ON THE SIMULATION OF REAL-TIME TURING MACHINES BY A MODIFIED SCHEPHERDSON-STURGIS' MACHINE

By

Seiiti HUZINO

(Received Nov. 10, 1965)

Abstract.

By making use of the programming technique developed in [1], we shall study a problem whether a given real-time Turing machine with k work-tapes is simulatable by a modified Scheperdson-Sturgis' machine or not, and give an affirmative solution of the question by constructing a simulation program in such a machine.

1. Introduction and preliminaries.

In [1] we have shown the simulatability of finite automata by some modified Scheperdson-Sturgis' machines. The programming technique used in [1] seems to be applicable for other cases, which we have noticed from our initial investigation of the simulatability problems. In this paper we shall take as an example a real-time Turing machine, firstly studied by M.O. Rabin ([3]), and shall consider a problem whether the machine is simulatable by a modified Scheperdson-Sturgis' machine, a solution of which we shall construct in this paper.

Firstly, let us give the definitions of a *real-time Turing machine*, following M.O. Rabin's paper ([3]).

Assume now that a finite (non-empty) set Σ , called the input alphabet, is given in advance and let the elements of the alphabet be $\sigma[1], \sigma[2], \dots, \sigma[p]$. A real-time Turing machine with k work-tapes over Σ is a system $\mathfrak{U}=(S, M, s[0], F, W)$, where S is a finite (non-empty) set of states, $s[0] (\in S)$ is the initial state of $\mathfrak{U}$, F is a subset of S , singled out as the set of designated final states, W is a working alphabet $\{\tau[0], \tau[1], \dots, \tau[q-1]\}$ and M is a mapping from $S \times \Sigma \times W^k$ into $S \times W^k \times Z_3^k$. Here Z_3 is the set $\{-1, 0, 1\}$. Intuitively speaking, the machine has a finite automaton as its body and has k two-way infinite linear work-tapes which are divided into squares. Furthermore, there is a reading-printing head which at any given time reads one

square on each of the work-tapes. The machine has another input tape in which a finite sequence of letters in Σ (i.e. word over Σ) is written as input.

The operations of M are interpreted as follows:

Assume that

$$\begin{aligned} M(s[i], \sigma[j], \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k]) \\ = (s[i'], \tau[\alpha'_1], \tau[\alpha'_2], \dots, \tau[\alpha'_k], e_1, e_2, \dots, e_k), \end{aligned}$$

where $s[i], s[i'] \in S$, $\sigma[j] \in \Sigma$, $\tau[\alpha_r], \tau[\alpha'_r] \in W$, $e_r \in Z_3$, ($r=1, 2, \dots, k$). Now if the input is $\sigma[j]$ and $\mathfrak{M}$ is in the state $s[i]$ and the sequence of letters $(\tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k])$ is read from k work-tapes by the reading-printing head, $\tau[\alpha_r]$ being on the r -th tape ($r=1, 2, \dots, k$), then $\mathfrak{M}$ will transit into state $s[i']$, overprint $\tau[\alpha'_r]$ on the square scanned on the r -th tape ($r=1, 2, \dots, k$) and move each tape one square left, or one square right, or not at all, according as to whether e_r is equal to 1, -1 , or 0. This action has been called an atomic move of $\mathfrak{M}$ by M. O. Rabin ([3]).

We denote by $\theta(\Sigma)$ the set of all finite sequences (called words) on the alphabet Σ , including the empty word Λ . Following M. O. Rabin we shall introduce a real-time definable set as follows: a word $w = w_1 \cdot w_2 \cdot \dots \cdot w_l$ ($w_r \in \Sigma$ ($r=1, 2, \dots, l$)) is called to be *accepted* by $\mathfrak{M}$ if, when started in state $s[0]$ and with the contents of work tapes $(\tau[0], \tau[0], \dots, \tau[0])$, $\mathfrak{M}$ will go under the word w through atomic moves and end in a state in F . Here we assume that every square of k work-tapes is filled with blank symbol $\tau[0]$ at the beginning. The set of all words accepted by $\mathfrak{M}$ is called the set defined by $\mathfrak{M}$ and is denoted by $T(\mathfrak{M})$.

When a real-time Turing machine $\mathfrak{M}$ with k work-tapes over Σ is given, there arises a problem whether a given word belongs to $T(\mathfrak{M})$ or not. The question can be immediately solved by following atomic moves of $\mathfrak{M}$ with initial state $s[0]$ and by checking whether the final state of $\mathfrak{M}$ belongs to F or not. *Isn't it possible to program the problem by a simple idealized machine with stored program?* The problem is one about a simulation of real-time Turing machine by another machine. We shall treat the problem more concretely by taking the following machine $M_3(\mathcal{A}, \mathfrak{Z})$ as an example.

The machine $M_3(\mathcal{A}, \mathfrak{Z})$ is a machine of S-S type with unlimited registers (the definition of an S-S type machine has been given in [1].) and it has the set $\Sigma \cup W$ and the following set of instructions as the alphabet $\mathcal{A}$ and the set $\mathfrak{Z}$, respectively:

- 3: (1) $P^{(i)}(l)$: After shifting each letter of $\langle l \rangle$ one place right, place $\tau[i]$ on the left end of the new $\langle l \rangle$, ($i=0, 1, \dots, q-1$).
- (2) $\tilde{P}^{(0)}$: After shifting each letter of $\langle 1 \rangle$ one place right, place $\sigma[1]$ on the left end of the new $\langle 1 \rangle$.
- (3) $D(l)$: Delete the first letter of $\langle l \rangle$. (As a consequence, each letter of $\langle l \rangle$ moves one place left.)
- (4) $J^{(i)}(l)[E1]$: Jump to exit $E1$ if $\langle l \rangle$ begins with $\tau[i]$. ($i=0, 1, 2, \dots, q-1$)
- (5) $Scd(l)[E1, E2, \dots, Ep]$: Scan the first letter of $\langle l \rangle$: if $\langle l \rangle = A$, take normal exit; if first letter of $\langle l \rangle$ is $\sigma[i]$, then delete this and proceed to exit Ei ($i=1, 2, \dots, p$).

Here $\langle l \rangle$ denotes the content of register l .

The machine is slightly different from a Scheperdson-Sturgis' machine $URM(A)$, for the latter has instructions of placing a letter on the right end of register, but the former has those of placing a letter on the left end of register, which makes us get easy programming of the problem. We are able to treat all registers as if they are push-down stores.

Now we shall give the definition of simulatability in this case: Let $\mathfrak{M}[\Sigma, W, k]$ be the set of all real-time Turing machines with k work-tapes over Σ , where the input alphabet Σ , the working alphabet W and the number of work-tapes k are fixed. The set $\mathfrak{M}[\Sigma, W, k]$ is said to be *simulatable* by a machine of S-S type with unlimited registers if for each machine $\mathfrak{U} \in \mathfrak{M}[\Sigma, W, k]$ there exists a program $\mathfrak{P}_{\mathfrak{U}}$ of a machine of S-S type such that the associated mapping $\pi_{\mathfrak{U}}$ from $\theta(\Sigma)$ into $\theta(\Sigma)$ has the property

$$\pi_{\mathfrak{U}}(w) = \begin{cases} \sigma[1] & \text{in register 1 if } w \in T(\mathfrak{U}), \\ A & \text{in register 1 if } w \notin T(\mathfrak{U}), \end{cases}$$

when w is inserted in register 1 in advance, where the mapping $\pi_{\mathfrak{U}}: \theta(\Sigma) \rightarrow \theta(\Sigma)$ is the mapping associated with the program $\mathfrak{P}_{\mathfrak{U}}$, the image of which is defined as the content of register 1 at the time when the machine stopped after a finite number of operations of the program $\mathfrak{P}_{\mathfrak{U}}$.

2. Simulatability of the set $\mathfrak{M}[\Sigma, W, k]$ by $M_3[A, \mathfrak{S}]$.

In this section we shall consider the problem of simulatability of the set $\mathfrak{M}[\Sigma, W, k]$ by the machine $M_3[A, \mathfrak{S}]$. There arises a problem which did not

happen in considering the problem of simulating finite automata. It is the problem how to embed the given k work-tapes into registers. We shall use the registers $2r$ and $2r+1$ as those containing the right half part and the left half part of the r -th work tape, respectively, considering the left end of the content of register $2r$ as the content of the scanned square and the left end of the content of the register $2r+1$ as the content of left one of scanned square of the tape ($r=1, 2, \dots, k$). And we use the register 1 as the place containing the input tape of $\mathfrak{A}$, and also use it as the register containing the output $\sigma[1]$ or A after the completion of operations.

We can prove the following theorem.

THEOREM. *The set $\mathfrak{M}[\Sigma, W, k]$ is simulatable by the machine $M_3[A, \mathfrak{S}]$.*

PROOF. Let $\mathfrak{A}=(S, M, s[0], F, W)$ be an element of $\mathfrak{M}[\Sigma, W, k]$, where $S, M, s[0], F, W$ have the meaning previously explained. For simplicity, let us write

$$a_{i,j,\alpha} \equiv M(s[i], \sigma[j], \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k])$$

for $s[i] \in S$, $\sigma[j] \in \Sigma$ and $\tau[\alpha_r] \in W$ ($r=1, 2, \dots, k$), where α is defined by $\alpha = \sum_{j=1}^k \alpha_j q^{k-j}$, i.e., α is a number which is represented as a q -ary number by $(\alpha_1, \alpha_2, \dots, \alpha_k)$. Then, by the definition of M , $a_{i,j,\alpha} \in S \times W^k \times Z_s^k$ for any $i \in {}_0N_{n-1}$, $j \in {}_1N_p$ and $\alpha \in {}_0N_{q^k-1}$. Here ${}_0N_{n-1}$ means the set $\{0, 1, \dots, n-1\}$. The set ${}_1N_p$ and ${}_0N_{q^k-1}$ have the similar meanings:

$${}_1N_p = \{1, 2, \dots, p\} \quad \text{and} \quad {}_0N_{q^k-1} = \{0, 1, 2, \dots, q^k-1\}.$$

The set $\{a_{i,j,\alpha}\}_{i,j,\alpha}$ determines the atomic moves of the machine $\mathfrak{A}$.

Firstly, we shall make a subroutine representing the behavior of the main part of an atomic move, i.e., overprinting of working-letters and movings of work-tapes.

Consider the following subroutines:

- $m_{+1}(j)$: send the first letter of $\langle 2j \rangle$ to the left end of the register $2j+1$ and if $\langle 2j \rangle$ gets empty after the above operation, then place $\tau[0]$ on the left end of the register $2j$.
- $m_{-1}(j)$: send the first letter of $\langle 2j+1 \rangle$ to the left end of the register $2j$, and if $\langle 2j+1 \rangle$ gets empty after the above operation, then place $\tau[0]$ on the left end of the register $2j+1$.
- $m_0(j)$: proceed to the next instruction without any action.
(Actually we omit it in the last step of programming).

Act $[j_1, j_2, \dots, j_k, e_1, e_2, \dots, e_k] (j_1, j_2, \dots, j_k \in {}_0N_{q-1}, e_1, e_2, \dots, e_k \in Z_3)$:

delete the first letters of $\langle 2 \rangle, \langle 4 \rangle, \dots, \langle 2k \rangle$ and place the letters $\tau[j_1], \tau[j_2], \dots, \tau[j_k]$ on the left end of the registers 2, 4, $\dots, 2k$, respectively. And do the operations $m_{e_1}(2), m_{e_2}(4), \dots, m_{e_k}(2k)$.

The subroutine $m_{+1}(j)$ (or $m_{-1}(j)$) is the operation simulating the behavior of letting the scanned square of j -th tape move one square right (or left). The subroutine Act $[j_1, j_2, \dots, j_k; e_1, e_2, \dots, e_k]$ has the meaning of simulating the following behavior of the machine under the conditions that we consider the combined set of the registers $2r$ and $2r+1$ as a representation of the r -th tape of $\mathfrak{M}$ ($r=1, 2, \dots, k$). The machine overprints the letters $\tau[j_1], \tau[j_2], \dots, \tau[j_k]$ on the scanned squares of each work-tape, i.e., $\tau[j_r]$ on the scanned square of the r -th tape, and let the scanned squares move one square left, right, or not at all, according to the values of the numbers $e_1, e_2, \dots, e_k$, respectively.

In order to construct the above subroutines, we shall firstly prepare the following subroutine:

$J(l)[E1, E2, \dots, Eq]$: if $\langle l \rangle = A$, take normal exit, and if $\langle l \rangle \neq A$ and the first letter of $\langle l \rangle$ is $\tau[i]$, then proceed to exit $E(i+1)$ ($i=0, 1, \dots, q-1$). It is easily constructed by using basic instructions as follows:

1. $J^{(0)}(l)[E1], J^{(1)}(l)[E2], \dots, J^{(q-1)}(l)[Eq]$.

The programs for $m_{+1}(j)$, $m_{-1}(j)$ and Act $[j_1, j_2, \dots, j_k; e_1, e_2, \dots, e_k]$ are now obtained as follows: (assuming that $m_{+1}(j)$ and $m_{-1}(j)$ are not used in the case when $\langle j \rangle$ is empty and that the content of the register $2k+2$ is always $\tau[0]$ during computation, which is realized by putting the instruction $P^{(0)}(2k+2)$ at the beginning of the complete program.)

- $m_{+1}(j)$:
1. $J(2j)[2, 3, \dots, q+1]$,
 2. $D(2j), P^{(0)}(2j+1), J(2j)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 3. $D(2j), P^{(1)}(2j+1), J(2j)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 - $\vdots$
 - $q+1$. $D(2j), P^{(q-1)}(2j+1), J(2j)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 - $q+2$. $P^{(0)}(2j)$.
- $m_{-1}(j)$:
1. $J(2j+1)[2, 3, \dots, q+1]$,
 2. $D(2j+1), P^{(0)}(2j), J(2j+1)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 3. $D(2j+1), P^{(1)}(2j), J(2j+1)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 - $\vdots$
 - $q+1$. $D(2j+1), P^{(q-1)}(2j), J(2j+1)[q+3, q+3, \dots, q+3], J^{(0)}(2k+2)[q+2]$,
 - $q+2$. $P^{(0)}(2j+1)$.

$Act[j_1, j_2, \dots, k_k; e_1, e_2, \dots, e_k]:$

1. $D(2), P^{(j_1)}(2), m_{e_1}(2),$
 $D(4), P^{(j_2)}(4), m_{e_2}(4),$
 $\vdots$
 $D(2k), P^{(j_k)}(2k), m_{e_k}(2k).$

Here we omit the subroutine $m_0(2r)$ in this program when $e_r=0$ for some r .

To let these subroutines move effectively, we must operate the program after we have inserted the letter $\tau[0]$ into all registers $2, 3, \dots, 2k+2$ at the start of the program. If we put a sequence of instructions $P^{(0)}(2), P^{(0)}(3), P^{(0)}(4), \dots, P^{(0)}(2k+1), P^{(0)}(2k+2)$ to the first part of a complete program, then we are able to use the above subroutines effectively. Thus we can easily construct a program corresponding to the work of the main part of an atomic move $a_{i,j,\alpha}$ by making use of the above subroutines: If

$$\begin{aligned} a_{i,j,\alpha} &\equiv M(s[i], \sigma[j], \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k]) \\ &= (s[i'], \tau[\alpha'_1], \tau[\alpha'_2], \dots, \tau[\alpha'_k], e_1, e_2, \dots, e_k), \end{aligned}$$

then the subroutine $Act[\alpha'_1, \alpha'_2, \dots, \alpha'_k; e_1, e_2, \dots, e_k]$ is the required one. To simplify the description, let us introduce a mapping $\text{ind}_2: S \times W^k \times Z_3^k \rightarrow N^k \times Z_3^k$ defined by

$$\begin{aligned} \text{ind}_2(s[i], \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k]; e_1, e_2, \dots, e_k) &= (\alpha_1, \alpha_2, \dots, \alpha_k; e_1, e_2, \dots, e_k) \\ \text{for } s[i] \in S, \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k] \in W, e_1, e_2, \dots, e_k \in Z_3. \end{aligned}$$

Then we can write the above subroutine $Act[\alpha'_1, \alpha'_2, \dots, \alpha'_k; e_1, e_2, \dots, e_k]$ as $Act[\text{ind}_2(a_{i,j,\alpha})]$.

It remains how to treat the state-transitions of the given machine, which is more complex than the cases of simulations of finite automata, but to which, essentially, the same techniques can apply.

We prepare the following subroutines:

$\text{Det}[E1, E2, \dots, E q^k]:$ If the first letters of $\langle 2 \rangle, \langle 4 \rangle, \dots, \langle 2k \rangle$ are $\tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k]$, respectively, then take exit $E(\alpha+1)$, where $\alpha = \sum_{j=1}^k \alpha_j q^{k-j}$.

$$K \left\{ \begin{array}{l} E1; E_{1,1}, E_{1,2}, \dots, E_{1,q^k} \\ E2; E_{2,1}, E_{2,2}, \dots, E_{2,q^k} \\ \vdots \\ Ej; E_{j,1}, E_{j,2}, \dots, E_{j,q^k} \\ \vdots \\ Ep; E_{p,1}, E_{p,2}, \dots, E_{p,q^k} \end{array} \right\} : \begin{array}{l} \text{It has } p \text{ entrances } E1, E2, \dots, Ep, \\ \text{and if the present entrance is } Ej \text{ and} \\ \text{the first letters of } \langle 2 \rangle, \langle 4 \rangle, \dots, \\ \langle 2k \rangle \text{ are } \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k] \text{ re-} \\ \text{spectively, then jump to } E_{j, \alpha+1}, \\ \text{where } \alpha = \sum_{j=1}^k \alpha_j q^{k-j}. \end{array}$$

The latter is obtained by using the former as follows:

$$\begin{array}{ll} E1. & \text{Det } [E_{1,1}, E_{1,2}, \dots, E_{1,q^k}], \\ E2. & \text{Det } [E_{2,1}, E_{2,2}, \dots, E_{2,q^k}], \\ & \vdots \\ Ep. & \text{Det } [E_{p,1}, E_{p,2}, \dots, E_{p,q^k}]. \end{array}$$

The first subroutine is essentially a program of selecting a number α . We have no calculating (or adding) instructions explicitly, but, by using combinations of jump-instructions, we can construct the required program. The program for $\text{Det } [E1, E2, \dots, E_{q^k}]$ is now:

$$\begin{array}{l} 1. \quad J(2) [2, 3, \dots, q+1], \\ q \left\{ \begin{array}{l} 2. \quad J(4) [q+2, q+3, \dots, 2q+1], \\ 3. \quad J(4) [2q+2, 2q+3, \dots, 3q+1], \\ \vdots \\ q+1. \quad J(4) [q^2+2, q^2+3, \dots, q^2+q+1], \end{array} \right. \\ q \left\{ \begin{array}{l} q+2. \quad J(6) [(q+1)q+2, (q+1)q+3, \dots, (q+2)q+1], \\ q+3. \quad J(6) [(q+2)q+2, (q+2)q+3, \dots, (q+3)q+1], \\ \vdots \\ 2q+1. \quad J(6) [2q^2+2, 2q^2+3, \dots, (2q+1)q+1], \end{array} \right. \\ q \left\{ \begin{array}{l} 2q+2. \quad J(6) [(2q+1)q+2, (2q+1)q+3, \dots, (2q+2)q+1], \\ 2q+3. \quad J(6) [(2q+2)q+2, (2q+2)q+3, \dots, (2q+3)q+1], \\ \vdots \\ 3q+1. \quad J(6) [3q^2+2, 3q^2+3, \dots, (3q+1)q+1], \end{array} \right. \\ q \left\{ \begin{array}{l} \vdots \\ q^2+2. \quad J(6) [(q^2+1)q+2, (q^2+1)q+3, \dots, (q^2+2)q+1], \\ q^2+3. \quad J(6) [(q^2+2)q+2, (q^2+2)q+3, \dots, (q^2+3)q+1], \\ \vdots \\ (q+1)q+1. \quad J(6) [(q+1)q^2+2, (q+1)q^2+3, \dots, q^3+q^2+q+1], \\ \vdots \\ \vdots \\ \vdots \end{array} \right. \end{array}$$

$$\begin{array}{l}
\left. \begin{array}{l}
q \\
q \\
q^{k-2}
\end{array} \right\} \begin{array}{l}
Q_{k-2}. \quad J(2(k-1)) \left[\begin{array}{c} \vdots \\ \vdots \\ \vdots \end{array}, q^{k-1} + q^{k-2} + \dots + q + 1 \right], \\
Q_{k-2}+1. \quad J(2k) [E1, E2, \dots, Eq], \\
Q_{k-2}+2. \quad J(2k) [E(q+1), E(q+2), \dots, E(2q)], \\
\vdots \\
Q_{k-2}+q. \quad J(2k) [E((q-1)q+1), E((q-1)q+2), \dots, Eq^2], \\
Q_{k-2}+q+1. \quad J(2k) [E(q^2+1), E(q^2+2), \dots, E(q^2+q)], \\
Q_{k-2}+q+2. \quad J(2k) [E(q^2+q+1), E(q^2+q+2), \dots, E(q^2+2q)], \\
\vdots \\
Q_{k-2}+2q. \quad J(2k) [E((2q-1)q+1), E((2q-1)q+2), \dots, E(2q^2)], \\
\vdots \\
\vdots \\
\vdots \\
Q_{k-1}-q+1. \quad J(2k) [E((q^{k-2}-1)q^2+1), E((q^{k-2}-1)q^2+2), \\
\qquad \qquad \qquad \dots, E((q^{k-2}-1)q^2+q)], \\
Q_{k-1}-q+2. \quad J(2k) [E(q^k-q^2+q+1), E(q^k-q^2+q+2), \\
\qquad \qquad \qquad \dots, E(q^k-q^2+2q)], \\
\vdots \\
Q_{k-1}. \quad J(2k) [E(q^k-q+1), E(q^k-q+2), \dots, E(q^k)].
\end{array}
\end{array}$$

Here, $Q_r = \sum_{j=1}^r q^j$, ($r=1, 2, \dots, k-1$). We could make the program for recognizing a number $\alpha \in {}_0N_{q^{k-1}}$.

Next we define $\text{Scd}'(l) [E1, E2, \dots, E(p+1)]$, a $(p+2)$ -exits instruction which differs from $\text{Scd}(l) [E1, E2, \dots, Ep]$ in that when $\langle l \rangle = A$ exit $E(p+1)$ is taken instead of normal exit (i.e., which provides a jump on A as well as the other jumps):

$$\begin{aligned}
&\text{Scd}'(l) [E1, E2, \dots, E(p+1)] = \\
&1. \quad \text{Scd}(l) [E1, E2, \dots, Ep], \quad P^{(1)}(l), \\
&\quad \text{Scd}(l) [E(p+1), E(p+1), \dots, E(p+1)].
\end{aligned}$$

We shall use this instruction as an initialization of the program and also as a jump instruction appearing in state-transitions.

We shall construct a complete program under these preparations. First of all we put a sequence of instructions $P^{(0)}[2], P^{(0)}[3], P^{(0)}[4], \dots, P^{(0)}[2k+1], P^{(0)}[2k+2]$ at the initial part of the program, which does storing of letter $\tau[0]$ into all registers $2, 3, \dots, 2k+2$. We assume from the beginning that

the content of the register 1 is w , a given input word. Next to the above instructions, we put an instruction $\text{Scd}'(l) [2, 3, \dots, p+1, np(1+q^k)+2]$. If the first letter of w in the register 1 is $\sigma[i]$, then, by the above instruction, the next instruction is taken from the register $i+1$, in which we put a sequence of instructions which does an operation recognizing the number α representing the present contents of the scanned squares of k work-tapes and so it can simulate the work recognizing the contents of the scanned squares of k work-tapes when the state is $s[0]$. If the word w is an empty word, then the next instruction is taken from $np(1+q^k)+2$, in which we put an instruction which simulates the work recognizing whether the final state (now, state $s[0]$) belongs to F or not. To accomplish these operations completely, let us introduce the following symbols $K(x)$:

$$K(x) \equiv K \left(\begin{array}{l} x+2; xq^k+np+2, xq^k+np+3, \dots, (x+1)q^k+np+1 \\ x+3; (x+1)q^k+np+2, (x+1)q^k+np+3, \dots, (x+2)q^k+np+1 \\ \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ x+r; (x+r-2)q^k+np+2, (x+r-2)q^k+np+3, \dots, (x+r-1)q^k+np+1 \\ \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\ x+p+1; (x+p-1)q^k+np+2, (x+p-1)q^k+np+3, \dots, (x+p)q^k+np+1 \\ (x=0, p, 2p, 3p, \dots, (n+1)p) \end{array} \right)$$

The $K(x)$ has p entrances $x+2, x+3, \dots, x+p+1$ and (pq^k+p) exits (but p exits are not used in this case), and it plays the operation corresponding to the work recognizing the present input letter and the present contents of the scanned squares of k work-tapes by the definition of the subroutine K . If every exit is connected to the starting position of a sequence of the instructions which simulates the work corresponding to the main part of atomic moves of $\mathfrak{A}$, then atomic moves of $\mathfrak{A}$ can be simulated by the program except the state-transitions. Therefore, the $(n-1)$ subroutines $K(x)$ are placed to the positions corresponding to the state $s[x/p]$ ($x=0, p, 2p, \dots, (n-1)p$).

Next we shall introduce the following two mappings $\text{ind}_1: S \times W^k \times Z_3^k \rightarrow N$ and $f: {}_0N_{n-1} \times {}_1N_p \rightarrow N$ defined by

$$\begin{aligned} \text{ind}_1(s[i], \tau[\alpha_1], \tau[\alpha_2], \dots, \tau[\alpha_k], e_1, e_2, \dots, e_k) &= i \\ \text{for } s[i] \in S, \tau[\alpha_r] \in W, e_r \in Z_3 \quad (r=1, 2, \dots, k) \end{aligned}$$

and

$$f(i, j) = ip + j + 1 \quad (i=0, 1, \dots, n-1; j=1, 2, \dots, p).$$

And let us write

$$B(i, j, \alpha) \equiv \text{Act}(\text{ind}_2(a_{i, j, \alpha}))$$

and

$$\begin{aligned} C(i, j, \alpha) \equiv & \text{Sed}' [f(\text{ind}_1(a_{i, j, \alpha}), 1), f(\text{ind}_1(a_{i, j, \alpha}), 2), \dots, \\ & f(\text{ind}_1(a_{i, j, \alpha}), p), np(1+q^k) + \text{ind}_1(a_{i, j, \alpha}) + 2] \\ & (i=0, 1, \dots, n-1; j=1, 2, \dots, p; \alpha=0, 1, \dots, q^k-1), \end{aligned}$$

where ind_2 has been defined before.

Using these notations let us denote by $H(x)$ the following sequences of instructions (with addresses). ($x=0, 1, \dots, n-1$):

$$\begin{array}{lll} xpq^k + np + 2. & B(x, 1, 0), & C(x, 1, 0), \\ xpq^k + np + 3. & B(x, 1, 1), & C(x, 1, 1), \\ \vdots & \vdots & \vdots \\ xpq^k + q^k + np + 1. & B(x, 1, q^k - 1), & C(x, 1, q^k - 1), \\ xpq^k + q^k + np + 2. & B(x, 2, 0), & C(x, 2, 0), \\ xpq^k + q^k + np + 3. & B(x, 2, 1), & C(x, 2, 1), \\ \vdots & \vdots & \vdots \\ xpq^k + 2q^k + np + 1. & B(x, 2, q^k - 1), & C(x, 2, q^k - 1), \\ \vdots & \vdots & \vdots \\ xpq^k + (p-1)q^k + np + 2. & B(x, p, 0), & C(x, p, 0), \\ xpq^k + (p-1)q^k + np + 3. & B(x, p, 1), & C(x, p, 1), \\ \vdots & \vdots & \vdots \\ (x+1)pq^k + np + 1. & B(x, p, q^k - 1), & C(x, p, q^k - 1) \\ (x=0, 1, 2, \dots, n-1). \end{array}$$

It is evident that the sequence of instructions $H(x)$ plays the role corresponding to the atomic moves of $\mathfrak{A}$ when the present state is $s[x]$. And in this sequence the next state transition is also considered, since each instruction $C(x, j, \alpha)$ follows after the subroutine $B(x, j, \alpha)$ and it connects the operation of the program to the instruction recognizing the number α' at the next state.

Finally we add to the bottom of the program a sequence of instructions which simulates the work recognizing whether the final state belongs to F or not.

We define a mapping $\text{Ad}: S \rightarrow \{np(1+q^k) + n + 2, np(1+q^k) + n + 3\}$ by

$$\text{Ad}(s[i]) = \begin{cases} np(1+q^k) + n + 2 & (s[i] \in F), \\ np(1+q^k) + n + 3 & (s[i] \notin F). \end{cases}$$

The sequence $J^{(0)}(2k+2) [\text{Ad}(s[0])], J^{(0)}(2k+2) [\text{Ad}(s[1])], \dots, J^{(0)}(2k+2)$

$[\text{Ad}(s[n-1])]$ is the required one when we put $\tilde{P}^{(0)}$ to the place $np(1+q^k)+n+2$.

Thus, the complete program is described as follows:

1. $P^{(0)}(2), P^{(0)}(3), P^{(0)}(4), \dots, P^{(0)}(2k+1), P^{(0)}(2k+2),$
 $\text{Scd}'(1)(2, 3, \dots, p+1, np(1+q^k)+2),$
 $K(0),$
 $K(p),$
 $K(2p),$
 $\vdots$
 $K(rp),$
 $\vdots$
 $K((n-1)p),$
 $H(0),$
 $H(1),$
 $\vdots$
 $H(n-1),$
 $np(1+q^k)+2. \quad J^{(0)}(2k+2) [\text{Ad}(s[0])],$
 $np(1+q^k)+3. \quad J^{(0)}(2k+2) [\text{Ad}(s[1])],$
 $\vdots$
 $np(1+q^k)+n+1. \quad J^{(0)}(2k+2) [\text{Ad}(s[n-1])],$
 $np(1+q^k)+n+2. \quad \tilde{P}^{(0)}.$

Acknowledgement

This paper had been prepared while the author was in the *University of Western Ontario*. He would like to thank for *National Research Council of Canada* which gave him the opportunity to visit the department of computer science of the University for an academic year 1964-1965. He would like also to thank for all staffs of the department, headed by Dr. J. F. Hart, for their kindness which he would not forget.

References

- [1] HUZINO, S., *Simulatability of finite automata by Scheperdson and Sturgis' machines*, Mem. Fac. Sci. Kyushu Univ. Ser. A, 20 (1966), 1-15.
- [2] SCHEPHERDSON, J. C. and STURGIS, H. E., *Computability of recursive functions*, J. ACM, 10 (1963), 217-255.
- [3] RABIN, M. O., *Real-time computation and control*, The University of Michigan Engineering Summer Conferences, Automata Theory (Advanced Concepts in Information Processing Systems), Summer, 1964.