数理解析研究所講究録
213巻 1974年 210-219

While Programs について

京大・数理解析研
笠井琢美

§1 序

while program とは goto を含まない program (goto-less program) の抽象的モデルである. 多くの人に指摘されているように [2, 3, 8] 任意の program は while program に変換できる. さらに while は1個で十分であることも知られている[6]. しかしこれらの変換は一般に program の本質的な構造を少しも減少させていない. [8] では (1) 新しい変数を導入しない (2) 実行順序を変えない, といった制限をつけた変換だけを考え, この変換のもとで while program に変換できないような program の例を示した. 著者は [7] でこういった制限の下で program S が while program に変換できる必要十分条件は S が modular であることを示した. ここではこの modularity という概念が Hecht, Ullman の collapsibility$^+$ [5], Allen [1] と Cocke [4] の reducibility$^+$ と同値な概念であることを報告する.

§1. 定義

$F = \{f, g, \cdots\}$ と $P = \{p, q, \cdots\}$ を集合とする. F の元を function symbol (または assignment statement), P の元を relation symbol (または logical expression)

と呼ぶ. Pの各元 p に対し $\bar{p}$ を一つの記号とみなし, $\hat{P}=P\cup\{\bar{p}\mid p\in P\}$ と置く.

flowchart とは次の条件 I ~ III を満す transition graph S のことをいう. (a,l,b) で頂点 a から頂点 b への label l を持つ辺を示す

I. Sの各頂点 a に対し

(i) a から出る辺はただ一つで (a,f,b), $f\in F$ の形, であるか

(ii) a から出る辺は 2 つで (a,p,b), $(a,\bar{p},c)$, $p\in P$ の形

(iii) a から出る辺はない.

のいずれかである. (iii) のような頂点を S の halt node と呼ぶ.

II. S の特定の頂点が start node として指定されている. start node を記号 ϕ で示す.

III. Sの各頂点は ϕ から halt node の path の上にある.

一般性を失うことなく各 flowchart はただ一つの halt node (記号 $ で示す) を持つとしてよい. ただ一つの頂点だけからなるような flowchart を null flowchart と呼び Λ で示す.

定義 a を Sの頂点とするとき $T(S,a)$ で a から $ への path によって spell out されるような語全体からなる集合を表わす. 特に $T(S,\phi)$ を $T(S)$ と書く したがって $T(S)$ とは S によって accept される語全体からなる集合である.

2 つの flowchart S_1 と S_2 が congruent (isomorphic ,

computation-sequence-equivalent) であるとは $T(S_1) = T(S_2)$ のときをいう.

flowchart S が minimal であるとは S の任意の頂点 $a \neq b$ に対し $T(S, a) \neq T(S, b)$ となるときをいう.

定義 while program は帰納的に次のように定義される

(1) λ (空語) は while program.

(2) F の元 f は while program

(3) α と β が while program なら次は while program

$\alpha\beta$, (if q then α else β), (while q do α)

但し q は $\hat{P}$ の元

定義 各 while program α に対し, α の trace set $T(\alpha)$ を帰納的に次のように定義する.

(1) $T(\lambda) = \lambda$,

(2) $T(f) = f$ for each $f \in F$,

(3) α と β が while program, $q \in \hat{P}$ なら

$$T(\alpha\beta) = T(\alpha)T(\beta)$$

$$T(\text{if } q \text{ then } \alpha \text{ else } \beta) = qT(\alpha) \cup \bar{q}T(\beta)$$

$$T(\text{while } q \text{ do } \alpha) = (qT(\alpha))^* \bar{q}$$

但し $\bar{\bar{p}} = p$ for each $p \in P$.

定義 flowchart S は $T(S) = T(\alpha)$ となる while program α が存在するとき translatable であるという.

§3 Modularity

定義 Sの頂点の集合 W が S の section であるとは W の任意の2元 a, b に対し a から b への nonnull な W の元だけを通る path が存在するときをいう.

辺 (a, l, b) が section W の exit であるとは $a \in W$ であり, b から $ へのWの元を通らない path が存在するときをいう。(したがって $b \notin W$)

flowchart S が modular であるとは S の任意の section がただ1つの exit を持つときをいう.

定理 3.1. modular な flowchart は translatable.

定理 3.2 S が minimal なら

S is translatable $\Leftrightarrow$ S is modular

注 定理 3.1 の逆は成立しない.

Algorithm 与えられた flowchart S が translatable であるかどうかを判定するには

(1) S と congruent な minimal な flowchart S′ を構成する.

(2) S′ が modular なら S は translatable

そうでないなら translatable でない.

(定理 3.1, 3.2 の証明は [7] を参照.

§4 Collapsibility$^+$

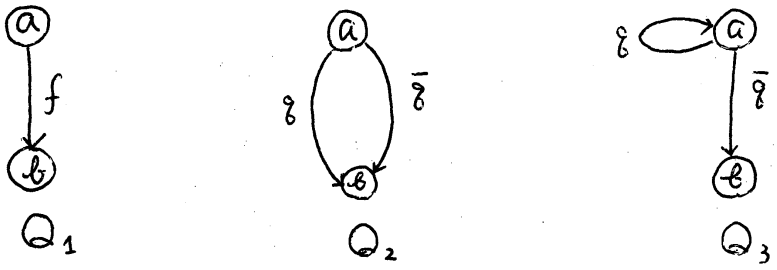
(注 ここで＋と書いたのはもとの定義[5]と多小異るためである. §5の reducibility$^+$ の＋についても同様)

Transformation $\Longrightarrow$. S が次の Q_1, Q_2, Q_3 のいずれかを subgraph として含んだとする. このとき a と b を一つの頂点にまとめて S' が得られるなら $S \Rightarrow S'$ と書く.

$$S_0 \Rightarrow S_1 \Rightarrow \cdots \Rightarrow S_k$$

のとき $S_0 \overset{*}{\Rightarrow} S_k$ と書く.

S が collapsible$^+$ であるとは $S \overset{*}{\Rightarrow} \Lambda$ となることである.



§5 Reducibility$^+$

Reducibility は次の Interval analysis$^+$ によって判定される.

Interval analysis 入力: S の頂点 b. 出力: exit b を持つ Interval $I(b)$.

(A) 1. $I(b) \leftarrow \{b\}$

2. $c \notin I(b)$ で c から出る辺はすべて $I(b) \cup \{c\}$ に入るなら $I(b) \leftarrow I(b) \cup \{c\}$

3. 可能なかぎり 2 をくり返す.

Sが与えられたとき，Sの頂点を次のalgorithmにしたがってintervalI(b_1), …, I(b_k)に分割する（Eは頂点の集合，Lはintervalのlist）

(B) 1. $H \leftarrow \{\$\}$, $L \leftarrow \lambda$

2. $H = \phi$ ならhalt

3. $b \in H$ を選びI(b)を計算；

$L \leftarrow L, I(b)$；

$H \leftarrow H - \{b\} \cup \{c \mid c$からI($b$)への辺があり$c \notin \cup L\}$；

goto 2

（但し $L = I(b_1), \cdots, I(b_t)$ のとき $\cup L = I(b_1) \cup \cdots \cup I(b_t)$ を意味する．

定義 Sが与えられたときSのderived flowchart I(S)を次のように構成する．

I(S)の頂点はSのinterval I(b_1), …, I(b_k)

I(S)は辺 (I(b_i), l, I(b_j)) を含む ⇔ Sは辺 (b_i, l, c), $c \in I(b_j)$ を含む

I(S)のstart nodeはSのstart nodeを含むinterval

定義 $S' = I(S)$ のとき $S \vdash S'$ と書く．又 $S_0 \vdash S_1 \vdash \cdots \vdash S_k$ のとき $S_0 \vdash^* S_k$ と書く．Sがreducibleであるとは $S \vdash^* \Lambda$ となるときを言う．

定理　flowchart S に対し 次は 同値である.

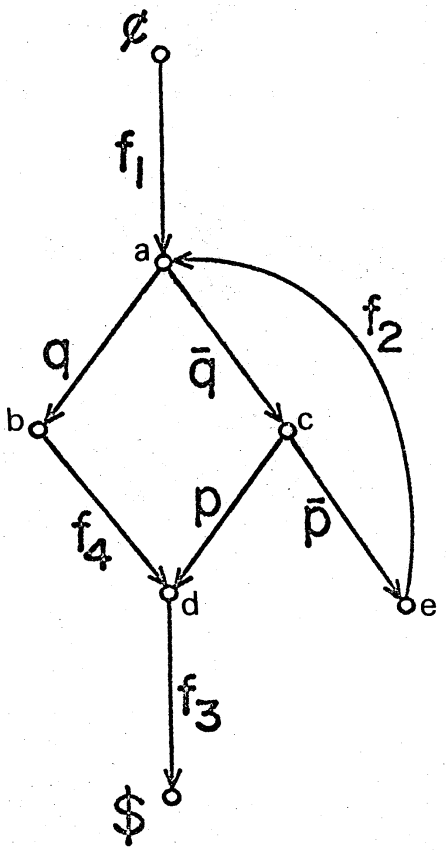
1. S は modular
2. S は collapsible
3. S は reducible

§6 Example

次は [8] で 与えられた translatable でない program である.

```
for i:=1 step 1 until n do
    if A[i]=x then go to found;
not found: n:=i; A[i]:=x; B[i]:=0;
found; B[i]:=B[i]+1;
```

この program の flowchart は 次図のようになる.



q ≡ i > n?

p ≡ A[i]=x?

f_1 ≡ i:=1

f_2 ≡ i:=i+1

f_3 ≡ B[i]:=B[i]+1

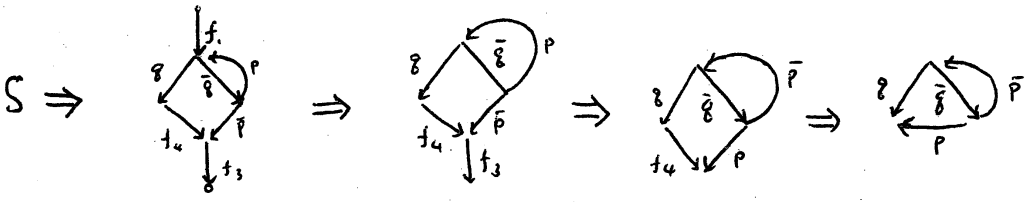
f_4 ≡ n:=i;A[i]:=x;B[i]:=0;

この flowchart を S とする. S は明らかに minimal である. S が translatable であるかどうかを3つの方法で調べてみる.

I Modularity

$W = \{a, c, e\}$ は S の section であるが. W は2つの異なる exit (a, g, b) と (c, p, d) を持つ. よって S は modular でない.

II Collapsibility⁺



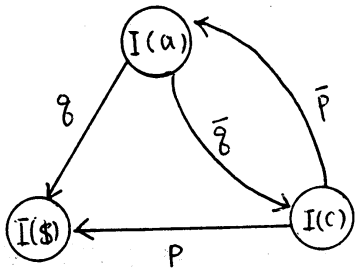
となり, 最後の flowchar はこれ以上 ⇒ で変換できない. よって S は collapsible でない.

注 $S \overset{*}{\Rightarrow} S_1$, $S \overset{*}{\Rightarrow} S_2$ なら $S_1 \overset{*}{\Rightarrow} S_3$, $S_2 \overset{*}{\Rightarrow} S_3$ となる S_3 が存在することがわかっている.

III Reducibility⁺

algorithm A で exit \$ を持つ S の interval $I(\$)$ を計算すると $I(\$) = \{\$, d, b\}$ となる. algorithm B で S を interval に分解すると $I(c) = \{c\}$, $I(a) = \{a, e, \phi\}$, $I(\$)$ となる.

よって derived flowchart $I(S)$ は次のようになる



$= S'$

$I(S') = S'$ であるから S は reducible でない.

§7 Algorithm としての評価

flowchart が translatable であるかどうかを判定する algorithm として Modularity, Collapsibility$^+$, reducibility$^+$ を考える. 今 S を頂点の個数が n である flowchart とする. このとき直接に上の性質を調べると

Modularity $O(2^n)$ step　　Collapsibility$^+$ $O(n \log n)$ step

Reducibility$^+$ $O(n^2)$ step

となり Collapsibility$^+$ がいちばんよい.

しかしながら Modularity は Collapsibility$^+$ = Reducibility の一つの static な characterization を与えると考えられる.

REFERENCES

1. F. E. Allen, Control flow analysis. SIGPLAN Notices 5 (1970), pp. 1-19.

2. J. Bruno and K. Steiglitz, The expression of algorithms by charts TR 88, Computer Science Labaratory, Princeton University, Princeton, 1971

3. D. C. Cooper, Böhm and Jacopini's reduction of flow charts, Comm. ACM 10 (1967), 463, 473.

4. J. Cocke, Global common subexpression elimination, SIGPLAN Notices, 5 (1970), pp. 5-15.

5. M. S. Hecht and J. D. Ullman, Flow graph reducibility, SIAM J. Comput. Vol. 1, No. 2, 1972, pp. 188-202

6. K. Hirose and M. Oya, General theory of Flowcharts, Comm. Math. Univ. SANCTI PAULI, to appear

7. T. Kasai, Translatability of flowcharts into while programs, JCSS to appear. 前刷の用意あり

8. D. E. Knuth and R. W. Floyd, Notes on avoiding GOTO statements, Information Processing Letters, Vol. 1, No. 1 (1971)

9. C. A. R. Hoare, An axiomatic basis of computer programing, Comm. ACM, 12 (1969), pp. 596-580, 583