simple elements or operaters. We call them the operands of $\alpha(\ )$. Then $\alpha(\ )$ operates to one of them at a time, for example $\{a_1, b_1\}$. As the result of operation of $\alpha(\ )$, we have $\alpha(a_1, b_1)_1, \cdots, \alpha(a_1, b_1)_k$. After this, $\alpha(\ )$ recovers the form before the operation, and the operand is vanished. If $\alpha(\ )$ operates an element c_1, we have $\alpha(c_1)_1, \cdots, (c_1)_p$, and then $\alpha(\ )$ also recovers similarly.

The set $\{\alpha, \alpha(a_1, b_1)_1, \cdots, \alpha(a_1, b_1)_k\}$ is written $\Pr(\alpha(\ ), a_1, b_1)$, and we may use the representation

$$\alpha(a_1, b_1) \rightarrow \{\alpha, \alpha(a_1, b_1)_1, \cdots, \alpha(a_1, b_1)_k\} .$$

or

$$\alpha(c_1) \rightarrow \{\alpha, \alpha(c_1)_1, \cdots, \alpha(c_1)_1\}$$

3.1.3. Suppressors of operaters

Some indicated pairs of an operater $\alpha(\ )$ and an element a disturb the operation of $\alpha(\ )$, that is $\alpha(\ )$ is considered as a simple element in this case. We call a is the suppressor of $\alpha(\ )$, and write $a = S_p(\alpha(\ ))$.

3.2. Operations of M.

3.2.1. A step of operations of M.

Let M be the finite set of elements in **P**, and $\alpha_1(\ ), \cdots, \alpha_k(\ )$ be the operaters in M. When $\alpha_1(\ )$ has its operand (a_1, b_1) or c_1 in M, the set $\{\alpha_1(\ ), a_1, b_1\}$ or $\{\alpha_1(\ ), c_1\}$ is called a suitable pair. If M contains the set of suitable pair $P_1, \cdots, P_n$, the operation of all these operaters for their operands are carried out simultaneously. Then M is transformed to

$$M_1 = (M + \Pr(P_1) + \cdots + \Pr(P_2)) - (P_1 + \cdots + P_n).$$

We shall use the notation $M \rightarrow M_1$ in such a case. Such a procedure is called a step, and M_1 is the result of the step.

3.2.2. The operation by M.

The repetition of the steps, starting from a set M, is called the operation by M. If there is no operand for any operaters after some steps, we call that the operation ends in the step and that the so-gained set in the last step is the result of M. The result of M is written $\mathrm{Res}(M)$. If $\mathrm{Res}(M) = P$, we write $M \rightarrow P$.

3.2.3. The consistent operation.

If there are more than two ways to make suitable pairs in a step of an operation in M, the step can be carried out in either way. So the result can not be decided uniquely in such a case.

Particularly, a step is called consistent, when the result of the step is

uniquely determined. When all steps of an operation in M are consistent, we call the operation is consistent.

Lemma. Every consistent procedure is an Alg ($\mathscr{F}$), when the mathematical structure $\mathscr{F}$ is taken suitably.

PROOF. We suppose that **Q** is the alphabet which contains the elements of **P** and the letters $S_1, \cdots, S_s$ representing the suitable pairs. Let $M=\mathbf{M}(\mathbf{Q})$.

Let F be the set of the next three sorts of functions.

1) $$[S_i \rightarrow P_r(S_i)](X) \qquad (i=1, \cdots, s)$$

where $P_r(S_i)$ indicates the result of the suitable pair S_i.

2) $$[\{\alpha(\ ), P\} \rightarrow S_i](X) \qquad (i=1, \cdots, s)$$

where $\{\alpha(\ ), P\}$ indicates a certain suitable pair. Such functions are made for all suitable pairs.

3) $$[S_i \rightarrow \{\alpha(\ ), P\}](X) \qquad (i=1, \cdots, s)$$

where $\{\alpha(\ ), P\}$ is the suitable pair corresponded to the letter S_i.

Let P be the set of such predicates $\{\alpha_1(\ ), P_1\}?\ (X)$ and $S_i?\ (X)$. If $\mathscr{F}=\langle \boldsymbol{F}, \boldsymbol{P}, \boldsymbol{M} \rangle$, every consistent operation is written in the form of Alg ($\mathscr{F}$). For example one step of the operation is written in the form of FC with the Fig. 2.

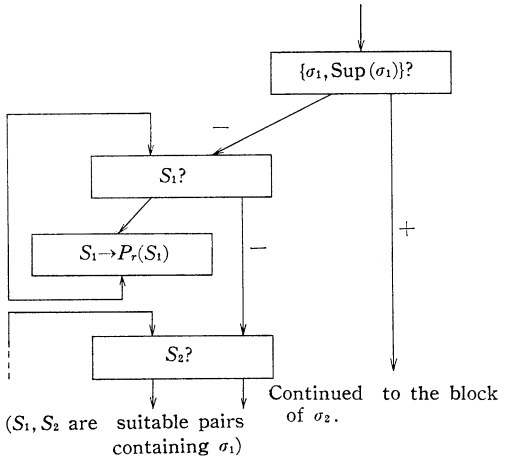

Fig. 2.

When there exists a suppressor for an operater, the blocks $\{\alpha_1, S_p(\alpha_1)\}?$ must be prepared. q.e.d.

3.3. Machine.

We assume that **P** is an alphabet involving operaters. Let Π be

an arbitrary set of operaters in **P**. When N is a set of simple elements, and the operation of these operaters are defined, the operation of $\Pi+N$ is called the operation of machine Π with the input N. If $\mathrm{Res}(\Pi+N)=P$, we may write $N, \Pi \rightarrow P$. And we write $\Pi(N)=Q$, where Q is the set of simple operaters in **P**.

The operation of M described in the preceeding section is the operation of Π_M with the input S_M, where Π_M is the set of the operaters in M and S_M is the set of the simple elements in M.

Theorem 3. *We assume that F is a s-algorithm over* **A**. *Then we can take such a Π of operaters that*

$$\Pi(X)=F(X) ,$$

if we define suitably the operation of operaters in Π.

PROOF. When $\mathscr{F}=\langle \boldsymbol{F}, \boldsymbol{P}, \mathbf{M}(\mathbf{A})\rangle$, let $F \in Alg(\mathscr{F})$. According to theorem 2, we can assume that $\boldsymbol{F}$ contains only functions with types $[a \rightarrow b](X)$, and that $\boldsymbol{P}$ contains only predicates $a?(X)$ without loss of generality $(a, b \in \mathbf{A})$.

When F is defined by a $FC(\mathscr{F})$, let $b_1, \cdots, b_k$ be all its blocks, b_1 the input, and b_k the output. Corresponding to all of their mathematical blocks, we prepare the operater $\sigma_{m_1}, \cdots, \sigma_{m_p}$. Similarly, to all logical blocks, we prepare $\sigma_1^+, \sigma_1^-, \cdots, \sigma_q^+, \sigma_q^-$. We take $\Pi=\{\sigma_1, \phi\}$, and define their operations as follows.

1) When b_m is mathematical, we assume that b_m contains a function $[a \rightarrow b](X)$.

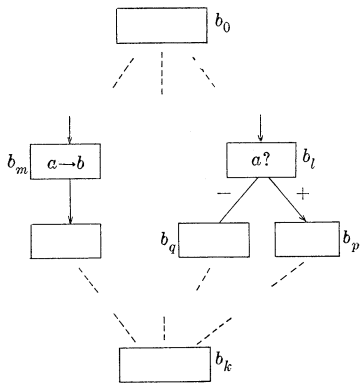

Fig. 3.

$$\sigma_m(a) \rightarrow \{\sigma_m, b, \mathrm{s}\}$$
$$\phi(s, \sigma_m) \rightarrow \{\phi, (\sigma_n)\} .$$

where (σ_n) is the operaters corresponded to the next mathematical block of b_m. So if it is logical, (σ_n) must be replaced to σ_n^+, σ_n^-.

In this case σ_m operates as follows:

$$\{\phi, \sigma_m, a, M^*\}$$
$$\rightarrow \{\phi, \sigma_m, p, s, M^*\}$$
$$\rightarrow \{\phi, (\sigma_n), b, M^*\}.$$

2) When b_l is logical, b_l contains a function a? (X).

The operations of operaters must be defined as follows:

$$
\begin{aligned}
a = S_p(\sigma_l(\ \ ))&\\
\sigma_l^+(a) &\longrightarrow \{\sigma_l^+, s, a\}\\
\phi(\sigma_l^+, s) &\longrightarrow \{\phi, (\sigma_p), t\}\\
\phi(\sigma_l^-, t) &\longrightarrow \{\phi, \sigma_0\}\\
\sigma_l^-(\sigma_l^+) &\longrightarrow \{\sigma_l^-, s\}\\
\phi(\sigma_l^-, s) &\longrightarrow \{\phi, (\sigma_q)\}
\end{aligned}
$$

where σ_0 is an element which has not any operand, and (σ_q) means just like the above. The operaters give us the next operation:

$$
\begin{aligned}
\phi, \sigma_l^+, \sigma_l^-, a, M^* &\longrightarrow \phi, \sigma_l^+, \sigma_l^-, s, a, M^* \longrightarrow \phi, (\sigma_p), \sigma_l^-, t, a, M^*\\
&\longrightarrow \phi, (\sigma_p), a, M^*, \sigma_0
\end{aligned}
$$

where a is contained,

$$\phi, \sigma_l^+, \sigma_l^-, M \longrightarrow \phi, \sigma_l^-, s, M \longrightarrow \phi, \sigma_p, M$$

where a is not contained.

We can define such an operation that gives no operands for all operaters at last. Then, the operation stops, and we have the desired result. q.e.d.

Corollary 1. *Every machine Ξ is written in the form of $\{\phi, \sigma_1\}$ with any change of its working function, where ϕ makes all operaters besides σ_1. That is, there is such a machine $\{\phi, \sigma_1\}$ that $\{\phi, \sigma_1\}(X) = Y$ whenever $\Pi(X) = Y$.*

PROOF. Let $\Pi(X) = Y$. We can consider the operation as a s-algorithm which transforms X to Y. According to the theorem 3, such a machine $\Xi(X) = \Pi(X)$, where Ξ is the machine $\{\phi, \sigma_1\}$. q.e.d.

Definition. *In the preceding corollary, ϕ is called the fundamental operater of Ξ.*

Corollary 2. *Let $\Xi = \{\sigma, \phi\}$ be any such a machine as in Corollary 1. If we can find such x that $\phi(x) \longrightarrow \{\phi, \sigma, \sigma\}$, and x is not an operand to other operaters appearing in the operation,*

$$x, \Xi \longrightarrow \Xi + \Xi\ .$$

where ϕ is the fundamental operater.

Corollary 3. *Let us assume the same assumption as above. If Y is the input of Ξ, and Y is not the operand of ϕ, then*

$$(x + Y) \longrightarrow \Xi + \Xi(Y)\ .$$

PROOF. ϕ doesn't operate in the first step of the operation. So, if x is an operand of ϕ, $\phi(x)$ is gained at first, and then the operation of Ξ to Y is carried out. q.e.d.

Definition. If Π is a machine and Z is its input, Π is called a self reproducing machine when

$$Z, \Pi \rightarrow \Xi \quad \text{and} \quad \Xi \supset Z .$$

3.4. The machine of polygons.

Let $\mathbf{E}=\{t_1, \cdots, t_n\}$ be a set of arbitrary subjects and we call every element of $\mathbf{E}$ is a vertax. Under the condition that $\rightarrow$ is not an element of $\mathbf{E}$, $t_i \rightarrow t_j (i, j=1, \cdots, n: i \neq j)$ is called a side. Any finite set of a side is called a polygon.

The operation, which we construct a polygon $\{t_i \rightarrow t_j, \cdots, t_k \rightarrow t_l\}$ starting from $\mathbf{E}$, is a s-algorithm. So we can write it in the form of such a machine Π as

$$\Pi(\mathbf{E})=\{t_i \rightarrow t_j, \cdots, t_k \rightarrow t_l\}$$

Then, by the corollary in the preceding section we have such a machine that

$$(x+\mathbf{E}), \Xi \rightarrow \Pi+\{t_i - t_j, \cdots, t_k \rightarrow t_l\} .$$

where x is a particular set of elements which $x, \Pi \rightarrow \Pi+\Pi$.

References

(1) A. W. Burks: Programming and the theory of automata. Computer programming and formal systems, Edited by P. Eraffort and D. Hirschberg. pp. 100-117. 1963.

(2) S. Beer: Towards the cybernetic factory. International tracts in computer science and thechnology and their application, Vol. 9. Principles of selforganization, pp. 25-80. 1962.

(3) N. Zama: On Models of Algorithms and Flow-charts. Commentatiorum mathematicorum universitatis Sancti Pauli, Vol. 14. pp. 123-134. 1966.