

goto 文のないプログラム形について

相模工大 岩田茂樹

一般にプログラムは goto 文のないプログラムに変換可能であるが、新しい変数を導入しない、かつ、実行順序を変更しない、という条件のもとでは、goto 文なしのプログラムに変換できないことが知られている。笠井氏は、これらの条件のもとで、halt 文を 1 つしかもたないような goto 文のないプログラムへ変換できるための必要十分条件を示した [1]

しかし一般にプログラムは複数個の halt 文をもつ。本稿は上記の条件のもとで、複数個の halt 文をもち、sequence, if-then-else, do-while の 3 種類の制御構造によってあらわされるようなプログラムへの変換を考え、変換できるための必要十分条件について述べる。

本稿においては、有向グラフの各 node の出次数を 2 以下と仮定する。フローグラフ (N, E, n) とは有向グラフ (N, E)

で, N のすべての node が N 中のある特定の node n からの path 上にあるものをいう. n を start node という. また, 出次数 0 の node を halt node, 出次数 1 の node を function node, 出次数 2 の node を test node という.

定義 $F=(N, E, n)$ をフローグラフとし, u, v は N の node で, u は halt node ではないとする. もし u が function node で $(u, v) \in E$ ならば, 変換 T_1 は (u, v) , u, v を 1 つの node y_v におきかえる. このとき u または v に入る edge $((u, v))$ は除く) は y_v に入り, v を去る edge は y_v を去る. もし u が test node で, u を去り v に入る edge が 2 つあるならば, 変換 T_2 は 2 つの edge のうちの片方の edge (u, v) を除去する. もし u が test node, v が halt node かつ $(u, v) \in E$ ならば, 変換 T_3 は (u, v) を除去する. このとき v に入る edge が (u, v) の他になければ, v も除去する. もし u が test node で u を去る edge のうちの 1 つが (u, u) の形をしているならば, 変換 T_4 は (u, u) を除去する.

注意 変換 T_1, T_2 においては, u, v は同一の node であってもよい.

定義 F, F', F'' をフローグラフとする. 変換 T_i , $i \in \{1, 2, 3, 4\}$ により, F が F' に変換されるとき, $F \Rightarrow F'$ と

かく、 $\Rightarrow^*$ は $\Rightarrow$ の推移閉包とする。 $F \Rightarrow^* F'$ で $F' \Rightarrow F''$ となるような F'' が存在しないならば、 $F \hat{\Rightarrow} F'$ とかく。

補助定理 1 $F \hat{\Rightarrow} F'$ かつ $F \hat{\Rightarrow} F''$ ならば $F' = F''$ である。

(証明) F の edge の数についての帰納法により証明する。

F が 1 つの edge (u, v) よりなるとき、 u は明らかに function node である。 よって $F' = F''$ となる。

F は n edge からなるフローグラフとし ($n > 1$)、 n edge 以下のフローグラフについては補助定理はなりたつものと仮定する。 いま $F \Rightarrow F_1 \hat{\Rightarrow} F'$ 、 $F \Rightarrow F_2 \hat{\Rightarrow} F''$ とする。

$F_1 \Rightarrow F_3$ 、 $F_2 \Rightarrow F_3$ となるような F_3 の存在を示せば、 F_3 は n edge 以下のフローグラフであるから $F_3 \hat{\Rightarrow} F'''$ となる。

また、 $F' = F'''$ 、 $F'' = F'''$ であるから $F' = F''$ である。

$F \Rightarrow F_1$ の変換で (a, b) が、 また、 $F \Rightarrow F_2$ の変換で (c, d) が除去されたものとする。 次の 4 つの場合にわけて考える。

Case 1 $a = c$ 、 $b = d$ のとき。 (図 1 参照)

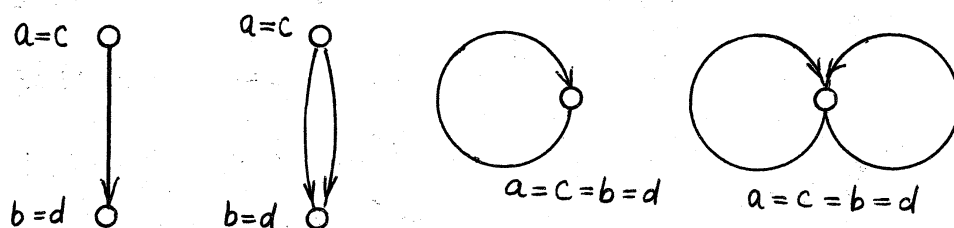
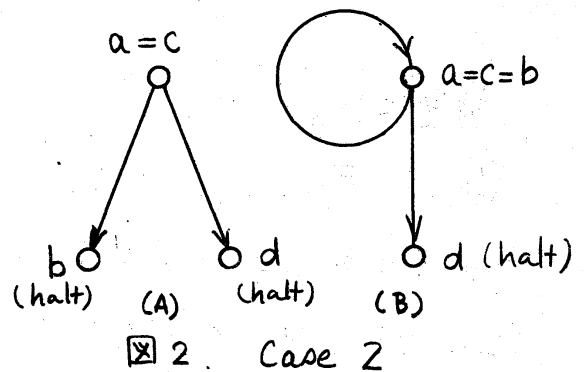


図 1. Case 1

明らかに $F_1 = F_2$ である. よって $F' = F''$.

Case 2. $a = c$, $b \neq d$ のとき. (図2参照)

a (または c) は test node
である. $a \neq b$, かつ $c \neq d$
ならば b, d 共に halt node
である. $a = b$ ならば d は
halt node で図2(B)のよう

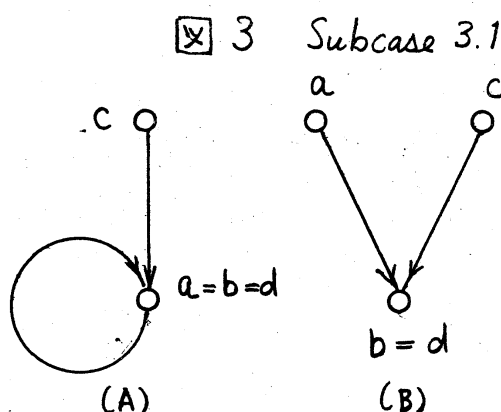


な場合である. いずれの場合も F_1 に (c, d) をとり除く変換
をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは
同一のグラフ F_3 である. $a = d$ のときは $a = b$ の場合と同
様に証明される.

Case 3 $a \neq c$, $b = d$ のとき.

Subcase 3.1. a, c が共に function node のとき.

$a = b$ とすると $c \neq d$ である. (図3(A)参照) この場合は
 F_1 に $(c, a/b)$ をとり除く変換をしたグラフと F_2 に $(a, c/d)$



をとり除く変換をしたグラ
フは同一のグラフ F_3 となる.
 $a \neq b$, $c = d$ のときは上と
同様である. $a \neq b$, $c \neq d$
のときは F_1 に $(c, a/b)$
をとり除く変換をしたグラ

F と F_2 に $(a, c/d)$ をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図3(B)参照)

Subcase 3.2. a が test node で c が function node のとき.

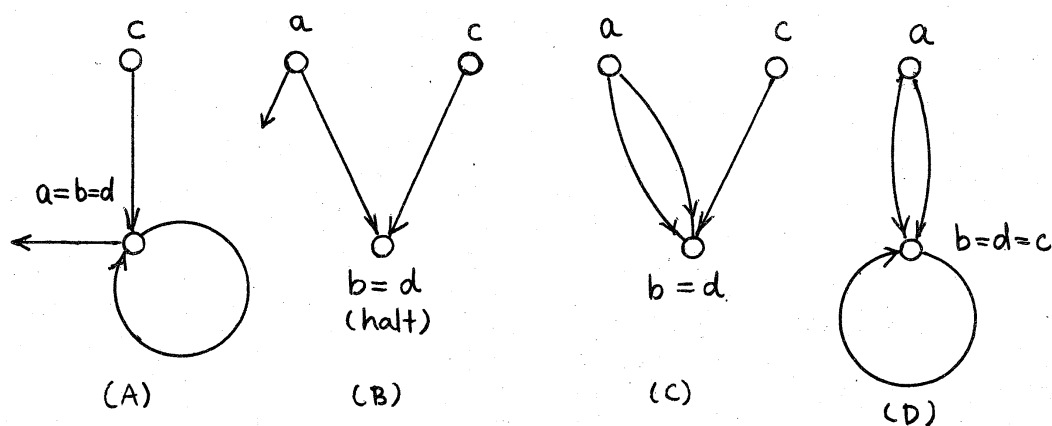


図4 Subcase 3.2.

$a = b$ ならば $c \neq d$. (図4(A)参照) F_1 に (c, d) をとり除く変換をしたグラフと F_2 に $(c/d, c/d)$ をとり除く変換をしたグラフは同一のグラフ F_3 となる. $a \neq b$ のときは, F_1 に (c, d) をとり除く変換をしたグラフと F_2 に $(a, c/d)$ をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図4(B), (C), (D)参照)

Subcase 3.3. a が function node で c が test node のとき.
 Subcase 3.2. と同様にして証明される.

Subcase 3.4. a, c 共に test node のとき (図5参照)
 F_1 に (c, d) をとり除く変換をしたグラフと F_2 に (a, b) をとり

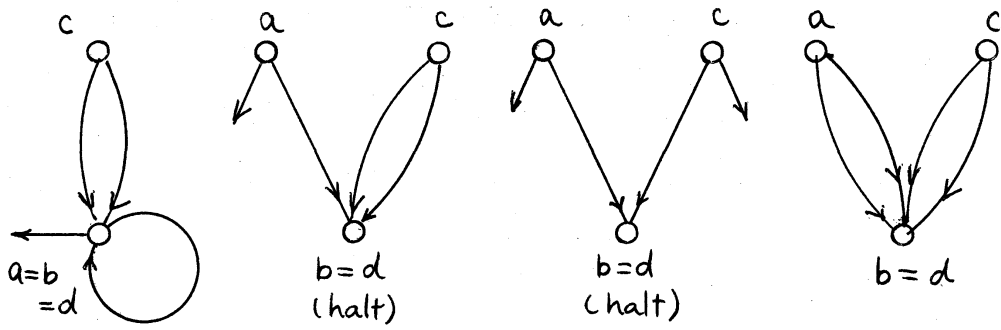


図5 Subcase 3.4

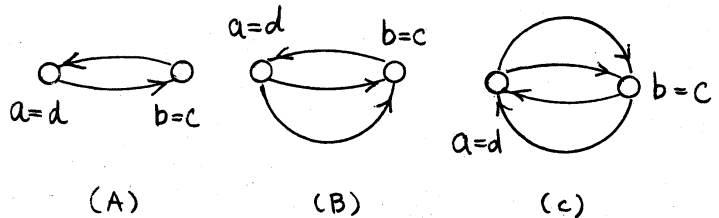
除く変換をしたグラフは同一のグラフ F_3 となる。

Case 4. $a \neq c, b \neq d$ のとき。

Subcase 4.1. $a = d, b = c$ のとき。

a, c が共に function node ならば, $F_1 = F_2$ となる。(図6(A)参照) a が test node で c が function node ならば, F_1 に (c, d) をとり除く変換をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは同一のグラフ F_3 となる。(図6(B)参照)

a が function node
で c が test node



ならば上と同様に

して証明される。

図6 Subcase 4.1.

a, c 共に test node ならば, F_1 に (c, d) をとり除く変換をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは同一のグラフ F_3 となる。(図6(c)参照)

Subcase 4.2. $a = d, b \neq c$ のとき。

仮定により, $a \neq b$ かつ $c \neq d$ である. a, c 共に function node ならば, F_1 に $(c, a/b)$ をとり除く変換をしたグラフと F_2 に $(a/d, b)$ をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図7(A)参照) a が test node で c が function node ならば, F_1 に (c, d) をとり除く変換をしたグラフと F_2 に $(a/d, b)$ をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図7(B),(C)参照) a が function node で c が test

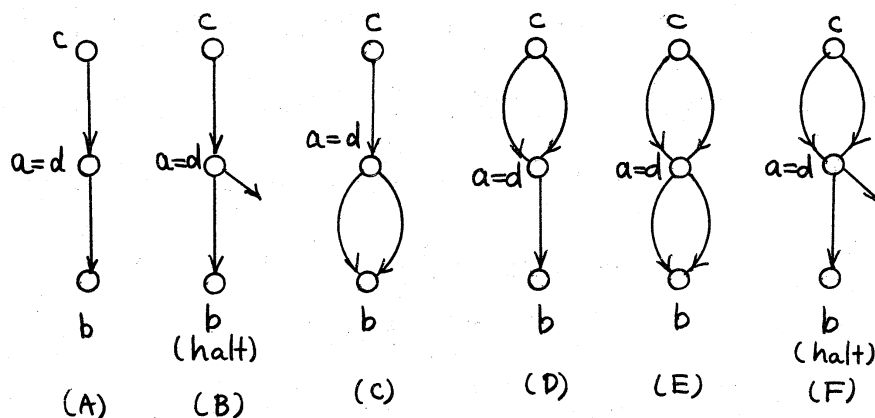


図7 Subcase 4.2.

node ならば, F_1 に $(c, a/b)$ をとり除く変換をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図7(D)参照) a, c 共に test node ならば, F_1 に (c, d) をとり除く変換をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは同一のグラフ F_3 となる. (図7(E),(F)参照)

Subcase 4.3. $a \neq d, b=c$ のとき.

この場合は Subcase 4.2と同様にして証明される。

Subcase 4.4. $a \neq d, b \neq c$ のとき。

F_1 に (c, d) をとり除く変換をしたグラフと F_2 に (a, b) をとり除く変換をしたグラフは同一のグラフ F_3 となる。

以上により, すべての場合をつくしたので, 補助定理1は証明された。 (証明終)

w をフローグラフ F の node とし, w が F の強連結部分グラフ S 上になく, とする。(1) x が S の node で, w が, (2) $y = w$ かまたは, S を通らないような y から w への path がある, のとき, S は w への 出口 (x, y) をもつという。

定義 S をフローグラフ F の極大強連結部分グラフとする。 S のすべての cycle c に対し, u が c 上にあるか, または, c が u への出口を1つしかもたないような S の node u が存在するならば, S は semimodular であるといい, F のすべての極大な強連結部分グラフが semimodular ならば, F は semimodular であるという。

補助定理2. $F \Rightarrow F'$ とする。 F が semimodular でないならば, F' も semimodular ではない。

(証明) $F \Rightarrow F'$ の変換で edge $e = (x, y)$ が除去された

とする. また, F は semimodular ではなく, かつ, F' は semimodular であるとする. すなわち, F の極大強連結部分グラフ S が存在して, (1) S は semimodular ではない, (2) e は S の edge である, かつ, (3) 変換後の S を S' とすると S' は semimodular である. 明らかに S' は F' の極大強連結部分グラフであり, 適用された変換は T_3 ではない. S' は semimodular であるから, S' の任意の cycle c' に対して, u' が c' 上にあるか, または c' から u' への出口が1つであるような S' の node u' が存在する.

u を S の node で, もし u' が変換によりまとめられた node なら $u = x$, その他のときは $u = u'$ とする. S は semimodular でないから S のある cycle c_1 に対して u への2つの出口 $e_1 = (v_1, w_1)$ と $e_2 = (v_2, w_2)$ が存在する. もし, c_1' を c_1 の変換後のグラフとすると c_1' は cycle である. なぜかといえば, v_1, v_2 を去る4つの edge とも e ではないからである. S' において, 2つの path: v_1 から u' と, v_2 から u' が存在し, (v_1, w_1) と (v_2, w_2) は2つの異なった c_1' から u' への出口である. すなわち c_1' は2つの出口を有し, S' が semimodular であることと矛盾する.

(証明終)

定理1 もし $F \xrightarrow{*} 0$ ならば, F は semimodular である.

(証明) F を semimodular ではないとある. 補助定理2より, 0 も semimodular ではない. これは矛盾である.

(証明終)

定義 [1] halt node を1つしかもたないフローグラフ F において, もし F のすべての強連結部分グラフがその halt node への出口を1つしかもたないならば, F は modular であるという.

定理2 [1] F を halt node が1つしかないフローグラフとし, F のすべての node は F の start node から F の halt node への path 上にあるものとする. F が modular であるための必要十分条件は, F が T_1, T_2, T_4 をくり返し適用することによって 0 に変換されることである.

定理3 フローグラフ F が semimodular ならば, $F \xrightarrow{*} 0$.

(証明) F 中の極大強連結部分グラフの数についての帰納法により証明する. もし F に強連結部分グラフがないならば, 変換 T_3 をくり返すことにより F は 0 に変換される. F に極大強連結部分グラフが1つ存在するならば, 変換 T_3 により, $F \xrightarrow{*} F'$ となる. 但し, F' は halt node をもたなく, かつ, 極大強連結部分グラフを1つもつフローグラフである.

$F' = (N', E', n')$, $(u, v) \in E'$ とする. F' の極大強連結部分グラフの任意の cycle c について, u は c 上にあるか, または, c から u への出口は 1 つである. いま,

$$(i) \quad N'' = N' \cup \{w, z\},$$

$$(ii) \quad E'' = (E' - \{(u, v)\}) \cup \{(u, w), (w, v), (w, z)\},$$

$$(iii) \quad w, z \text{ は } N' \text{ 中の node ではない,}$$

とすると, $F'' = (N'', E'', n')$ は halt node が z 1 つで, F'' のすべての強連結部分グラフは z への出口を 1 つしかもたない. よって F'' は modular であるから定理 2 より $F'' \rightsquigarrow 0$ である. 明らかに $F'' \rightsquigarrow F'$. 補助定理 1 より $F' \rightsquigarrow 0$. よって $F \rightsquigarrow 0$.

F 中に極大強連結部分グラフの数が k 以下のとき定理はなりたつと仮定する. ($k \geq 2$) 変換 T_3 により F は halt node のないフローグラフ F_1 に変換される. F_1 中には out-edge のない極大な強連結部分グラフが少なくとも 1 つは存在するので, $k=1$ のときと同様にして, F_1 は $(k-1)$ の極大強連結部分グラフをもつフローグラフ F_2 に変換される. 帰納法の仮定により $F_2 \rightsquigarrow 0$. ゆえに $F \rightsquigarrow 0$.

(証明終)

定理 4. F をフローグラフとする. F が semimodular であるための必要十分条件は $F \rightsquigarrow 0$ である.

(証明) 定理1, 3より明らか.

(証明終)

形式的ではないが, フローチャートの計算を表わす箱を node におきかえることによりフローグラフにおきかえられるものと考えられる. アルゴリズム言語において, フローグラフの変換 T_1 で edge (u, v) が除去されることは, u, v が異なる場合についてみると, u, v に対応するフローチャート上の計算を sequence により, プログラム上にかきあらわすことに対応している. また同様に, T_2 と T_3 は if-then-else により, T_4 と2つの node が一致する場合の T_1 は do-while により, それぞれプログラム上にかきあらわすことに対応している. このとき, if-then-else, do-while の制御構造において, then-clause, else-clause, while-clause の中に halt 文が入ることに注意する. またフローグラフが O に変換されることは, そのフローグラフが3種の制御構造でかきあらわされることを示し, O に変換されないならば, 3種の制御構造ではかきあらわされないこと, すなわち goto 文が必要であることを示している. 定理4は, フローグラフ F が3種の制御構造でかきあらわされるための必要十分条件は, F が semimodular であることを述べている.

文献

- [1] T. Kasai , Translatability of flowcharts into while programs , JCSS 9 (1974) pp.177-195.
- [2] M.S. Hecht and J.D. Ullman , Flowgraph reducibility , SIAM J. Computing 1 (1972) pp.188 - 202.