数理解析研究所講究録
119巻 1971年 112-132

プログラムの正当性を機械的に証明する一方法

東大 工学部　鈴木則久

§1 序論

プログラムは、プログラムの入力状態としてある関係がある、すなわちある命題が成りたつとき、出力状態が欲する関係にある、すなわち予想した命題がなりたつときに正当であるという。ここでの問題は出力状態の命題を求める方法を発見することである。またそのような命題を求める事できるか決定する、すなわちプログラムが停止するかどうか決定するアルゴリズムを求めなければならない。しかし一般的に停止問題を決定するアルゴリズムは存在しないことはすでに知られているので、ここでは停止するかどうか決定できるための条件を与え、またその条件のもとでの出力命題を求めるアルゴリズムを与える。

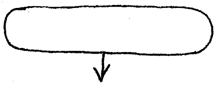
この問題はまずFloyd (1)が、述語をflow diagramに対応させることによって正当性と停止の問題をのべたが、

一般の問題を簡単に扱えるわけではなかった。 Manna (2)はこの方向を理論的に進めて、第一階述語論理でプログラムを記述し、プログラムの停止問題と述語論理式の決定問題を対応させ、決定可能な wff から、停止するプログラムの組を導いた。 Engeler (3)はプログラムの各命令に番号をつけることによって、プログラムのあらゆる可能な制御の流れを regular expression で表わし、regular expression に infinitary logic を対応する方法を示すことによってプログラムを記述した。 これによってある入力変数に対してプログラムが停止することと、対応する述語が決定可能であることは同値であることを言ったが、infinitary logic の決定問題は一般に可解でないので、問題を一般的に扱うのは困難であった。

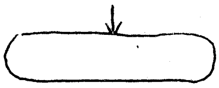
§2 Flow diagram に述語を対応させる

Flow diagram は次の5つの要素(1)から構成され、Start と Halt は1つだけあるものである。

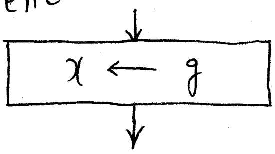
① Start



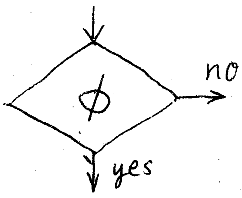
② Halt



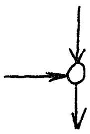
③ Assignment



④ Condition



⑤ Join



Flow diagram を上の5つの node と制御の流れを表わす arc で記述する。 各 arc に述語を対応させていく。そのときの述語には次の様な条件を加える。

ある node から出る arc が n本あり、これに対応する述語をそれぞれ $P_1, \cdots, P_n$ とすると、

$\forall_{\substack{1\leq i,j\leq n \\ i\neq j}} (\bar{x})(P_i \wedge P_j)$ はなりたたない。

$(\exists \bar{x})\ \bigvee_{i=1}^{n} P_i$ はなりたつ。

ただし、$\bar{x}$ はプログラムにあらわれる変数の集合を表わす。

また各nodeには、入力arcに付いている述語の組から、出力arcに付いている述語の組への写像を対応させる。入力arcに付いている述語の組を$\mathbb{P}$、出力arcに付いている述語の組を$\mathbb{Q}$とすると、

$$\mathbb{Q} = \mathbb{f}(\mathbb{P}) \tag{2.1}$$

あるnodeへの入力がn arc、出力がm arcとすると、

$$\begin{aligned} Q_1 &= f_1(P_1, \cdots, P_n) \\ Q_2 &= f_2(P_1, \cdots, P_n) \\ &\vdots \\ Q_m &= f_m(P_1, \cdots, P_n) \end{aligned} \tag{2.2}$$

この写像は$P_1, \cdots, P_n$のどれかが真ならば（すなわちそこからこのnodeに入る）、$Q_1, \cdots, Q_m$のどれか1つが必ず真になるように選ぶ。ところが直列処理のプログラムを考えると、i番目の入力arcからnodeにはいったときは、その時の状態、即ちP_iによってどのarcから出るかは決定されるので、(2.2)は次のように変換できる。

$$\begin{aligned} Q_1 &= Q_{11} \vee Q_{12} \vee \cdots \vee Q_{1n} \\ &\vdots \\ Q_m &= Q_{m1} \vee Q_{m2} \vee \cdots \vee Q_{mn} \end{aligned} \tag{2.3}$$

$$Q_{ij} = f_{ij}\ (P_j) \tag{2.4}$$

matrix で表わすと、(2.1) は次の様になる。

$$\begin{pmatrix} Q_{11} & \cdots & Q_{1n} \\ \vdots & & \vdots \\ Q_{m1} & \cdots & Q_{mn} \end{pmatrix} = \begin{pmatrix} f_{11}(P_1) & \cdots & f_{1n}(P_n) \\ \vdots & & \vdots \\ f_{m1}(P_1) & \cdots & f_{mn}(P_n) \end{pmatrix} \tag{2.5}$$

Floyd 流の述語で ① から ⑤ までに写像を対応すると次の様になる。

① Start

$$Q_1 = f_1(T) \tag{2.6}$$

② Halt

$$Q_1 = f_1(P_1) = T \tag{2.7}$$

③ Assignment

$$Q_1 = f_1(P_1) = (\exists x_0)(S^x_{x_0}(P_1) \wedge x = S^x_{x_0}(g)) \tag{2.8}$$

④ Condition

$$Q_1 = f_1(P_1) = P_1 \wedge \phi$$
$$Q_2 = f_2(P_1) = P_1 \wedge \neg\phi \tag{2.9}$$

⑤ Join

$$Q_1 = Q_{11} \vee Q_{12}$$

$$Q_{ii} = f_{ii}(P_i) = P_i \tag{2.10}$$

しかしこの定義では substitution を含んでいて、入力の述語を変えてしまうので、以後の展開に不便なので、次の様に再定義する。

Assignment の左辺の変数の値は、assignment の前後では値が変わるので、superfix をつけて区別すると、

$$Q_1(x^{(j)}) = f_1(P_1(x^{(j)})) = (P_1(x^{(j)}) \wedge x^{(j+1)} = g(x^{(j)})) \tag{2.11}$$

これによってすべての写像は入力の述語を変化させることなく、Q matrix の各要素は、入力述語にある述語を ∧ で結合したものとして表わせる。新しく $\bigwedge$ という operator を次の様に定義する。

$$\begin{pmatrix} Q_{11} \wedge P_1 & \cdots & Q_{1n} \wedge P_n \\ \vdots & & \vdots \\ Q_{m1} \wedge P_1 & \cdots & Q_{mn} \wedge P_n \end{pmatrix} \equiv \begin{bmatrix} Q_{11} & \cdots & Q_{1n} \\ \vdots & & \vdots \\ Q_{m1} & \cdots & Q_{mn} \end{bmatrix} \bigwedge \begin{pmatrix} P_1 \\ \vdots \\ P_n \end{pmatrix} \tag{2.12}$$

すると (2.5) は次の様に書き変えられる。

$$\begin{bmatrix} Q_{11} & \cdots & Q_{1n} \\ \vdots & & \vdots \\ Q_{m1} & \cdots & Q_{mn} \end{bmatrix} = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \wedge \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix} \tag{2.13}$$

実際にプログラムが実行されるときは、start node から、そこを出る arc を通り、次の node に入り、そこからまた arc を通って次の node に行く操作が操り返される。これに応じて、入力の述語が変換されていくので、述語の列が、1つの入力変数の組に対して決定される。これらは Q 行列の1つの元であるから、全部 atomic formula を ∧ でつなげた formula である。

§3 プログラムをグラフとして見る

Flow diagram を node と arc からなるものとみると、これは directed graph となる。

定義1. Subgraph G_1 が strong component of a graph G ということは、次の2つの条件を満足することである。

1. G_1 は strongly connected (G_1 の任意の2つの node の順序

組の間には path が存在する。)

2. $\forall G_2 \subseteq G$ such that $G_1 \subset G_2$ に対して、G_2 は strongly connected ではない。

定義2. α が isolated arc of a graph G ということは次の2つの条件を満足することである。

1. $\alpha \in A$ (G の arc の集合)

2. α は strongly connected component の一部ではない。

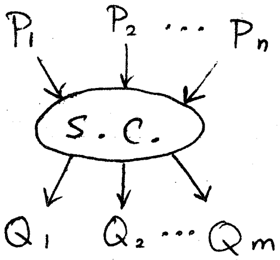
定理1. プログラムで制御が2度以上通る可能性のある arc は strong component の arc に対応する。又逆もなりたつ。

定理2. Strong component への入口は join であり、出口は condition である。

Strong component を1つの node で表わすと、入力が1つの場合、その入力 arc での述語を P とすると、出力 arc での述語も $P \wedge A$ の形になる。

定理3. 入力が2つ以上の strong component は入力が1つの strong component に分解できる。

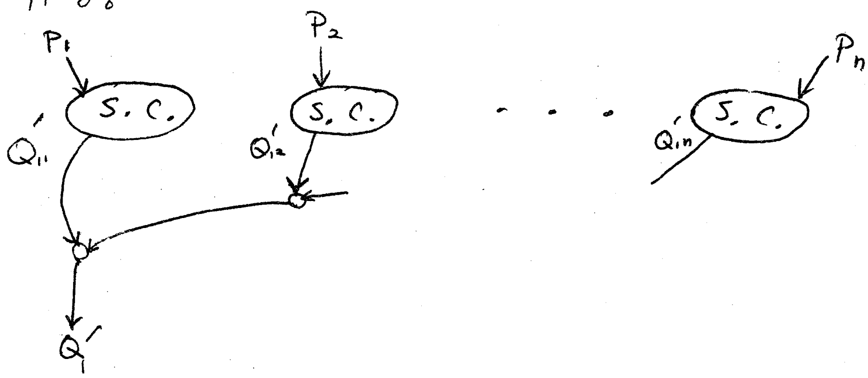
証明 Strong component を1つの node であらわすと、次の図の様に表わすことができる。



出力 arcの述語はまた（2.3）、（2-4）で表わせる。上図では、

$$Q_1 = Q_{11} \vee Q_{12} \vee \ldots \vee Q_{1n}$$
$$= f_{11}(P_1) \vee f_{12}(P_2) \vee \ldots \vee f_{1n}(P_n)$$

上図と同値のものを得るために、それをn回写し、入力arcはそれぞれに違うものを1つだけ残し、出力arcは同じ番号のものをjoinでつないで、下図のようなものを作る。



得られた述語は、

$$Q'_1 = Q'_{11} \vee \ldots \vee Q'_{1n}$$
$$= f_{11}(P_1) \vee \ldots \vee f_{1n}(P_n)$$

$$\therefore \quad Q_1 = Q_1'$$

同様にして、$Q_i = Q_i'$ $(i=1, m)$

この様にしてflow diagramを有限個の入力1つのstrong componentと有限個のisolated arcに分解した後、次の手順で分解する。

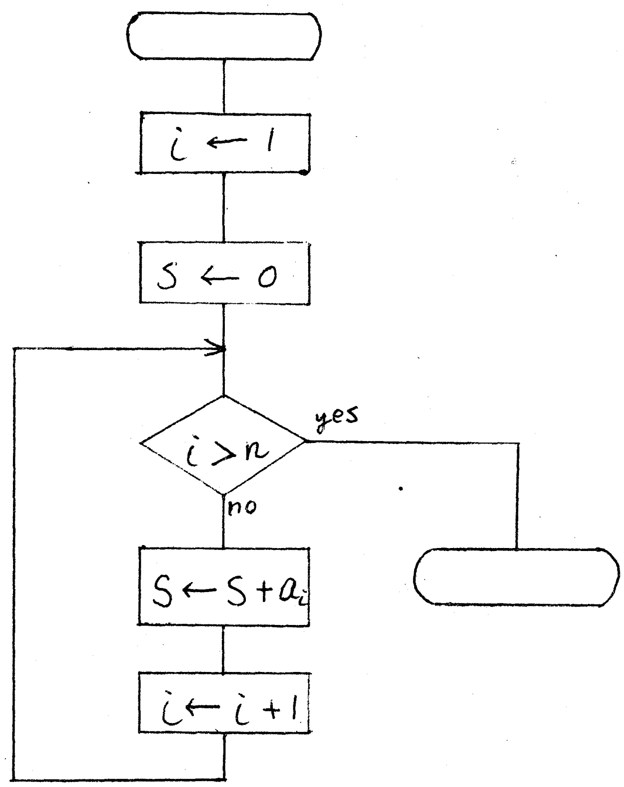
最初のflow diagramを0次のflow diagramと呼ぶ。ここですべてのstrong componentを1つのnodeとみると、0次のflow diagramにはisolated arcのみとなる。これを0次のreduced flow diagramと呼ぶ。0次のreduced flow diagramではすべてのpathは有限である。各strong componentの写像が決定できれば、出力arcのpredicateはきまる。0次のflow diagramに含まれるstrong componentはすべて1次のstrong componentと呼ぶ。

i次のstrong componentは、入力node（定理2，3よりこれは1つであり、joinである。）で切ると、1つのflow diagramと同形のグラフができるので、これをi次のopened strong componentと呼ぶ。再帰的にi次のopened strong componentを$i+1$次のstrong componentと呼ぶ。

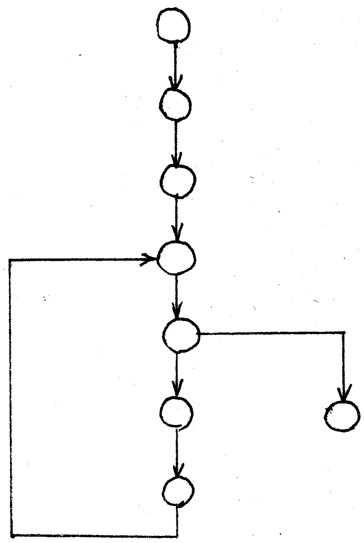
これらに写像を対応させて、1つのnodeで表わすと、これもまたisolated arcだけからなるグラフになる。この入力arcに対応する述語は決まっているので、すべてのarc

に述語を対応できる。　しかし作り方からも判るように、入力 node と出力 node は同一であるから、最後までくるとまた入力点にもどり、2回目の述語が各 arc に付けられる。　この様にして、各 arc には述語の列が対応される。

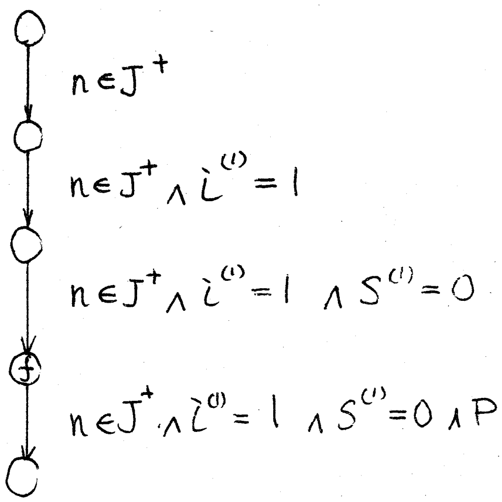
この操作は有限回で終る。　この回数でプログラムの複雑さを示すことができる。　次の flow diagram の例で考える。



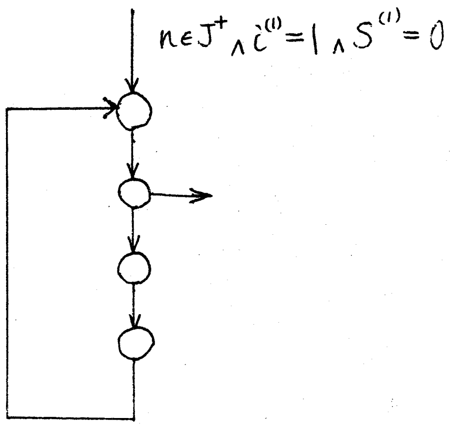
0次の flow diagram、0次の reduced flow diagram、1次の strong component、1次の opened strong component はそれぞれ次の様になる。



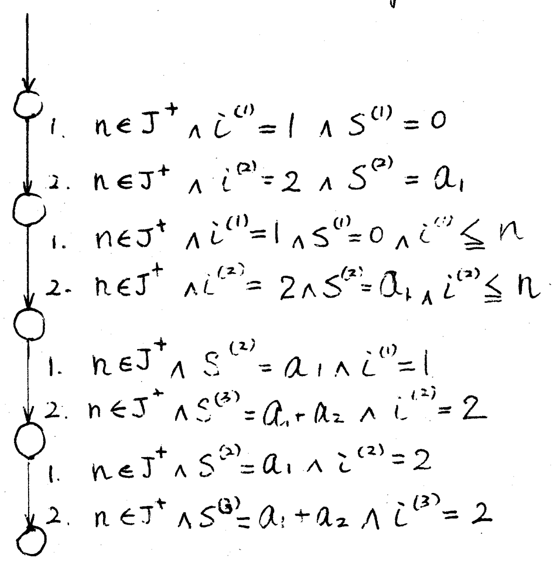
0次の flow diagram



0次の reduced flow diagram



1次の strong component



1次の opened strong component

このプログラムの複雑さは1であるということができる。

§4 停止する条件

定義　0次のreduced flow diagramでは有限個のpathがある。各pathが停止するとは、そのpath上のすべてのnodeに写像が対応できることである。

定理4　プログラムは、0次のreduced flow diagramのすべてのpathが停止すれば停止する。

定義　i次のopened strong componentが停止するとは、各arcに対応する述語の列が有限個で終ることである。

定義　i次のstrong componentのassignmentの左辺の変数をi次のプログラム変数と呼ぶ。

定理5　i次のstrong componentで、i次のプログラム変数が出口のcondition formulaの中になければ、このstrong componentは1度はいると停止しない。

定理2より、出口はconditionなので、以後の話ではcondition formulaを適当に変えて、falseのときstrong componentを出るようにする。

定義　整数空間の連続した区域をinteger intervalと呼ぶ。

定義　ある条件 ϕ を満たす変数の値域（整数空間を考えている）を C_ϕ とあらわす。

定理6　i次のflow diagram上のあらゆるpathに出口があり、かつ各path上の出口の少くとも1つのconditionに次の条件のどちらかがなりたつなら停止する。

1. $\exists n$ such that $C_{\neg\phi_1} \vee C_{\neg\phi_2} \vee \cdots \vee C_{\neg\phi_n} = J$
2. C_{ϕ_i}内のあらゆるinteger intervalが$C_{\phi_{i-1}}$内のあらゆるinteger intervalに強い意味で含まれる。（強い意味で含まれるとは、$C_{\phi_{i-1}}$のあるinteger intervalにC_{ϕ_i}のあるinteger intervalが含まれる場合、その大きさには必ず不等号がなりたっているということである。）（またsuffixは通る回数を示す。）

たとえば11頁の例では、1次のopened strong componentにはpathは1つ、出口も1つである。また

$$C_{\phi_1} = [-\infty, n]$$
$$C_{\phi_2} = [-\infty, n-1]$$
$$\vdots$$
$$C_{\phi_i} = [-\infty, n-i+1]$$

で、これは定理6.2を満たすので停止する。このとき出口のarcのpredicateは

$$n \in J^{+_2} \wedge i^{(n+1)} = n+1 \wedge S = \sum_{k=1}^{n} a_k$$

よって

$$P \equiv i = n+1 \wedge S = \sum_{k=1}^{n} a_k \quad \text{である.}$$

§5　高級言語で書かれたプログラムの正当性

高級言語のプログラムには、特別な問題として、変数の同定と制御の流れの記述がある。これらは特有な記号で示されているので、それを定義するなんらかの公理系が必要となる。ここでは flow diagram に変換して §4. の判定条件を使うことを考える。ALGOL を写像して、Tree Sort (5) のプログラムについて正当性を調べる。

```
procedure siftup (i, n) ; value i, n ; integer i, n;
  begin real copy ; integer j ;
    copy := M[i] ;
   loop: j := 2 × i ;
    if j ≤ n then
    begin if j < n then
       begin if M[j+1] > M[j] then
              j := j + 1
       end ;
       if M[j] > copy then
       begin M[i] := M[j] ; i := j ;
             goto loop
       end
```

```
        end ;
        M[i] := copy ;
    end
```

これだけの文を記述する syntax は次のように書ける。

① ⟨block⟩ ::= begin ⟨declaration list⟩ ; ⟨statement list⟩ end

② ⟨compound statement⟩ ::= begin ⟨statement List⟩ end

③ ⟨statement list⟩ ::= ⟨statement⟩ | ⟨statement⟩ ⟨statement list⟩

④ ⟨statement⟩ ::= ⟨conditional statement⟩ | ⟨basic statement⟩

⑤ ⟨conditional statement⟩ ::= ⟨if statement⟩ | ⟨if statement⟩ else ⟨statement⟩

⑥ ⟨if statement⟩ ::= if ⟨Boolean expression⟩ then ⟨unconditional statement⟩

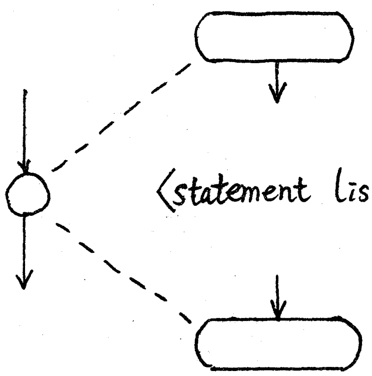
⑦ ⟨unconditional statement⟩ ::= ⟨basic statement⟩ | ⟨compound statement⟩ | ⟨block⟩

⑧ ⟨basic statement⟩ ::= ⟨unlabeled basic statement⟩ | ⟨label⟩ ⟨basic statement⟩

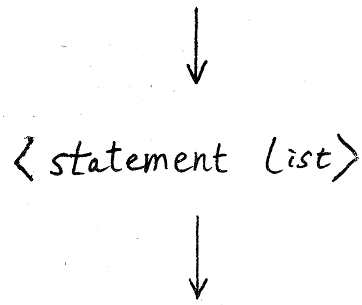
⑨ ⟨unlabeled basic statement⟩ ::= ⟨assignment statement⟩ | ⟨go to statement⟩

それぞれの syntax equation に次の様な flow diagram element を対応させる。

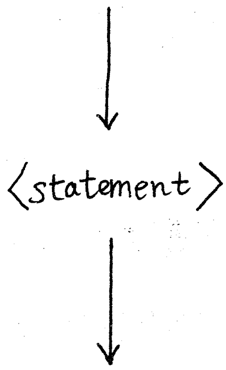
①



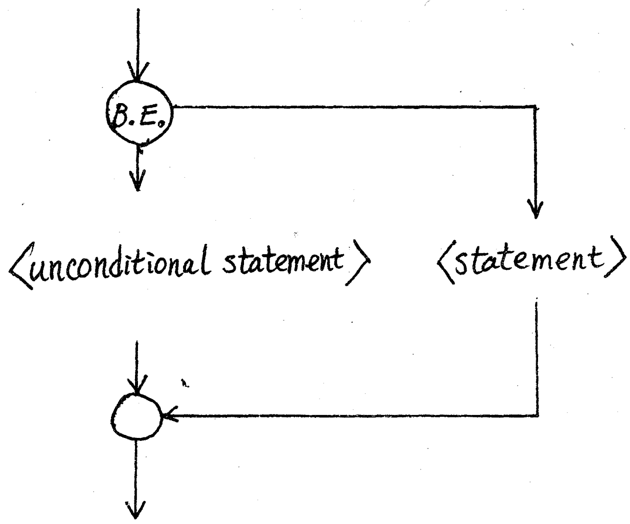
②



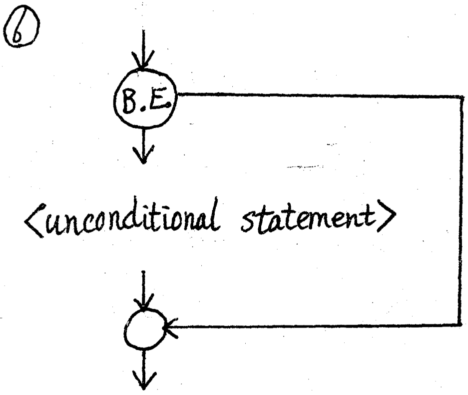
③



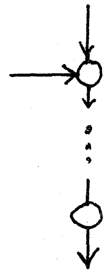
⑤



⑥

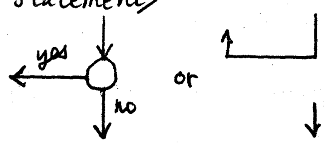


⑦

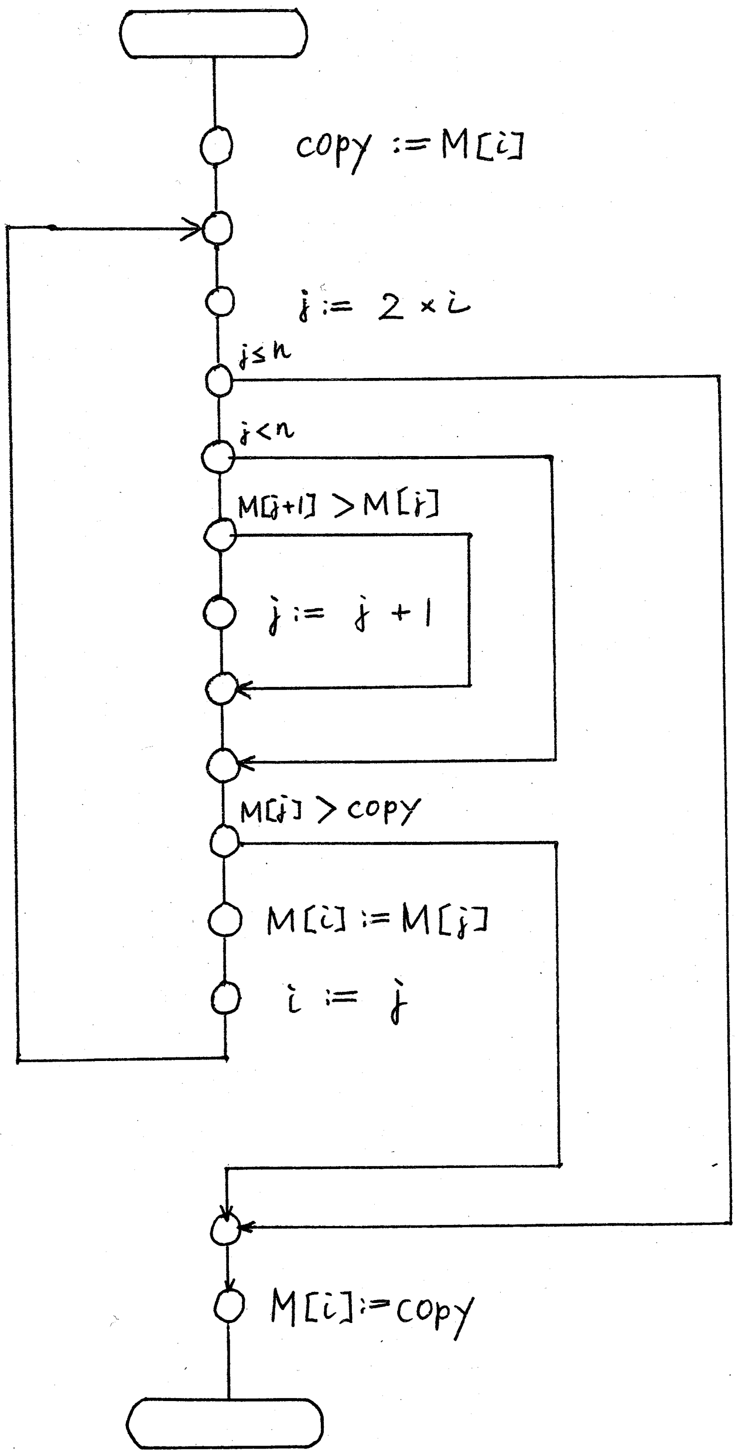


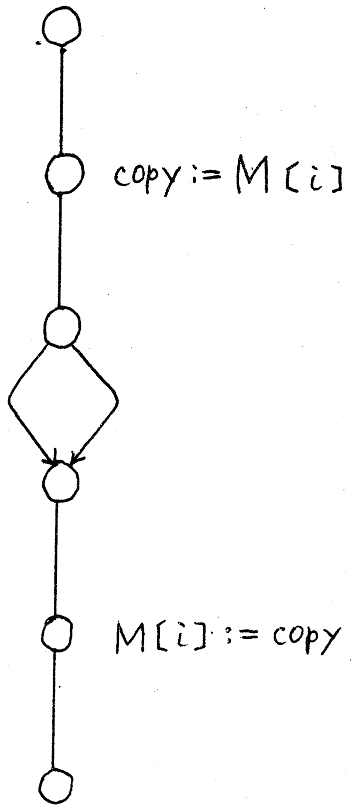
⑨ <assignment statement>

⑨ <go to statement>

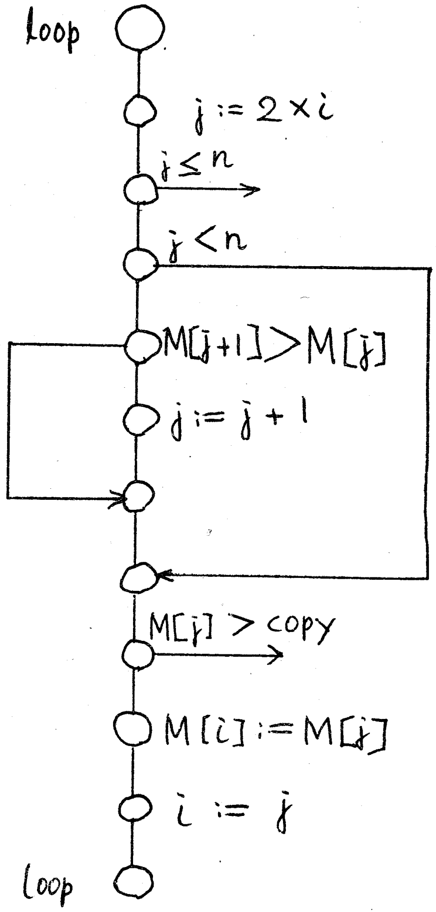


これらの定義を適用していくと、次のプログラムになる。





0 次の reduced flow diagram



1 次の opened strong component

出口は2つあるが、上の出口の condition predicate は、

$$\begin{aligned}\phi_{i+1}(j^{(i+1)}) &\equiv j^{(i+1)} \leq n \\ &\equiv 2 \times j^{(i)} \leq n \\ &\equiv 2 \times j^{(i)} \leq n \\ &\equiv j^{(i)} \leq \frac{n}{2}\end{aligned}$$

よって　$C_{\phi_{i+1}} \subset C_{\phi_i}$ で、停止する。

この predicate は、停止することが判っているので、逐次にやっても、また数学的帰納法でも計算できる。

§6 コンパイラー、コンパイラー・コンパイラー、正当性を証明するプログラムへの応用

今までのコンパイラーでは、syntax analyser が、syntax equation にあわせて、入力 string の構造を調べると、それを表わす tree 等に変換していたが、§5で述べたと同じやり方でグラフを対応させ、グラフに変換すれば、より簡単なコンパイラーを設計できるようになる。

またコンパイラー・コンパイラーを作るときにも、syntax equation とグラフを対応させて入力してやるのは、今までのセマンティックスの記述よりもはるかに簡単になる。

正当性や、停止問題を証明するプログラムを、コンパイラーの作り方で述べた方法で、プログラムに対応するグラフを出せば、数学的帰納法を証明するプログラムで証明できる。

§8 文献

(1) Floyd, R.W., "Assigning Meanings to Programs," Proc. of a Symposium in Applied Mathematics, Vol. 19, AMS, Providence, R.I., 1967, 19-37.

(2) Manna, Z., "Properties of Programs and the First-Order Predicate Calculus," JACM, 16: 244-255, 1969.

(3) Engeler, E., "Algorithmic Properties of Structures," Mathematical Systems Theory, 1: 183-195, 1967.

(4) Harary, F., Graph Theory, Addiso-Wesley, 1969.

(5) London, R.L., "Proof of Algorithms, Certification of Algorithms 245," CACM, 13: 371-373, June 1970.