(Journal of Computer and System Sciences,
Vol. 29, No. 3 (1984), pp. 312~329)

AN APPLICATION OF COHEN'S RESULT ON STAR HEIGHT TO THE THEORY OF CONTROL STRUCTURES

TATSUYA MOTOKI

Department of Information Science
Ibaraki University
Hitachi 316, Japan

Abstract.

We apply a result/concept on star height to two problems in the theory of control structures. First, we generalize Kosaraju's systems RE_n of control structures and show, by using Cohen's result on star height, that the generalized systems constitute a path-wise hierarchy with respect to their expressive powers. Second, by using a concept on star height, we sharpen Peterson et al.'s result that RE_∞ is path-wisely complete.

1. INTRODUCTION

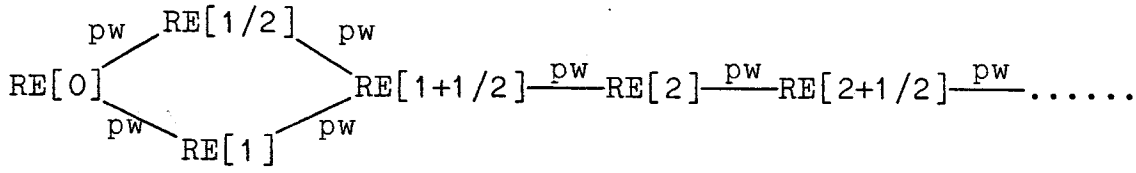
Although regular events have been extensively studied, the problem of determining the star height of a regular event is only partially solved. Among results on star height our concern is Cohen's result [2] that for a certain family of regular events the star height equals a certain measure of complexity of an automaton recognizing the event.

The theory of control structures arose from Dijkstra [3] who advocated avoiding the use of unnecessarily complex control structures in programs. Control structures have been widely studied before 1975 and many theoretical results have been presented on their expressive powers. For instance, Kosaraju [6] introduced RE[n]-structures (or RE_n-charts in terms of Kosaraju) and showed the semantic hierarchy:

$$RE[1] \overset{sem}{\longrightarrow} RE[2] \overset{sem}{\longrightarrow} RE[3] \overset{sem}{\longrightarrow} RE[4] \overset{sem}{\longrightarrow} \cdots\cdots$$

where each line means that the right system of control structures has more expressive power than the left from the semantic viewpoint. Other notable results are reviewed in Ledgard et al. [7].

Now we can easily see the language-theoretic similarity between regular expressions and program schemes. In this paper we apply Cohen's result on star height to a problem in the theory of control structures. Explicitly speaking, we generalize Kosaraju's RE[n]-structures and show the path-wise (or language-theoretic) hierarchy:



where each line means the right system of control sructures has more expressive power than the left from the language-theoretic viewpoint. In fact, the hierarchy

$$RE[1] \xrightarrow{pw} RE[2] \xrightarrow{pw} RE[3] \xrightarrow{pw} RE[4] \xrightarrow{pw} \cdots\cdots$$

follows immediately from Kosaraju's semantic hierarchy.

In addition, we sharpen Peterson et al.'s result [8] that the RE[∞]-structure is language-theoretically complete.

2. PRELIMINARIES

2.1. FLOWCHARTS AND PROGRAMS

A (deterministic) finite incomplete automaton M is a quintuple (S,A,δ,s_0,F), where S is a finite set of states, A is an alphabet, $s_0 \in S$ is the initial state, $F \subseteq S$ is the set of final states and δ is the transition partial function from $S \times A$ to S. The language recognized by M, denoted by L(M), is defined in the usual way. Let $\langle M,s \rangle$ denote the finite incomplete automaton obtained from M by changing its initial state into s. We say that M is reduced if and only if (1) it has neither any dead state nor any inaccessible state from the initial state, and (2) $s_1 \neq s_2$ implies $L(\langle M,s_1 \rangle) \neq L(\langle M,s_2 \rangle)$ for any pair of states s_1, s_2.

In the sequel, we use three pairwisely disjoint infinite

sets of symbols $\mathbb{F}$, $\mathbb{P}$ and $\{\overline{p} \mid p \in \mathbb{P}\}$; and let $\Sigma = \mathbb{F} \cup \mathbb{P} \cup \{\overline{p} \mid p \in \mathbb{P}\}$. The set $\mathbb{F}$ means a set of primitive actions (e.g. assignment statements), and the set $\mathbb{P}$ means a set of primitive predicate. The symbol of identity action, denoted by λ, is also in $\mathbb{F}$.

A flowchart is a finite incomplete automaton (S,A,δ,s_0,F) satisfying the following three conditions:

(1) A is a finite subset of Σ.

(2) For every $s \in S$, either

(2.1) $\{a \in A \mid (s,a) \text{ is defined}\} = \{p,\overline{p}\}$ for some $p \in \mathbb{P}$,

(2.2) $\{a \in A \mid (s,a) \text{ is defined}\} = \{f\}$ for some $f \in \mathbb{F}$, or

(2.3) $\{a \in A \mid (s,a) \text{ is defined}\} = \phi$.

(3) $F = \{s \in S \mid (s,a) \text{ is undefined for every } a \in A\}$.

A flowchart (S,A,δ,s_0,F) is often denoted by its state graph (S,E,s_0), where $E = \{(s,a,t) \mid \delta(s,a) = t\}$ is the set of edges.

The class of programs and the height of a program Q, denoted by h(Q), are defined as follows:

(1) Any element of $\mathbb{F}$, exit(i) for any positive integer i, and halt are programs of height 0.

(2) For any $p \in \mathbb{P} \cup \{\overline{p} \mid p \in \mathbb{P}\}$ and programs Q and R,

Q;R and if p then Q else R fi

are programs of height $\max(h(Q),h(R))$, and

while p do Q end and repeat Q end

are programs of height $h(Q)+1$.

The execution of exit(i) is intended to cause termination of i enclosing while/repeat-loops. Thus the meaning of exit(i) is different from Kosaraju's. All other commands/control structures have their usual meanings. For any nonnegative

integer n, an n-program is a program containing neither any occurrence of exit(i), i > n, nor halt, and an (n+1/2)-program is a program containing no occurrence of exit(i), i > n. The class of c-program is denoted by RE[c].

We may consider a program to be a flowchart. This is accomplished by regarding occurrences of ;, then, else and do in the program as states. We denote by G[Q] the flowchart associated with a program Q. For example, the following program Q has the associated flowchart G[Q] in Fig. 1.

```
if p then[1] λ else[2] halt fi ;[3]
while q do[4] g ;[5]
        repeat if r then[6] exit(1) else[7] f fi end
    end
```

Fig.1

Let Q' be the subprogram

repeat if r then exit(1) else f fi end

in Q. The state s_5 of G[Q] is the entry state of Q', s_3 is the exit state of Q', and s_6 and s_7 are interior states of Q'. The formal definition of these notions will be clear.

Now we define the trace set of a program Q, denoted by L(Q), to be L(G[Q]). Here the symbol of identity action is considered to be the empty word; that is $\lambda w = w\lambda = w$ for every $w \in \Sigma^*$. Thus a flowchart is actually a finite incomplete automaton with ε-moves, but these ε-moves are obviously inessential ones. We say that a flowchart G is convertible to a program Q if and only if L(G) = L(Q). We also say that a flowchart G is c-convertible if and only if G is convertible to some c-program.

Suppose that a flowchart G is convertible to a program Q.

For a subprogram R of Q, we say that a state s of G corresponds to the entry state of R if and only if L(<G,s>) = L(<G[Q], the entry state of R>). If the entry state of R is accessible from the initial state of G[Q], there exists such a state s. We define similarly for the exit state and interior states of R.

2.2. COHEN'S RESULT ON STAR HEIGHT

The star height of a regular expression over an alphabet A is a nonnegative integer defined as follows:

(1) ϕ and elements of A have star height 0.

(2) $(\alpha+\beta)$ and $(\alpha\beta)$ have star height max(star height of α, star height of β).

(3) α^* has star height (star height of α)+1.

The star height of a regular event L, denoted by sh(L), is the least star height of regular expressions denoting L.

Consider the operation over state graphs of deleting one state and related edges from each maximal strongly connected set of states. The rank of a state graph G, denoted by rank(G), is the least number of applications of the operation to break all loops in G. For example, the rank of state graph in Fig. 2 is 1, since the exhaustive search of ways of applications, shown in Fig. 3, reveals that the optimum way is to delete s_0 and s_4 at once. Likewise, the rank of state graph in Fig. 4 is 2.

Fig.2.
Fig.3.
Fig.4.

We say that a regular event L has the finite intersection property if and only if $x\backslash L \neq y\backslash L$ implies $|(x\backslash L) \cap (y\backslash L)| < \infty$ for every pair of words x, y.

THEOREM 2.1. (Cohen [2]) Suppose that a reduced finite incomplete automaton M recognizes a regular event L. If L has the finite intersection property, sh(L) = rank(M).

We now have the following corollary.

COROLLARY 2.2. If a reduced flowchart G is convertible to a program Q and L(Q) has the finite intersection property, then $h(Q) \geq \text{rank}(G)$.

Proof. We first obtain

$$\text{rank}(G[Q]) \geq \text{sh}(L(G[Q]))$$

from Eggan's result [4] that the star height of a regular event does not exceed the rank of any state graph recognizing that event. The corollary follows immediately from Theorem 2.1, the above inequality and the fact $h(Q) \geq \text{rank}(G[Q])$. Q.E.D.

Corollary 2.2, under the condition $h(Q) = 1$, is often applied to inconvertibility proofs.

3. A HIERARCHY OF CONTROL STRUCTURES

This section is the main part of the paper and is devoted to the path-wise hierarchy in Theorem 3.8.

LEMMA 3.1. There exists a flowchart which is 1-convertible but not 1/2-convertible.

Proof. Let $G=(S,E,s_0)$ denote the flowchart in Fig. 4. Then the flowchart $(S, E-\{(s_3,g_1,s_0)\}, s_4)$ is convertible to the following 1-program Q_1.

f_2 ;
repeat if p_3 then f_3 ;
if q_2 then g_2 ;.
if p_2 then f_2 else exit(1) fi
else λ fi
else g_2 ;
if p_2 then f_2 else exit(1) fi fi
end

Therefore G is convertible to the following 1-program.

while p_1 do if q_1 then Q_1 ; g_1
else if p_2 then Q_1 ; g_1
else g_1 fi fi
end

Now assume that G is convertible to a 1/2-program P. We may assume that G[P] contains no inaccessible state from the initial state. Let Q be a 1/2-program obtained from P by changing each subprogram of the form while $\overline{p_1}$ do ... end into the program while $\overline{p_1}$ do halt end or equivalently the program if p_1 then λ else halt fi. Then $L(P) = L(Q)$ since G has exactly one edge of the form $(\ldots,\overline{p_1},\ldots)$ and this edge enters to the final state s_∞. Hence G is also convertible to Q. Let Q' be a subprogram of Q of height 1 and of the form while r do ... end or repeat ... end. By the construction of Q, the subprogram Q' does not have the form while $\overline{p_1}$ do ... end.

Suppose that Q' is of the form while $\overline{p_2}$ do R end. Then

the state s_3 of G corresponds to the entry state of R and the state s_2 corresponds to the exit state of R, since G has exactly one edge of the form $(\ldots,\overline{p_2},\ldots)$. Thus, since R is a loop-free subprogram, in an automaton $(S, E-\{(s_2,\overline{p_2},s_3)\}, s_0)$ there must exist only a finite number of paths from s_3 to s_2, a contradiction. Likewise, Q' cannot have the form <u>while</u> r <u>do</u> ... <u>end</u> for any other $r \in \{p_1,p_2,p_3,\overline{p_3},q_1,\overline{q_1},q_2,\overline{q_2}\}$.

Suppose that Q' is of the form <u>repeat</u> ... <u>end</u>. Let s denote the state of G corresponding to the entry state of Q'. We may also assume $s \neq s_\infty$, because if $s = s_\infty$ we may change Q' into the program <u>halt</u> by the same reason as the case that we obtain Q from P. Then, since <u>halt</u> is the only command to cause termination of Q',

$$L(Q') = L(G[Q']) = L(\langle G,s\rangle).$$

Thus from Corollary 2.2 and the fact that $x\backslash L(\langle G,s\rangle) \neq y\backslash L(\langle G,s\rangle)$ implies $(x\backslash L(\langle G,s\rangle)) \cap (y\backslash L(\langle G,s\rangle)) = \phi$ for every pair of words x, y,

$$\mathrm{rank}(\langle G,s\rangle) \leq 1.$$

But there does not exist such s in G, a contradiction. Q.E.D.

As readers see in the proof of Lemma 3.1, Corollary 2.2 allows us to do without considering any detailed structure of programs.

LEMMA 3.2. <u>There exists a flowchart which is 1/2-convertible but not 1-convertible.</u>

Fig.5.

<u>Proof.</u> The flowchart in Fig. 5, denoted by $G=(S,E,s_0)$, is

convertible to the following 1/2-program.

```
while p1 do f1 ;
        while p2 do f2 ;
                while p3 do f3 ;
                        if p4 then halt
                              else g3 fi
                        end ;
                g2
        end ;
    g3
end
```

Now assume that G is convertible to a 1-program Q. We may assume that G[Q] contains no inaccessible state from the initial state. Let Q' be a subprogram of Q of height 1 and of the form *while* ... *do* ... *end* or *repeat* ... *end*. Let s be the state of G corresponding to the entry state of Q', s' be the state of G corresponding to the exit state of Q', and W be the set of states of G corresponding to interior states of Q'. We may also assume that $s \neq s_\infty$ and that Q' represents at least one loop, by the same reason as in the proof of Lemma 3.1. We now consider the case $s_0 \in W$, the case $s_8 \in W$ and the remaining case, and deduce a contradiction in each case.

The case $s_0 \in W$. Since $s_0 \in W$ and *exit*(1) is the only command in Q' to cause termination of Q, s' must be the final state and hence control leaving from Q' means control reaching to the terminal state. Thus we obtain

$$L(Q') = L(\langle G, s \rangle),$$

and hence from Corollary2.2

rank(<G,s>) $\leq$ 1.

But there does not exist such a state s in G, a contradiction. Likewise, the condition $s_8 \in W$ leads to a contradiction.

The case $s_0 \notin W$ and $s_8 \notin W$. The subprogram Q' represents exactly one loop $s_2 \to s_4 \to s_5 \to s_6 \to s_2$, and hence two states s_3 and s_7 of G are in W. Thus, since h(Q')=1, in the flowchart $(S, E-\{(s,a,t) \mid a \in \Sigma, t \in S\}, s_0)$ there must exist only a finite number of paths from s_3 or s_7 to s. But we cannot find out such a state s in G, a contradiction. Q.E.D.

LEMMA 3.3. There exists a flowchart which is (1+1/2)-convertible but neither 1/2-convertible nor 1-convertible.

Fig.6.

Proof. The flowchart in Fig. 6 is seen to be (1+1/2)-convertible in the same way as in the 1-convertibility proof of Lemma 3.1. From Lemma 3.1 and 3.2, the flowchart is seen to be neither 1/2-convertible nor 1-convertible. Q.E.D.

LEMMA 3.4. There exists a flowchart which is both 1/2-convertible and 1-convertible but not 0-convertible.

Fig.7.

Proof. The flowchart in Fig. 7 is convertible to the 1/2-program

while p **do** **if** q **then** **halt** **else** f **fi** **end**,

and convertible to the 1-program

while p **do** **if** q **then** **exit**(1) **else** f **fi** **end**.

The inconvertibility follows immediately from Kasai [5],

since the flowchart has a loop $s_0 \to s_1 \to s_2 \to s_0$ and this loop has two exits to the final state, $(s_0, \overline{p}, s_4)$ and (s_1, q, s_4). Q.E.D.

LEMMA 3.5. Given an (n+1/2)-program Q, where n is a positive integer, we can find out an (n+1)-program of trace set L(Q).

Proof. We give a required transformation procedure, which inserts some dummy repeat-end structures into Q. First, we explain the idea by using an example: Given a (2+1/2)-program

```
while p1 do
   while p2 do
      repeat if p3 then if p4 then exit(2) else f1 fi
                    else if p5 then halt else exit(1) fi fi
      end
   end ;
   if p6 then exit(1) else f2 fi
end ;
g,
```

the following is a required 3-program.

```
repeat
   while p1 do
      repeat
         repeat if p2 then λ else exit(2) fi ;
            repeat if p3 then if p4 then exit(3)
                                    else f1     fi
                          else if p5 then exit(2)
                                    else exit(1) fi fi
         end
```

```
            end ;
            exit(3)
         end ;
         if p6 then exit(1) else f2 fi
      end ;
      g ;
      exit(1)
   end
```

Formally, let $\{T^*, T_1, T_2, \ldots, T_n\}$ be the transformation system on programs, defined inductively as follows:

(1) $T^*(Q)$ = repeat $T_1(Q)$; exit(1) end.

(2) For every integer i, $1 \leq i \leq n$,

(2.1) if Q is in $\mathbb{F}$, then $T_i(Q) = Q$;

(2.2) if Q = exit(j), $j < i$, then $T_i(Q)$ = exit(j) ;

(2.3) if Q = exit(j), $j \geq i$, then $T_i(Q)$ = exit(j+1) ;

(2.4) if Q = halt, then $T_i(Q)$ = exit(i) ;

(2.5) if $Q = R_1;R_2$, then $T_i(Q) = T_i(R_1);T_i(R_2)$;

(2.6) if Q = if p then R_1 else R_2 fi, then

$T_i(Q)$ = if p then $T_i(R_1)$ else $T_i(R_2)$ fi ;

(2.7) if Q = repeat R end and $i < n$, then

$T_i(Q)$ = repeat $T_{i+1}(R)$ end ;

(2.8) if Q = repeat R end and $i = n$, then

$T_i(Q)$ = repeat repeat $T_1(R)$ end ; exit(n+1) end ;

(2.9) if Q = while p do R end and $i < n$, then

$T_i(Q)$ = while p do $T_{i+1}(R)$ end ; and

(2.10) if Q = while p do R end and $i = n$, then

$T_i(Q)$ = repeat repeat if p then λ else exit(2) fi ;

$T_1(R)$

```
        end ;
        exit(n+1)
    end.
```

The program T*(Q) is seen to be a required (n+1)-program. Q.E.D.

LEMMA 3.6. For every $n \in \{2,3,4,5,\ldots\}$, there exists a flowchart which is (n+1/2)-convertible but not n-convertible.

Proof. We first define flowcharts $G_n=(S_n,E_n,s(1))$, $n \in \{2,3,4,5,\ldots\}$, as follows:

(1) $S_n = \{s(i) \mid i \text{ is an integer satisfying } 1 \leq i < 2^{n+2}\}$
$\cup \{s(\infty)\} \cup \{t(i,j) \mid i \text{ and } j \text{ are integers satisfying } 2^{n+1} \leq i < 2^{n+2} \text{ and } 0 \leq j \leq n\}$.

(2) $E_n = \{(s(i),p(i),s(2^i)) \mid 1 \leq i < 2^{n+1}\}$
$\cup \{(s(i),\overline{p(i)},s(2^{i+1})) \mid 1 \leq i < 2^{n+1}\}$
$\cup \{(s(i),f(i),t(i,0)) \mid 2^{n+1} \leq i < 2^{n+2}\}$
$\cup \{(t(i,0),q(i,0),s(\infty)) \mid 2^{n+1} \leq i < 2^{n+2}\}$
$\cup \{(t(i,0),\overline{q(i,0)},t(i,1)) \mid 2^{n+1} \leq i < 2^{n+2}\}$
$\cup \{(t(i,j),q(i,j),s(\lfloor i/2^{n-j+2} \rfloor)) \mid 2^{n+1} \leq i < 2^{n+2}, 0 < j < n\}$
$\cup \{(t(i,j),\overline{q(i,j)},t(i,j+1)) \mid 2^{n+1} \leq i < 2^{n+2}, 0<j<n\}$
$\cup \{(t(i,n),q(i,n),s(\lfloor i/4 \rfloor)) \mid 2^{n+1} \leq i < 2^{n+2}\}$
$\cup \{(t(i,n),\overline{q(i,n)},s(\lfloor i/2 \rfloor)) \mid 2^{n+1} \leq i < 2^{n+2}\}$.

As an example, G_2 is shown in Fig. 8. We will show that G_n is (n+1/2)-convertible but not n-convertible.

Fig.8.

We show the (n+1/2)-convertibility by inductively constructing programs R_i, $1 \leq i < 2^{n+2}$, as follows:

(1) For each integer i, $2^{n+1} \leq i < 2^{n+2}$, let

$$R_i = \begin{cases} f(i) \ ; \\ \underline{\text{if}}\ q(i,0)\ \underline{\text{then}}\ \underline{\text{halt}}\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}\ ; \\ \underline{\text{if}}\ q(i,1)\ \underline{\text{then}}\ \underline{\text{exit}}(n)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}\ ; \\ \underline{\text{if}}\ q(i,2)\ \underline{\text{then}}\ \underline{\text{exit}}(n-1)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}\ ; \\ \qquad\qquad \ldots\ldots\ldots\ldots \\ \underline{\text{if}}\ q(i,n)\ \underline{\text{then}}\ \underline{\text{exit}}(1)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}. \end{cases}$$

(2) For each integer i, $1 \leq i < 2^{n+1}$, let

R_i = repeat if p(i) then R_{2i} else R_{2i+1} fi end.

The flowchart G_n is easily seen to be convertible to the (n+1/2)-program R_1.

Now assume that G_n is convertible to an n-program Q. We may assume that G[Q] contains no inaccessible state from the initial state. Let Q_1 be a subprogram of Q of height 1 and of the form while ... do ... end or repeat ... end; and for each integer $i \geq 2$, let Q_i be the smallest subprogram containing Q_{i-1} properly and of the form while ... do ... end or repeat ... end. Let m be the greatest number i such that Q_i exists, and let S be the set of states of G_n corresponding to exit states of Q_1, Q_2, ..., $Q_{\min(m,n)}$. Because of the symmetry in G_n, Q_1 may be assumed to represent a loop passing through $s(2^{n+1})$. Finally, we partition S_n as follows:

(1) $U_{-1} = \{s(\infty)\}$.

(2) For each integer k, $0 \leq k \leq n-2$,

$$U_k = \{s(2^k)\} \cup \{s(i) \mid (2^{k+1}+1)2^j \leq i < (2^{k+1}+2)2^j,\ 0 \leq j \leq n-k\} \cup \{t(i,j) \mid (2^{k+1}+1)2^{n-k} \leq i < (2^{k+1}+2)2^{n-k},\ 0 \leq j \leq n\}.$$

(3) $U_{n-1} = S_n - \cup_{k=-1}^{n-2} U_k$.

The partition of S_3 is illustrated in Fig. 9.

Fig.9.

We shall prove that $S \cap U_j \neq \phi$ for every j, $-1 \leq j \leq n-1$.

Let

$$k = \min\{i \mid Q_1 \text{ represents a loop passing through } s(2^i), s(2^{i+1}), s(2^{i+2}), \ldots, \text{ and } s(2^{n+1})\}.$$

Then not only for every $j \in \{k, k+1, \ldots, n-1\}$ but also for every $j \in \{-1, 0, 1, \ldots, k-1\}$ the state $s(2^j)$ corresponds to an entry or interior state of Q_1, since Q_1 represents a loop of the form

$$s(2^k) \to \ldots \to s(2^{k+1}) \to \ldots \to s(2^{n+1}) \to t(2^{n+1},0) \to t(2^{n+1},1) \to \ldots \to t(2^{n+1},k+1) \to s(2^k),$$

and since $(t(2^{n+1},0),q(2^{n+1},0),s(\infty))$, $(t(2^{n+1},1),q(2^{n+1},1),s(1))$, $(t(2^{n+1},2),q(2^{n+1},2),s(2)),\ldots,$ $(t(2^{n+1},k),q(2^{n+1},k),s(2^{k-1}))$ are edges of G_n. Thus

$$S \cap U_{-1} \neq \phi$$

since Q is an n-program and hence either exit(1), exit(2),..., exit(n-1) or exit(n) must be used in Q_1 to represent an edge to the final state, $(t(2^{n+1},0),q(2^{n+1},0),s(\infty))$. And

$$S \cap U_j \neq \phi \quad \text{for every } j \in \{0,1,2,\ldots, n-1\}$$

since Q_1 cannot represent a whole part of U_j from Corollary 2.2, $h(Q_1) = 1$ and $\text{rank}((U_j, E_n \cap (U_j \times \Sigma \times U_j), s(2^j))) \geq 2$.

From the above inequality we derive a contradiction,

$$n+1 = \sum_{j=-1}^{n-1} 1 \leq \sum_{j=-1}^{n-1} |S \cap U_j| = |S| \leq \min(m,n) \leq n. \quad \text{Q.E.D.}$$

LEMMA 3.7. For every positive integer n, there exists a flowchart which is (n+1)-convertible but not (n+1/2)-convertible.

Proof. We first define flowcharts $H_n=(S_n,E_n,s(0))$, $n \in \{1,2,3,\ldots.\}$, as follows:

(1) $S_n = \{s(i) \mid i \text{ is an integer satisfying } 0 \le i < 2^{n+3}\}$
$\cup \{s(\infty)\} \cup \{t(i,j) \mid i \text{ and } j \text{ are integers satisfying } 2^{n+2} \le i < 2^{n+3} \text{ and } 0 \le j \le n\}$.

(2) $E_n = \{(s(0),p(0),s(1)), (s(0),\overline{p(0)},s(\infty))\}$
$\cup \{(s(i),p(i),s(2^i)) \mid 1 \le i < 2^{n+2}\}$
$\cup \{(s(i),\overline{p(i)},s(2^{i+1})) \mid 1 \le i < 2^{n+2}\}$
$\cup \{(s(i),f(i),t(i,0)) \mid 2^{n+2} \le i < 2^{n+3}\}$
$\cup \{(t(i,0),q(i,0),s(0)) \mid 2^{n+2} \le i < 2^{n+3}\}$
$\cup \{(t(i,0),\overline{q(i,0)},t(i,1) \mid 2^{n+2} \le i < 2^{n+3}\}$
$\cup \{(t(i,j),q(i,j),s(\lfloor i/2^{n-j+2} \rfloor)) \mid 2^{n+2} \le i < 2^{n+3}, 0 < j < n\}$
$\cup \{(t(i,j),\overline{q(i,j)},t(i,j+1)) \mid 2^{n+2} \le i < 2^{n+3}, 0<j<n\}$
$\cup \{(t(i,n),q(i,n),s(\lfloor i/4 \rfloor)) \mid 2^{n+2} \le i < 2^{n+3}\}$
$\cup \{(t(i,n),\overline{q(i,n)},s(\lfloor i/2 \rfloor)) \mid 2^{n+2} \le i < 2^{n+3}\}$.

As an example, H_1 is shown in Fig. 10. The flowchart H_n is similar to G_{n+1} in the proof of Lemma 3.6. We will show that H_n is (n+1)-convertible but not (n+1/2)-convertible.

Fig.10.

We show the (n+1)-convertibility by inductively constructing programs R_i, $1 \le i < 2^{n+3}$, as follows:

(1) For each integer i, $2^{n+2} \le i < 2^{n+3}$, let

$$R_i = \begin{cases} f(i)\ ; \\ \underline{\text{if}}\ q(i,0)\ \underline{\text{then}}\ \underline{\text{exit}}(n+1)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}\ ; \\ \underline{\text{if}}\ q(i,1)\ \underline{\text{then}}\ \underline{\text{exit}}(n)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}\ ; \\ \quad\quad \ldots\ldots\ldots\ldots \\ \underline{\text{if}}\ q(i,n)\ \underline{\text{then}}\ \underline{\text{exit}}(1)\ \underline{\text{else}}\ \lambda\ \underline{\text{fi}}. \end{cases}$$

(2) For each integer i, $2 \le i < 2^{n+2}$, let

R_i = repeat if p(i) then R_{2i} else R_{2i+1} fi end.

(3) Let

R_1 = **while** p(0) **do if** p(1) **then** R_2 **else** R_3 **fi end**.

The flowchart H_n is easily seen to be convertible to the (n+1)-program R_1.

Now assume that H_n is convertible to an (n+1/2)-program Q. We may assume that G[Q] contains no inaccessible state from the initial state. Let Q_i, m and S be the same as defined in the proof of Lemma 3.6. Because of the symmetry in H_n, Q_1 may be assumed to represent a loop passing through $s(2^{n+2})$. And we partition S_n as follows:

(1) $U_0 = \{s(0), s(1)\} \cup \{s(i) \mid 3\times 2^j \le i < 4\times 2^j,\ 0 \le j \le n+1\}$
$\cup \{t(i,j) \mid 3\times 2^{n+1} \le i < 4\times 2^{n+1},\ 0 \le j \le n\}$.

(2) For each integer k, $1 \le k < n$,

$U_k = \{s(2^k)\}$
$\cup \{s(i) \mid (2^{k+1}+1)2^j \le i < (2^{k+1}+2)2^j,\ 0 \le j \le n-k+1\}$
$\cup \{t(i,j) \mid (2^{k+1}+1)2^{n-k+1} \le i < (2^{k+1}+2)2^{n-k+1},$
$0 \le j \le n\}$.

(3) $Un = S_n - \{s(\infty)\} - \cup_{k=0}^{n-1} U_k$.

The partition of S_2 is illustrated in Fig. 11.

Fig.11

Then we obtain

$S \cap U_j \neq \phi$ for every $j \in \{0,1,2,\ldots,n\}$

in the same manner as in the proof of Lemma 3.6. Thus

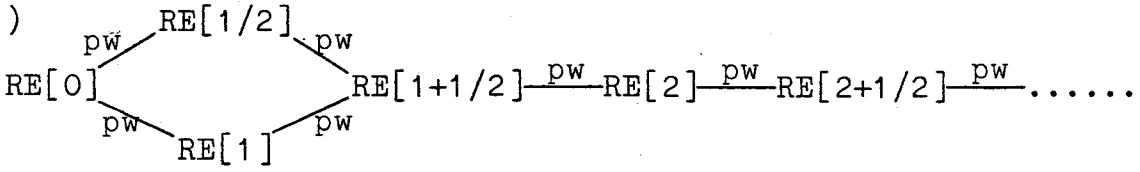
$n+1 = \sum_{j=0}^{n} 1 \le \sum_{j=0}^{n} |S \cap U_j| \le |S| \le \min(m,n) \le n$,

a contradiction. Q.E.D.

From above seven lemmas we obtain the following result.

THEOREM 3.8.

(1)



where each line from RE[c] to RE[d] means

$$L(RE[c]) \subsetneq L(RE[d])$$

i.e. $\{L(Q) \mid Q \in RE[c]\} \subsetneq \{L(Q) \mid Q \in RE[d]\}$.

(2) $L(RE[1]) - L(RE[1/2]) \neq \phi$, $L(RE[1/2]) - L(RE[1]) \neq \phi$,

$L(RE[1+1/2]) - (L(RE[1/2]) \cup L(RE[1])) \neq \phi$,

$(L(RE[1/2]) \cap L(RE[1])) - L(RE[0]) \neq \phi$.

Incidentally, Peterson et al. [8] shows that

$L(RE[\infty]) = \{L(G) \mid G \text{ is a flowchart}\}$.

This can be sharpened by considering ranks of flowcharts. The proof is similar to the proof of Eggan's result [4] that given an incomplete automaton M, we can find out a regular expression of star height rank(M) and of language L(M).

THEOREM 3.9. Let $G=(S,E,s_0)$ be a flowchart. If rank(G) = 0, then we can construct a 0-program of trace set L(G) by using only semicolon and if-then-else-fi structures; if rank(G) = 1, then $L(G) \in L(RE[1/2]) \cap L(RE[1])$; and if rank(G) > 1, then $L(G) \in L(RE[(rank(G)-1)+1/2])$.

Proof. An i-section is defined to be a maximal strongly connected set T of states such that $rank((S,(T\times\Sigma\times T) \cap E,s_0)) = i$. Let n(i,G) be the number of i-sections in G, and let

$$k(G) = \sum_{i=1}^{rank(G)} n(i,G)|S|^i.$$

The proof proceeds by induction on k(G).

Basis, k(G) = 0. Since rank(G)=0 and hence G contains no loop, we can easily construct a required 0-program.

Induction step, k(G) > 0. Let K be the class of rank(G)-sections, and for any $T \in K$ let s(T) be a state in T such that

$$\mathrm{rank}((S, ((T-\{s(T)\})\times\Sigma\times(T-\{s(T)\})) \cap E, s_0)) = \mathrm{rank}(G)-1.$$

We gradually construct a required program.

In the first step we construct a required program Q[T] for each flowchart $\langle G, s(T)\rangle$, $T \in K$. Let

$$EXIT[T] = \{s \in S-T \mid (T\times\Sigma\times\{s\}) \cap E \neq \phi\},$$

and let

$$G[T] = (S, (E - (EXIT[T] \cup \{s(T)\})\times\Sigma\times S) \cup \{(t,f(t),s(T)) \mid t \in EXIT[T]\}, s_0)$$

where f(t)'s are new symbols in $\mathbb{F}$. The flowchart G[T] is illustrated in Fig. 12.

(1) The case that rank(G) > 1 and $(s(T),g,s_1) \in E$ for some $s_1 \in S$ and $g \in \mathbb{F}$. Fig.12. From the inductive hypothesis and Lemma 3.5, we can construct (rank(G)-1)-programs $Q[s_1]$ of trace set $L(\langle G[T],s_1\rangle)$. From the inductive hypothesis and the fact that for any $t \in EXIT[T]$ the flowchart $\langle G,t\rangle$ can be reduced into a flowchart G' satisfying $k(G') < k(G)$, for each $t \in EXIT[T]$ we can also construct a ((rank(G)-1)+1/2)-program R[t] of trace set $L(\langle G,t\rangle)$. Then

repeat g ; $Q[s_1]\{(R[t]$; halt$)/f(t)\}_{t \in EXIT[T]}$ end

is a required ((rank(G)-1)+1/2)-program of trace set $L(\langle G,s(T)\rangle)$ where $Q[s_1]\{(R[t]$; halt$)/f(t)\}_{t \in EXIT[T]}$ is a program obtained from $Q[s_1]$ by replacing each occurrence of f(t), $t \in EXIT[T]$, by

the program R[t];halt.

(2) The case that rank(G) > 1 and $(s(T),p,s_1)$, $(s(T),\bar{p},s_2) \in E$ for some s_1, $s_2 \in S$ and $p \in \mathbb{P}$. We can construct a required program in the same manner as in the case (1). In fact,

repeat if p then $Q[s_1]\{(R\{t];\underline{halt})/f(t)\}_{t \in EXIT[T]}$
else $Q[s_2]\{(R[t];\underline{halt})/f(t)\}_{t \in EXIT[T]}$ fi
end

is a required program where $Q[s_2]$ is a (rank(G)-1)-program of trace set $L(\langle G[T],s_2\rangle)$.

(3) The case that rank(G) = 1 and $(s(T),g,s_1) \in E$ for some $s_1 \in S$ and $g \in \mathbb{F}$. From the inductive hypothesis we can construct a 0-program $Q[s_1]$ of trace set $L(\langle G[T],s_1\rangle)$ by using only semicolon and if-then-else-fi structures. From the inductive hypothesis and the same fact as in the case (1), for each $t \in EXIT[T]$ we can also construct a 1/2-program P[t] and a 1-program R[t] of trace set $L(\langle G,t\rangle)$. Then

repeat g ; $Q[s_1]\{(P[t];\underline{halt})/f(t)\}_{t \in EXIT[T]}$ end

is a required 1/2-program of trace set $L(\langle G,s(T)\rangle)$; and

repeat g ; $Q[s_1]\{(R[t];\underline{exit}(1))/f(t)\}_{t \in EXIT[T]}$ end

is a required 1-program of trace set $L(\langle G,s(T)\rangle)$ since $Q[s_1]$ has no while/repeat-loop.

(4) The case that rank(G) = 1 and $(s(T),p,s_1)$, $(s(T),\bar{p},s_2) \in E$ for some s_1, $s_2 \in S$ and $p \in \mathbb{P}$. We can construct required programs in the same manner as in the case (3).

Fig.13.

In the second step we construct a required program for G. Let

$$H = (S \cup \{t\},\ (E - \{s(T) \mid T \in K\} \times \Sigma \times S) \cup \{(s(T),g(T),t) \mid T \in K\},\ s_0)$$

where t is a new state and g(T)'s are new symbols in $\mathbb{F}$. The flowchart H is illustrated in Fig. 13. From inductive hypothesis and the fact that H can be reduced into a flowchart H' satisfying k(H') < k(H), we can construct a ((rank(G)-2)+1/2)-program R of trace set L(H). Then

$$R\{Q[T]/g(T)\}_{T \in K}$$

is a required ((rank(G)-1)+1/2)-program of trace set L(G) where Q[T]'s are ((rank(G)-1)+1/2)-programs obtained in the first step. Q.E.D.

ACKNOWLEDGMENTS

I wish to thank Professor Teruyasu Nishizawa for introducing me to the area, and the referee and Professor Kojiro Kobayashi for reading carefully the manuscript. They also gave me many valuable suggestions/advices.

REFERENCES

1. R. S. Cohen and J. A. Brzozowski, General properties of star height of regular events, J. Comput. System Sci. 4 (1970), 260-280.
2. R. S. Cohen, Star height of certain families of regular events, J. Comput. System Sci. 4 (1970), 281-297.
3. E. W. Dijkstra, GoTo statement considered harmful, Comm. ACM 11 (1968), 147-148.

4. R. C. Eggan, Transition graphs and the star height of regular events, Michigan Math. J. 10 (1963), 385-397.

5. T. Kasai, Translatability of flowcharts into while programs, J. Comput. System Sci. 9 (1974), 177-195.

6. S. R. Kosaraju, Analysis of structured programs, J. Comput. System Sci. 9 (1974), 232-255.

7. H. F. Ledgard and M. Marcotty, A genelogy of control structures, Comm. ACM 18 (1975), 629-639.

8. W. W. Peterson, T. Kasami and N. Tokura, On the capabilities of While, Repeat, and Exit statements, Comm. ACM 16 (1973), 503-512.

Descriptive legends for figures.

Fig.1. An example of G[Q].

Fig.2. A state graph of rank 1.

Fig.3. An explanation of the fact that the rank of the state graph in Fig.2 is 1.

Fig.4. A flowchart of rank 2 in Lemma3.1.

Fig.5. A flowchart in Lemma3.2.

Fig.6. A flowchart in Lemma3.3.

Fig.7. A flowchart in Lemma3.4.

Fig.8. A flowchart G_2 in the proof of Lemma3.6.

Fig.9. The partition of S_3 in the proof of Lemma3.6.

Fig.10. A flowchart H_1 in the proof of Lemma3.7.

Fig.11. The partition of S_2 in the proof of Lemma3.7.

Fig.12. A flowchart G[T] in the proof of Theorem3.9.

Fig.13. A flowchart H in the proof of Theorem3.9.

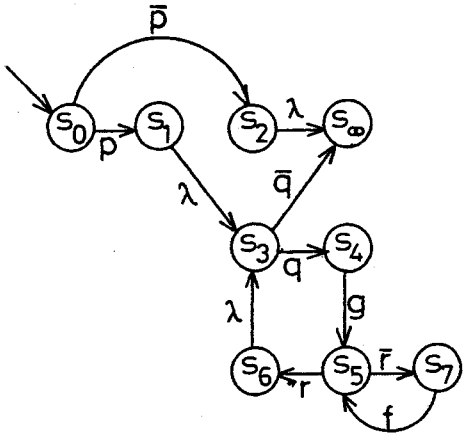


Fig.1.

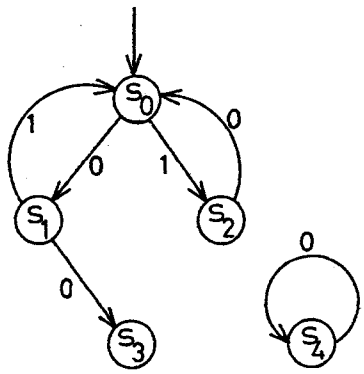


Fig.2.

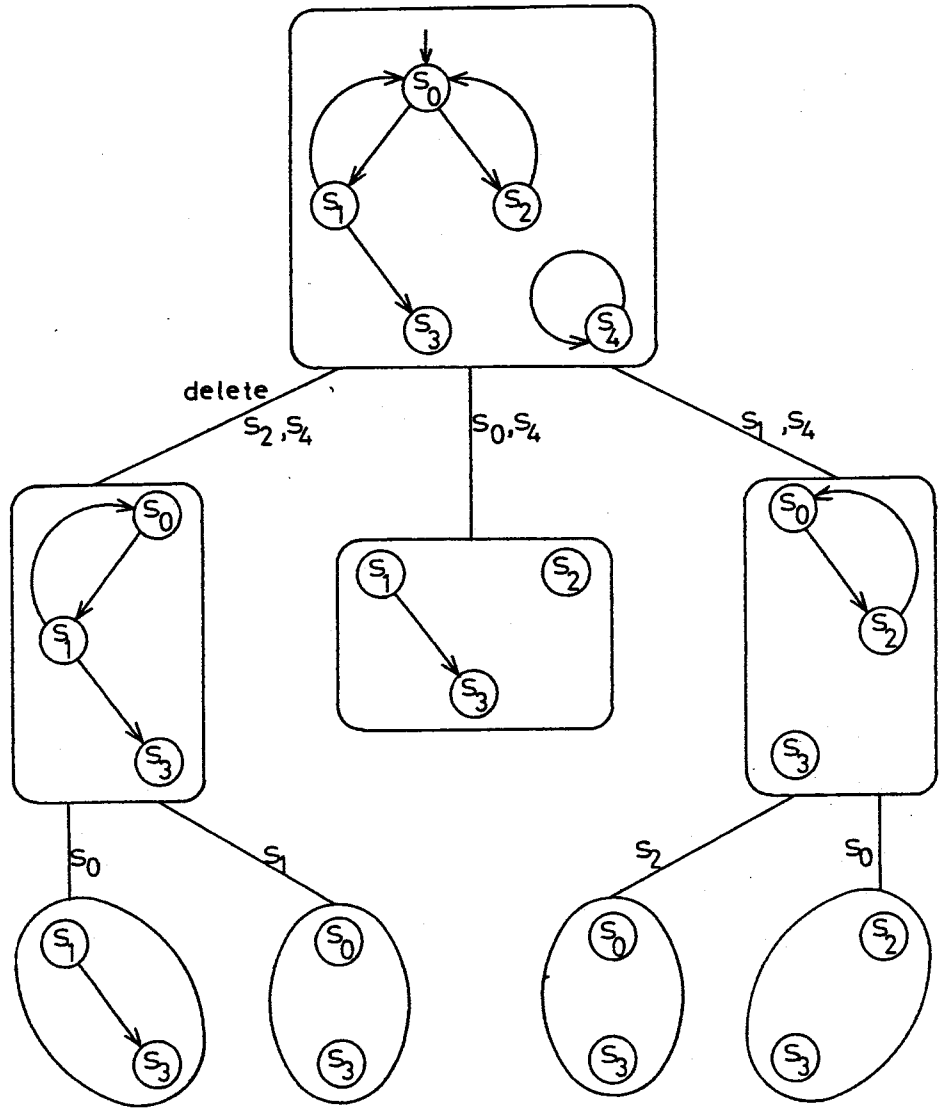

delete
S_2, S_4
S_0, S_4
S_1, S_4
S_0
S_1
S_2
S_3
S_4
S_0
S_1
S_2
S_3
S_0
S_1
S_2
S_3
S_0
S_1
S_0
S_3
S_2
S_0
S_0
S_3
S_2
S_3

Fig.3.

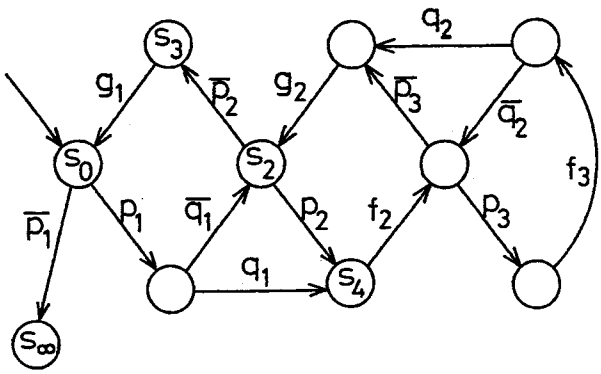

s_3
g_1
$\overline{p}_2$
g_2
q_2
$\overline{p}_3$
$\overline{q}_2$
s_0
s_2
f_3
$\overline{p}_1$
p_1
$\overline{q}_1$
p_2
f_2
p_3
q_1
s_4
s_∞

Fig.4.

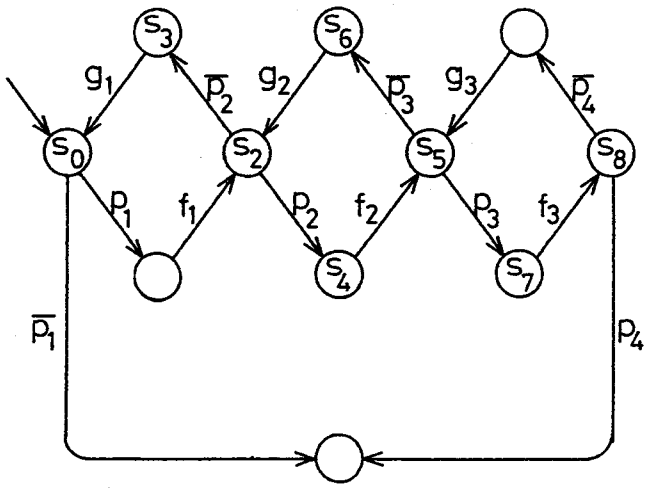

s_3
s_6
g_1
$\overline{p}_2$
g_2
$\overline{p}_3$
g_3
$\overline{p}_4$
s_0
s_2
s_5
s_8
p_1
f_1
p_2
f_2
p_3
f_3
s_4
s_7
$\overline{p}_1$
p_4

Fig.5.

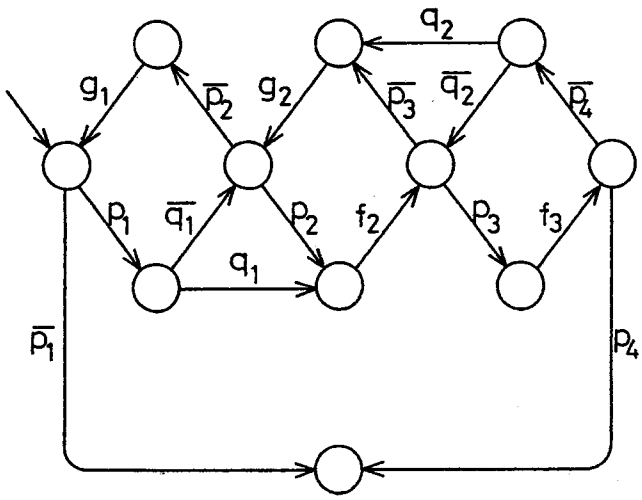

q_2
g_1
$\overline{p_2}$
g_2
$\overline{p_3}$
$\overline{q_2}$
$\overline{p_4}$
p_1
$\overline{q_1}$
p_2
f_2
p_3
f_3
q_1
$\overline{p_1}$
p_4

Fig.6.

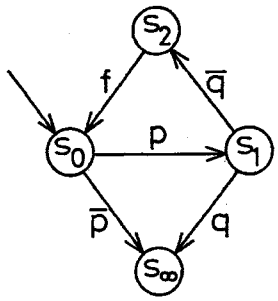


Fig.7.

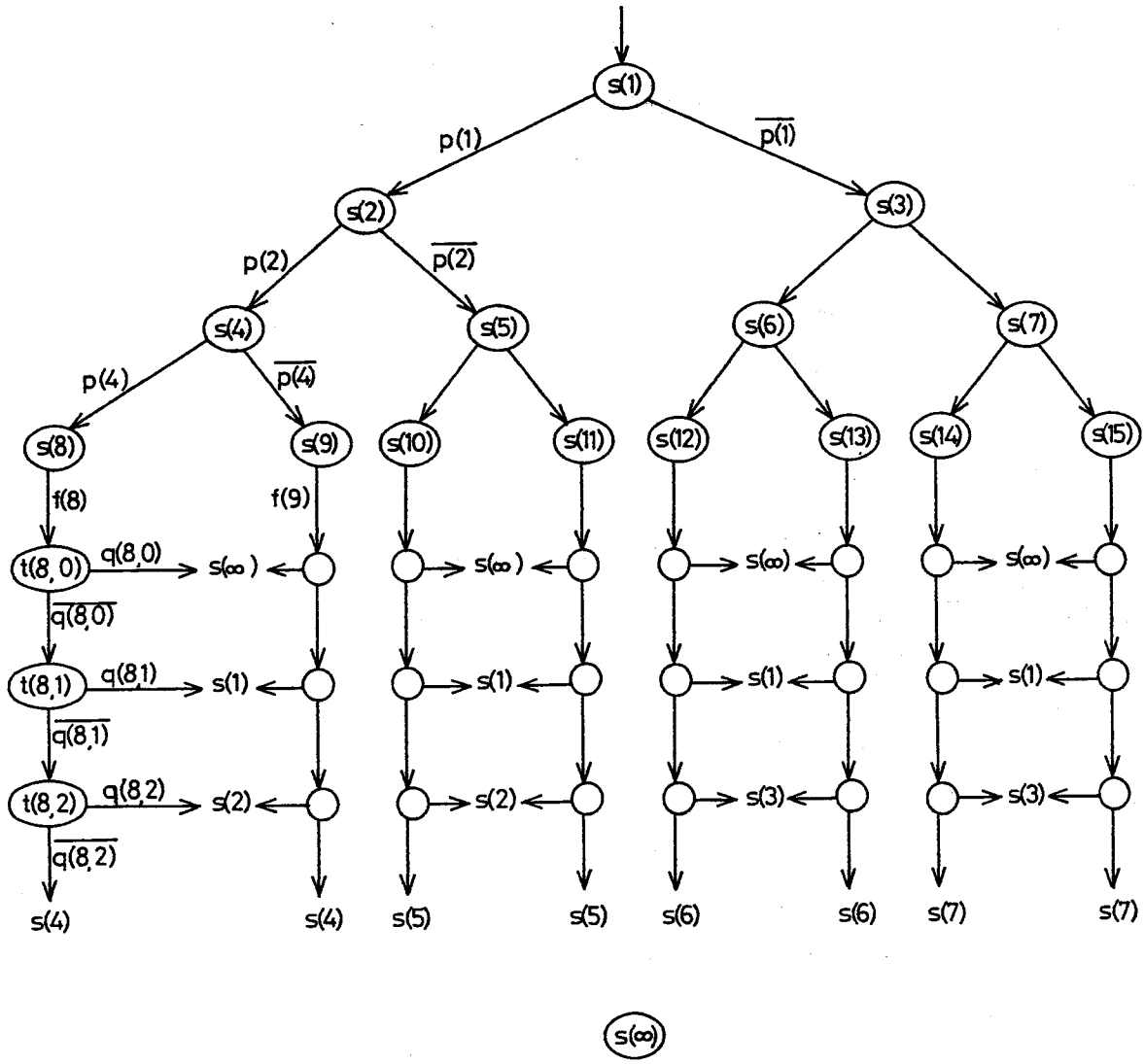

s(1)
p(1)
p(1)
s(2)
s(3)
p(2)
p(2)
s(4)
s(5)
s(6)
s(7)
p(4)
p(4)
s(8)
s(9)
s(10)
s(11)
s(12)
s(13)
s(14)
s(15)
f(8)
f(9)
t(8,0)
q(8,0)
s(∞)
q(8,0)
t(8,1)
q(8,1)
s(1)
q(8,1)
t(8,2)
q(8,2)
s(2)
q(8,2)
s(4)
s(4)
s(5)
s(5)
s(6)
s(6)
s(7)
s(7)
s(∞)

Fig.8.

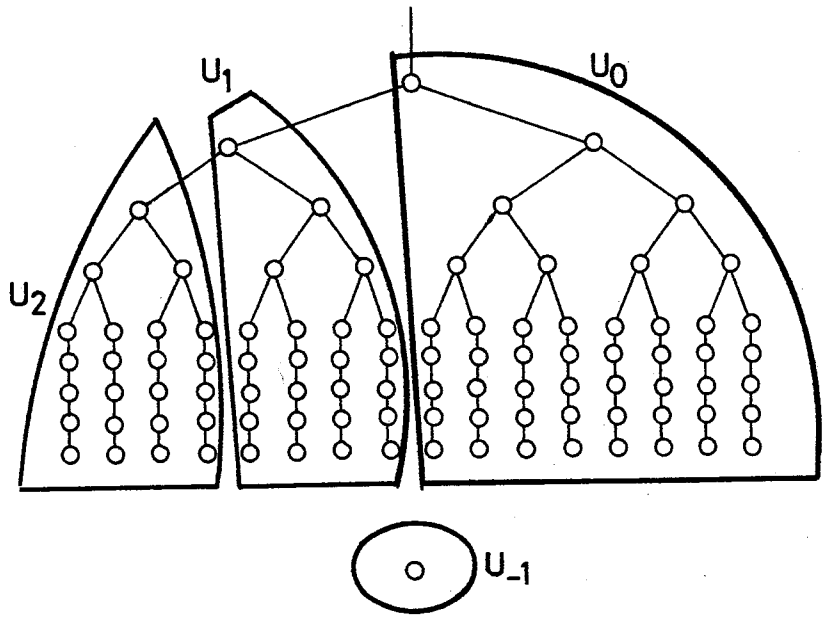


Fig.9.

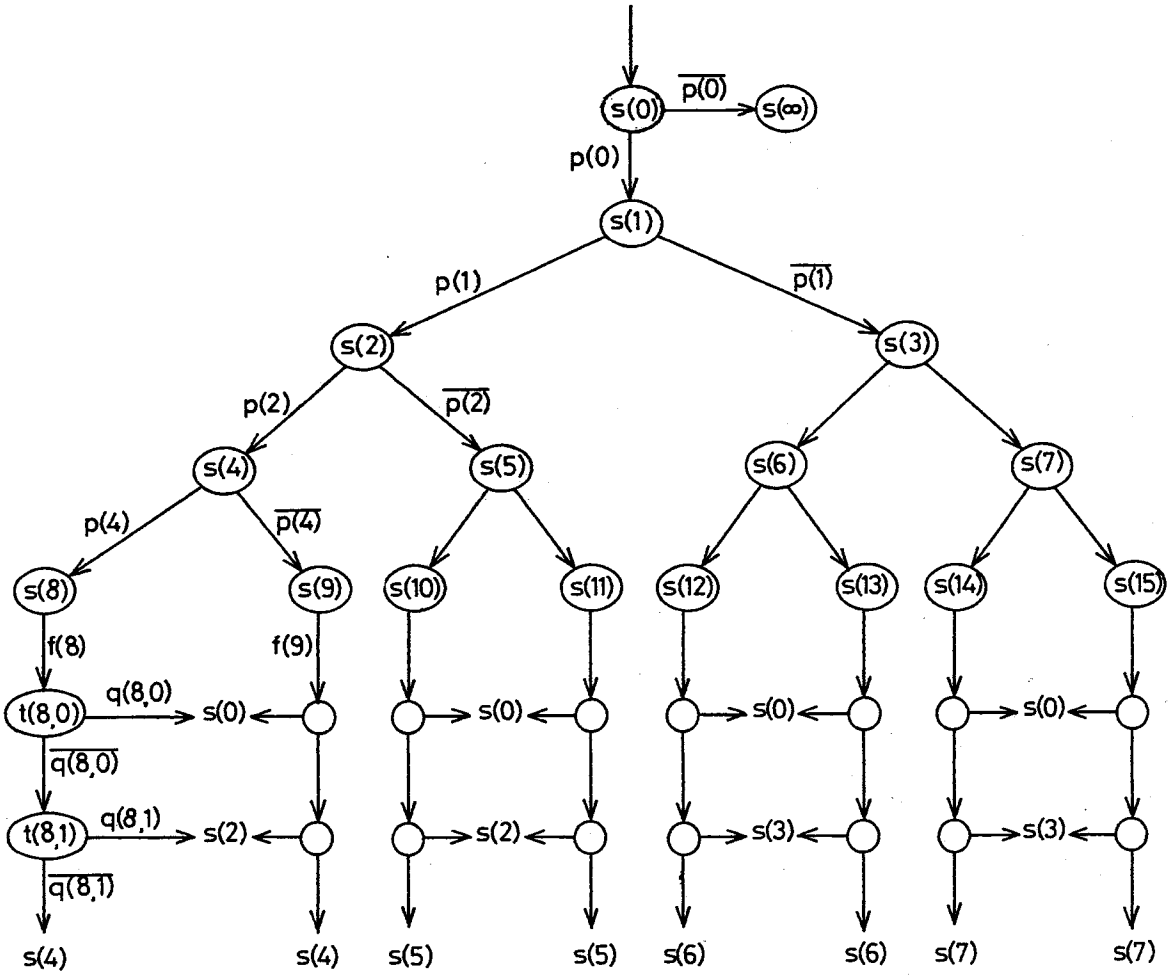

s(0)
p(0)
s(∞)
s(1)
p(1)
p(0)
s(2)
s(3)
p(2)
p(1)
s(4)
s(5)
s(6)
s(7)
p(4)
p(2)
s(8)
s(9)
s(10)
s(11)
s(12)
s(13)
s(14)
s(15)
f(8)
f(9)
t(8,0)
q(8,0)
s(0)
t(8,1)
q(8,1)
s(2)
s(3)
s(4)
s(5)
s(6)
s(7)

Fig.10.

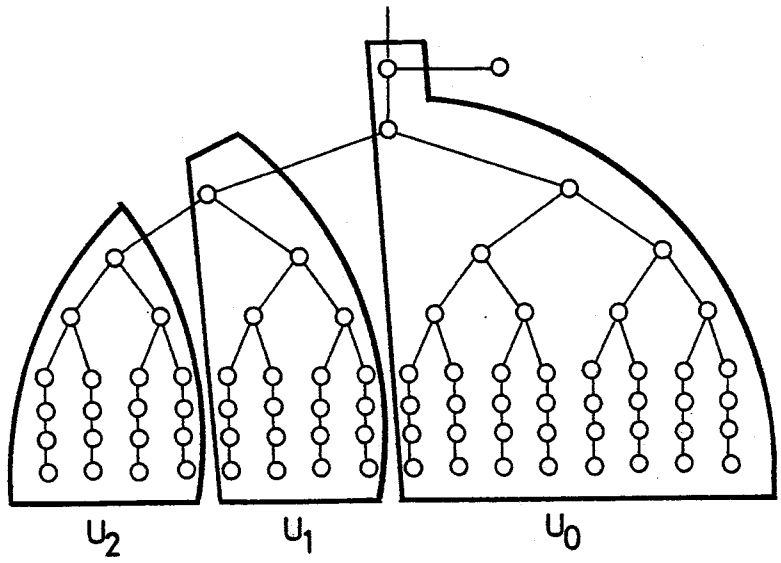


Fig.11.

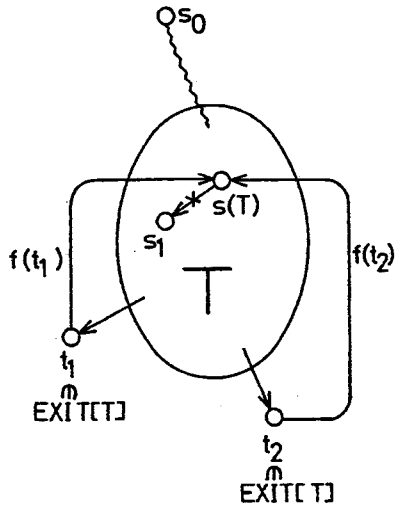


Fig.12.

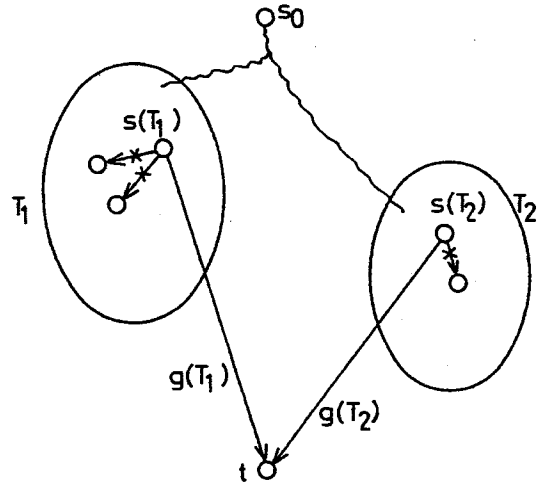


Fig.13.