On the Logical Schemes of Algorithms

SHIGERU IGARASHI*

We have Yu. I. Yanov's study based on the concept of logical schemes of algorithms [1] as a pioneering work in the theoretical approach to computer programs. An improvement of Yanov's formulation will be intended in section 1, and the logical schemes will be redefined so that the logical schemes in Yonov's sense will be contained as a special case of them. In section 2, the equivalence problem of these schemes will be shown reducible to that of finite automata, and then an algorithm to discriminate the equivalence by a computer-oriented procedure will be given.

1. *Logical Schemes of Algorithms with the Implicit and the Explicit Operators*

Let us base the following analysis on the matricial scheme for the convenience of description. Firstly we consider a finite set of *points* $x_1, \cdots, x_n$, which will simplify the treatment of the multiple occurrence of an operator in the same scheme. (See [1] for the terminology.)

With each point x_i is associated one of the (*implicit*) *operator symbols* $A_1, A_2, \cdots$. Symbols or expressions associated with the point x_i shall be marked by the subscription x_i hereafter, so that the operator corresponding to x_i shall be denoted by A_{x_i}. A_{x_i} and A_{x_j} may be the same. Secondly we consider the *external* (*logical*) *variables* $p_1, \cdots, p_r$ and the *internal* (*logical*) *variables* $q_1, \cdots, q_s$.

The former are treated as the information source so that they are to represent the indefinite changes of the information that will occur in computer programs such as of the values of the logical, or Boolean, variables and expressions, containing those in ALGOL's sense induced by arithmetic operations as well as by input-outputs. The latter are treated as the logical variables or expressions which can be memorized and transformed by the program so that it will represent Boolean variables and expressions and switch variables in ALGOL's sense.

Besides the implicit operator symbols, to each point is attached an *explicit operator* F_{x_i} which can be written as

$$(q_1', \cdots, q_s') = F_{x_i}(p_1, \cdots, p_r, q_1, \cdots q_s). \qquad (1)$$

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 3, No. 2 (1962), pp. 66-72.

* Faculty of Engineering, University of Tokyo.

Namely we assume that by the execution of F_{x_i} the values of the internal variables change depending on both the external and the internal variables. It must be noted that there remains the possibiliy of indefinite changes of the values of the internal variables because of the indefiniteness of the values of the exteanal variables, though the mapping F_{x_i} is definite.

The *matrix* representing the connective property represents the connection among the points so that it shall be denoted by $(\alpha_{x_ix_j})$, where $\alpha_{x_ix_j}$ is a logical function depending at most on the internal variables $q_1, \cdots q_s$, not on any of the external variables. The following relationship (2) is supposed for the process of execution to be well-defied.

$$\begin{aligned} &\bigvee_{j=1}^{n} \alpha_{x_ix_j} \equiv 1 \quad (i=1, \cdots, n), \\ &\alpha_{x_ix_j} \wedge \alpha_{x_ix_k} \equiv 0 \quad (j \neq k). \end{aligned} \tag{2}$$

The entrance and the exit of the scheme need not be defined separately, because we have only to consider additional points representing them if necessary.

Definition 1. By a *logical scheme of algorithms with the implicit and the explicit operators*, or a logical scheme, for the simplicity, is meant what is defined as the above. It must be mentioned that the concept of shift distribution is not employed in our formulation because it can be regarded as a special case of the explicit operators. (See [1].) In this sense Yanov's model and also his suggested model are contained in our model. Therefore the equivalence of the schemes will be defined in such a manner that Yanov's formulation is extended, which will be done in the following manner. (See the Introduction of [1].)

A *sequence of assignments* is an infinite sequence of assignments of values to the external variables $p_1, \cdots, p_r$, corresponding to the thought that the essential information given to the program is carried by them. Before defining the process and the values of the scheme, we need some consideration. Firstly, the process will depend not only on the beginning point of the execution but also on the initial values of the internal variables. Therefore we consider the pair (x, Q) of a point and an assignment of values to the internal variables, which shall be termed an *initial condition*. Secondly, as regards the value of the scheme it is not adequate to consider only the sequence of the operator symbols. For, on one hand, the changes of values of the internal variables must be considered as essential to our problem when the original program (before abstraction) is to handle the logical or Boolean variables. On the other hand, the values of the variable corresponding to the expessions associated with the conditional branches (i.e. conditional *go to* statements) and of those variables which are used only as the parameters of the program

such as switch variables, working storages, and etc. will not be essential to our problem. Thus we shall call some of the internal variables the *significant variables*, whose values together with the sequence of the operator symbols are to define the value of the scheme. We shall assume that $q_1, \cdots, q_h$ $(0 \leq h \leq s)$ are significant, hereater.

Definition 2. The *process of carrying out the scheme* with the initial condition (x, Q) for the sequence of assignments $\mathcal{P}$:

$$P^1, P^2, \cdots, P^l, \cdots \tag{3}$$

is defined by induction as follows.

First step: Let us write the pair (x, Q) and the pair $(A_x, \bar{Q})$ as the first elements of the two lines (4) and (5), where $\bar{Q}$ denotes the values of the significant variables induced by Q. If the process has been done up to the l-th step and as the consequence lines

$$(x^1, Q^1), (x^2, Q^2), \cdots, (x^l, Q^l) \tag{4}$$

and

$$(A_{x^1}, \bar{Q}^1), (A_{x^2}, \bar{Q}^2), \cdots, (A_{x^l}, \bar{Q}^l) \tag{5}$$

have been written, then we compute

$$Q^{l+1} = F_{x^l}(P^l, Q^l)$$

and determine x^{l+1} by the relationship

$$\alpha_{x^l x^{l+1}}(Q^{l+1}) \equiv 1 .$$

(x^{l+1}, Q^{l+1}) and $(A_{x^{l+1}}, \bar{Q}^{l+1})$ will be added to (4) and (5) respectively.

Definition 3. The sequence (5) shall be termed the *value of the scheme* $\mathcal{A}$ with the initial condition (x, Q) for the sequence of assignements $\mathcal{P}$, and be denoted by $\bar{\mathcal{A}}_{(x, Q)}(\mathcal{P})$, while the process of carrying out the scheme shall be denoted by $\mathcal{A}_{(x, Q)}(\mathcal{P})$.

In Yanov's formulation the concept of permissible sequence was important in the definition of the equivalence of the schemes. Therefore we shall redefine it as follows. (See [1].)

Definition 4. A sequence of assignments (3) is termed a *permissible sequence* if for any l, P^l and P^{l+1} coincide except for the values of those external variables on which F_{x^l} depends actually.

Definition 5. Let $\mathcal{A}$ and $\mathcal{B}$ denote logical schemes of algorithms of r external variabes and h significant variables (the number of the internal variables may be different), and let ϕ denote a correspondence between the sets of all the initial conditions of $\mathcal{A}$ and $\mathcal{B}$. Then $\mathcal{A}$ and $\mathcal{B}$ are stated to be *equivalent in* ϕ if for any sequence of assignments $\mathcal{P}$ that is permissible in $\mathcal{A}$ or $\mathcal{B}$, $\bar{\mathcal{A}}_{(x, Q)}(\mathcal{P})$ coincides with $\bar{\mathcal{B}}_{\phi(x, Q)}(\mathcal{P})$, where if

$\phi(x, Q)$ is many-valued we consider all the images to determine the equivalence. This definition comes from the following fact, though the word "permissible" becomes rather inadequate.

Proposition 1.* For any Yanov's logical scheme of s logical variables and any given sift distribution, we can construct a logical scheme of s internal and s external variables that satisfies the condition: The permissible sequences for the former and for the latter are identical, and the values of the both schemes coincide for each permissible sequence.

Proposition 2. Let $\mathcal{A}$ and $\mathcal{B}$ denote logical schemes in Yanov's sense, and let $\mathcal{A}^*$ and $\mathcal{B}^*$ denote the logical schemes constructed from $\mathcal{A}$ and $\mathcal{B}$ respectively for a given shift distribution so as to satisfy the condition of proposition 1. Then $\mathcal{A}$ and $\mathcal{B}$ are equivalent in Yanov's sense if and only if $\mathcal{A}^*$ and $\mathcal{B}^*$ are equivalent in ϕ_0, which denotes the correspondence of the beginning points of $\mathcal{A}^*$ and $\mathcal{B}^*$.

The logical schemes of algorithms thus redefined are capable of describing the property of programs much better than Yanov's schemes. For example, the flow-diagram of 1(a) which can be represented by the scheme graphically shown in Fig. 1(b), has a property that the loop

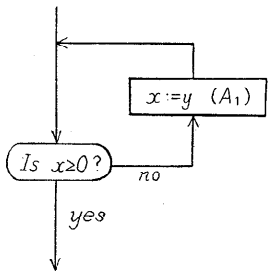

Fig. 1 (a)

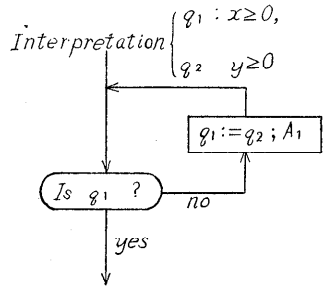

Fig. 1 (b)

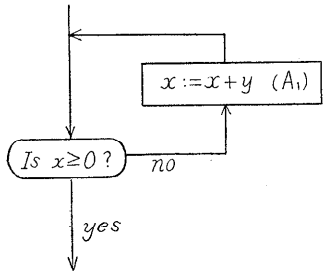

Fig. 2 (a)

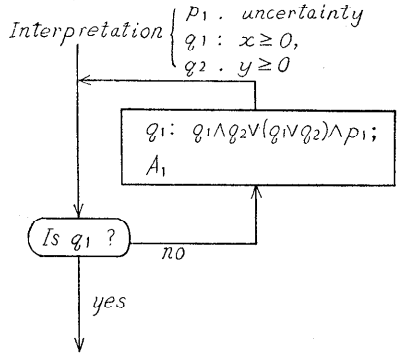

Fig. 2 (b)

* The proofs of the propositions and also the detailed discussion will be found in [6].

cannot be terminated unless it is not terminated after the first execution. Fig. 2 (a) is a flow-diagram containing an arithmetic operation whose property can be represented by the scheme of Fig. 2 (b). It must be mentioned that any program that performs logical operations can completely be represented by our schemes.

2. *Decision Procedure of the Equivalence and the Reduction of the Logical Schemes of Algorithms*

It will be the first problem after formulating the model to find a procedure deciding, for given schemes $\mathcal{A}$ and $\mathcal{B}$ and a given correspondence ϕ, whether $\mathcal{A}$ and $\mathcal{B}$ are equivalent in ϕ or are not. This problem can be reduced to that of finite automata, which will be discussen in this section.

Definition 6. $\mathcal{M}[\mathcal{A}]$ denotes a sequential machine that is constructed from a logical scheme $\mathcal{A}$ as follows.

The set of *states* is the direct product of the set of points and the set of assignments of values to the internal variables of $\mathcal{A}$ (thus the states of $\mathcal{M}[\mathcal{A}]$ and the initial conditions of $\mathcal{A}$ are identical). The set of *inputs* is the set of assignments of values to the external variables of $\mathcal{A}$. With each state (x, Q) is uniquely associated an *output* $(A_x, \bar{Q})$.

Proposition 3. The sequence of outputs of $\mathcal{M}[\mathcal{A}]$ for the input sequence $\mathcal{P}$ starting from the state (x, Q) coincides with $\bar{\mathcal{A}}_{(x,Q)}(\mathcal{P})$.

The above is obvious, but it is not apparent that the equivalence of $\mathcal{A}$ and $\mathcal{B}$ coincides with the equivalence of $\mathcal{M}[\mathcal{A}]$ and $\mathcal{M}[\mathcal{B}]$, because the former depends only on those $\mathcal{P}$ that are permissible while the latter depends on all $\mathcal{P}$. We proceed as follows.

Proposition 4. For any scheme $\mathcal{A}$ and any initial condition (x, Q), there exists an equivalence relation $\sim$ in the set of the sequence of assignments satisfying the following relationships.

1) $\mathcal{P}\sim\mathcal{P}'$ implies $\bar{\mathcal{A}}_{(x,Q)}(\mathcal{P})=\bar{\mathcal{A}}_{(x,Q)}(\mathcal{P}')$.
2) For any $\mathcal{B}$, there is exactly one permissible sequence $\hat{\mathcal{P}}$ such that $\mathcal{P}\sim\hat{\mathcal{P}}$.

Proposition 5. For the schemes $\mathcal{A}$ and $\mathcal{B}$ to be equivalent in ϕ it is necessary and sufficient that for any sequence of assignments $\mathcal{P}$, $\bar{\mathcal{A}}_{(x,Q)}(\mathcal{P})=\bar{\mathcal{B}}_{\phi(x,Q)}(\mathcal{P})$.

Definition 7. For the sequential machines $\mathcal{M}$ and $\mathcal{N}$ and for a correspondence ϕ between the sets of states of them, we write $\mathcal{M}\underset{\phi}{\simeq}\mathcal{N}$ if for any t such that $\phi(s)=t$, the output sequence of $\mathcal{M}$ starting from the state s and that of $\mathcal{N}$ starting from t for any input sequence coincide

with each other.

Proposition 6. Schemes $\mathcal{A}$ and $\mathcal{B}$ are equivalent in ϕ if and only if $\mathcal{M}[\mathcal{A}] \underset{\phi}{\simeq} \mathcal{M}[\mathcal{B}]$, so that it is effectively decidable.

E. F. Moore's well-known method of determining the equivalence [2] can be applied in principle. His method needs, however, a very large number of steps, because it is based on the assumption that the internal structure of the machines are unknown. Therefore it will be useful to find a method determining the equivalence with the assumption that the structure is known, for we always know the structure of the logical schemes and therefore that of the corresponding machines.

Definition 8. Let s and t denote one of the states of sequential machines $\mathcal{M}$ and $\mathcal{N}$, then the pair (s, t) is stated to be *apparently incompatible* if the outputs corresponding to them are different. By $C[\mathcal{M}, \mathcal{N}]$ is denoted the precedence matrix representing the precedence relation $\prec$ in the set of all the pairs of states that are *not* apparently incompatible and a special element "!", such that $(s, t) \prec (s', t')$ either if $s=s'$ and $t=t'$ or if there exists at least one input by which s and t change into s' and t' respectively. If the pair (s', t') is apparently incompatible and the latter condition is satisfied, then we write $(s, t) \prec !$.

Proposition 7. Let C^* denote the limit of the sequence

$$C,\ C^2,\ C^4=(C^2)^2,\ C^8=(C^4)^2,\ \cdots,$$

and let $\overset{*}{\prec}$ denote the precedence relation represented C^*. Then $\mathcal{M} \underset{\phi}{\simeq} \mathcal{N}$ if and only if for any s and $\phi(s)$ the relationship $(s, \phi(s)) \overset{*}{\prec} !$ does not hold. For the proof it will suffice to notice that $(s, \phi(s)) \overset{*}{\prec} !$ if and only if there is some sequence such as $(s, \phi(s)) \prec \cdots \prec !$, by the well-known property of precedence matrices [5], so that if $(s, \phi(s)) \overset{*}{\prec} !$ is not the case, then for any input sequence the output sequences starting from s and $\phi(s)$ coincide.

The procedure given by proposition 7 is rapid in convergence and the necessary size of the storage is moderate. Moreover, precedence or Boolean matrices can easily be handled by computers. Therefore this method might be said more computer-oriented than those of M. C. Paull and S. H. Unger [4], and of S. Ginsburg [3].

The problem of reducing or simplifying a given logical scheme $\mathcal{A}$ depends on the criterion of the simplicity. A solution to this problem will be obtained by reconstructing a scheme from the minimal state machine reduced from $\mathcal{M}[\mathcal{A}]$, for which we can use the above method again as follows.

Proposition 8. Let $C^*[\mathcal{M}, \mathcal{M}]$ denote the precedence matrix satisfying the condition of proposition 7 for the same machine $\mathcal{M}$. Then the states

s and t are equivalent if and oly if *not* $(s, t) \overset{*}{\prec} !$, where $\overset{*}{\prec}$ denotes the precedence relation represented by $C^*[\mathcal{M}, \mathcal{M}]$. It is well-known that the minimal state machine or canonical form of $\mathcal{M}$ can be obtained simply by superposing each equivalent class of the states of $\mathcal{M}$ [2].

It will surely be one of the proper theoretical approach to computer programs, for which we have only a few examples yet, to check and simplify them by the help of the theory of the logical schemes of algorithms. Though a sufficient improvement of this kind of theoretical approach to computer programs seems not to be easy, the present writer considers that it is important and promising.

Acknowledgement

The writer sincerely acknowledges Professor S. Moriguti, and Professor M. Takata, of the University of Tokyo for their suggestions.

References

[1] YANOV, Y. I., O logicheskikh skhemakh algoritmov. *Problemy Kibernetiki, 1* (1958). English edition: The logical schemes of algorithms. *Problems of Cybernetics, 1* (1960), 82–140.

[2] MOORE, E. F., Gedanken-Experiments of sequential machines. *Automata Studies*, Annals of Mathematics Studies. No. 34, Princeton University Press (1956), 129–153.

[3] GINSBURG, S., A technique for the reduction of a given sequential machine to a minimal-state machine. *IRE Trans. on Electric Computers, EC-8, 3* (Sept. 1951), 346–355.

[4] PAULL, M. C. AND S. H. UNGER, Minimizing the number of states in incompletely specified sequential switching functions. *IRE Trans. on Electric Computers, EC-8, 3* (Sept. 1959) 346–367.

[5] PROSSER, R. T., Applications of boolean matrices to the analysis of flow diagrams. *Proc. of EJCC*, 1959, 133–138.

[6] IGARASHI, S., Contributions to the theory of computer programs. Dissertation for the Master's Degree submitted to the University of Tokyo, 1962.