

半順序グラフと平行プログラムについて

長崎大医 中村 剛
植村憲治

非同期式計算機として有名なイリヤックⅡが発表されて15年余りになり、いわゆる“非同期”という思想は広まり、除々に計算機設計にも活かされてきているように思われるが、未だイリヤックⅡを超える本格的な非同期式計算機が設計されたという話は聞かない。もっともこれは我々の耳に入らないだけで、どこかに存在しているのかもしれない。非同期回路の中でも *Speed-independent* 回路の長所は、使用する素子の反応速度についてはなんら知らなくて良いということである。すなわち、各素子の速度には無関係に設計者の意図したとおりの正しい動作をすることであり、この為にたとえ使用している素子が動かなくなってしまう(出力が変わらなくなってしまう)場合にも誤動作は生じず、ただその素子と論理的なつながりを持った素子の一部の動作が制限されてしまうだけである。又ひどく速度の違う素子を使用する場合に

は同期式に比べて、はるかに全体としての処理速度が上がる
ことが期待される。一方短所はなんといっても論理設計の困
難さが第一である。イリヤックⅡの場合はフローチャートな
る指図書をもとにカンと経験で設計しつくしたと報告されて
いる。現在でもその状況は変わらず、統一的あるいは普遍的
な設計手順で充分実用的なものは開発されていないようであ
る。イリヤックⅡを設計したグループのうちのD. E. Muller
らにより創始されたセミモジュラー回路理論はその後むしろ
数学的理論として発展してきて、最近ようやく、もとの設計
図がセミモジュラーの時にはそれをある基本的な論理素子の
みで実現できる手法が開発された。しかしながら、その手法
により構成される回路では素子の数が多くなり、実用に適す
るかどうかが疑問である。一つにはもとの設計図がセミモジュ
ラーという点にも問題があるように思える。すなわち、セミ
モジュラーというのは実際の回路設計においては広すぎる概
念と思える。実際イリヤックⅡで用いられたフローチャート
は、一部の特殊なものを除けば、セミモジュラーよりはるかに
狭いクラスで記述されてしまう。又最近のLSIの発展をみ
る時に、回路を構成する最小単位を論理素子にするよりも、
より機能の広いオートマトンとしたアプローチもなされねば
ならないであろう。オートマトンを設計に使用するさいの理

論的な方面からの難点としては、個々のオートマトンを記述する為の手段が状態遷移図であるということである。状態遷移図は直観的にとらえにくくしかも冗長なことが多い。代数的オートマトンの考えにもとづいたコンパクトなオートマトンを想定してみるのが上の難点を克服できる一つの手段かもしれない。ここでは論理設計を単純化しかつ通常必要とされる記述能力を備えた指図書のクラスについての一考察を述べることにする。

§ 1. Partially Ordered Graph

H を有限集合、 \mathcal{A} を $H \times H$ の部分集合とする。 H の要素 h を ノード と呼び、 \mathcal{A} の要素 (h, k) を h を始点、 k を終点とする 矢 と呼ぶ。 \mathcal{A} は通常のダイグラフ (向きつけられたグラフ) である。 H の要素の列 h_1, h_2, \dots, h_m で $(h_i, h_{i+1}) \in \mathcal{A} (1 \leq i \leq m-1)$ となるものを \mathcal{A} 上の列と呼ぶ。特に異なる要素からなるものを単純列と呼ぶ。 \mathcal{A} が次の P1, P2, P3 を満足するとき Partially Ordered Graph (POG) という。

(P1) 全ての $h \in H$ について $(h, h) \in \mathcal{A}$

(P2) 特別なノード $s \in H$ が存在して s から s 以外のノードには \mathcal{A} のパスを通して到達可能であるが、 s へは他のどのノードからも到達できない。 s を START

と書く。

(P3) START から始まる単純列において、 h_1 と h_2 ($h_1 \neq h_2$)がこの順序で現れたとすると、 START から始まり、 h_1, h_2 を含む全ての単純列において h_1 が h_2 より先に現われる。

補題1 \mathcal{H} をPOGとする。 START から始まる単純列において h_1, h_2 がこの順序にあらわれる時に $h_1 > h_2$ とかき $h_1 > h_2$ 又は $h_1 = h_2$ の時 $h_1 \geq h_2$ と書くことにすれば、 \geq は半順序になる。

補題2 \mathcal{H} をPOGとする。 $h(1) \rightarrow \dots \rightarrow h(m) \rightarrow h(1)$ を \mathcal{H} のループとし $h(i) \neq h(j)$ ($i \neq j$)とすると、 $h(1), \dots, h(m)$ は simply order である。又、 $h \rightarrow k$ であったとすると $h > k$ 又は $h < k$ のいづれか一方が成り立つ。 $h > k$ の時 $h \rightarrow k$ を下向き、 $h < k$ の時上向きと呼ぶ。

補題3 \mathcal{H} をPOGとする。 $h_1 | h_2$ ($h_1 \geq h_2, h_2 \geq h_1$ のどちらも成り立たないことを示す)で h_1 から h_2 にいたる \mathcal{H} の列、 $h_1 = h(0) \rightarrow \dots \rightarrow h(m) = h_2$ が存在したとすると、ある $0 < k < m$ について、 $h(k) > h_1$ かつ $h(k) > h_2$ が成り立つ。

\mathcal{H} をダイグラフとする。 \mathcal{H} のノード h, k について

て $u \xrightarrow{*} v$ により u から v へ到達できるか又は $u = v$ なることを示す。 $u \xrightarrow{*} v$ かつ $v \xrightarrow{*} u$ なるとき $u \leftrightarrow v$ と書く。 \leftrightarrow は同値関係になる。 M のノードの集合 H の \leftrightarrow による同値類で u を含むものを $C(u)$ で表わし u の連結成分と呼ぶ。

補題 4 M を POG とする。任意のノード u について $C(u)$ は最大ノードをもつ。又、 $u' \rightarrow u$, $u' \notin C(u)$, となるのは u が $C(u)$ の最大ノードの時に限られる。

§ 2. 平行動作

この節では M のノードを実行命令、 M を実行順序を指定した指図書とみなす。まず M の頂点分離 (Vertex Separator) の概念を定義する。 M の V.S. とは M の全ての矢の上に白丸をおいたものであり、 M_0 で示す。この定義は非常に直観的であるが分かり易いし、ここでは十分厳密性を保てる。勿論厳密な定義は可能であるが非生産的に思えるのでこの直観的な定義を採用する。矢 $u \rightarrow v$ 上の白丸を so_{uv} で示す。 M のノードの実行される順序を定義するのに M_0 にポインターの概念を導入する。ポインターは最初は START から出ている全ての白丸 so_{uk} の上に分離して位置する。一般にポインターが白丸 so_{uk} の上にあり、更に v に入っている“必要な矢”の上の白丸にポインターがそろろうと、 v が実行される。すると v に入って

いる矢の上の白丸のポインターは全てなくなり、 k から出ている全ての矢の上の白丸に分離して位置する。正確に述べると次の様になる。

(EP1): はじめはポインターはSTARTから出ている全ての白丸の上のみ存在する。

(EP2): ポインターが k_0 上であり、 k からでていない矢がない時にはそのポインターは k_0 で停止する。

(EP3): k から出ている矢があるものとする。 k に入っている矢のうち以下で述べる“必要な矢”の上の白丸にポインターがそろると、 k に入っている矢の上の白丸のポインターは全てなくなり、 k から出ている全ての矢の上の白丸に分離して位置する。必要な矢とは:

(a) k に入っている矢が k 以外にないときは
 $k \rightarrow k$

(b) k からでていない矢の上の白丸にまだ一度もポインターがいたことがない時は、全ての^の下向き
 $k \rightarrow k$

(c) k にすでにポインターがいたことがある時は全ての上向きの $k \rightarrow k$ と、下向きの $k \rightarrow k$ のうちである $k' \geq k$ が存在して k が上向きの

矢を受けているもの。

補足 k に入っている必要な矢の上の白丸にポインタースがそろりと、 k で表わされている実行命令が実行されるが、実行時間は未知であり、いつ終わるか予測することはしない。ただいつかは終わり、終わったならば k に入っている矢の上のポインタースが k から出ている矢の上に移動する。 k 上のループ $k(1) \rightarrow k(2) \rightarrow \dots \rightarrow k(m) \rightarrow k(1)$ で $k(i) \rightarrow k(j)$ ($i \neq j$) なるものを単純ループと呼ぶ。 k 上のパス $k(1) \rightarrow \dots \rightarrow k(m)$ に対して k 上のパス $k(1) \circ k(2), \dots, k(m-1) \circ k(m)$ が対応する。もとのパスが単純列の時単純列、単純ループの時単純ループという。又簡単の為にポインタースが " $k \circ k$ 上にいたことがある" というかわりに、"ポインタースが k にいたことがある" という。 k に入っている必要な矢の上の白丸にポインタースがそろっている時、 k が excite しているという。同時にいくつものノードが excite する。従って現在のポインタースの位置だけでなくノードの実行における相対的な速さも又次のポインタースの位置に影響する。我々はノード実行の速さについては知らないわけであるから、起こりうるすべてのポインタースの動作を考えることになる。あるポインタースの動作にお

いて、 hok の上にポインタが存在し、 h から矢がでているのにそこにいつづけたままでいる時、 h を illegal halting node といい、全ての h がそうでない時に H に illegal state は生じないという。又あるポインタの動作において、 hok 上にポインタがあり、同時に h が excite している時に h に multi call が生じたという。全てのノード h に multi call が生じない時に H に multi call が生じないという。POG H が次の条件を満たす時 Tailless POG (TPOG) という。(T) H 上のループ $h(1) \rightarrow \dots \rightarrow h(m) \rightarrow h(1)$ とその一員 $h(i)$ が存在して $h(i) \rightarrow h$ ならば $h \rightarrow h(i)$ である。

定理 1. H を POG とすると、 H に illegal halting は生じない。又 H に multi call の生じない条件は H が TPOG なることである。

§ 3. TPOG の蓄積グラフ

$H = \{h_1, \dots, h_n\}$ とし、 H を $H' = \{s_1 h_1, \dots, h_n\}$ をノードの集合として H' を TPOG とする。 Z を非負整数の集合 $\{0, 1, 2, \dots\}$ とし Z^H により H から Z への写像の全体を示す。 $a \in Z^H$, $h_i \in H$ について $a(h_i) \in Z$ を a_i で示す。 a をポインタの蓄積実行状態 (あるいは簡単に状態) といい、ノード h_i a_i 回実行されたことを示す。ポインタが START にい

る時は $a = (0, \dots, 0)$ であり、ポインタの移動とともにノードが実行されることに対応する成分がカウントアップされていく。ノード k_i の連結成分が2つ以上のノードからなる時 k_i をループエレメントという。 $k_i \gg k_j$ により $k_i > k_j$ かつ $k_i > k > k_j$ となる k が存在しないことを示す。

補題5 M を TPOG とし、ポインタが H 上を移動していくとする。ある時点においてポインタの蓄積実行状態が a であったとする。ノード k_j が excite している為の必要条件は次の (イ) 又は (ロ) が成り立つ時である。

(イ) $a_j = 0$ の時 全ての $k_i \gg k_j$ について

$$a_i = 1$$

(ロ) $a_j \geq 1$ の時 k_j はループエレメントであって全ての $k_i \gg k_j$ なるループエレメント k_i に対して

$$a_i = a_j + 1$$

又全ての上向き $k_i \rightarrow k_j$ について

$$a_i = a_j$$

記号 状態 a で excite しているノードの集合を $E(a)$ で示す。 $a' \in Z^H$ を

$$\begin{aligned} a'_i &= a_i & k_i &\notin E(a) \\ &= a_i + 1 & k_i &\in E(a) \end{aligned}$$

と a' を定義すると、 a' も蓄積状態である。 a' を a

の目標状態という。さて $k_i \in E(a)$ であり、状態が a から b に移り $a_i = b_i$ であったとする。すなわち、ノード k_i は a で *excite* していたが、他のノードが先に実行されて状態が b に変わった時を考える。ポインタは消えてしまうことはないから、ノード k_i は b でも *excite* している。すなわち $b_i = a_i$ である。これは蓄積状態の集合 (今後 V で示す) が *Semimodular* になることを示している。実際には次の補題が成り立つ。

補題 6. M を TPOG とし、ポインタが M 上を移動していくとする。ポインタの蓄積状態の集合を V で示す。 V は *distributive* になる。すなわち次の条件をともに満足する。

- (1) $(0, \dots, 0) \in V$
- (2) $a, b \in V$ ならば $a \vee b \in V$
- (3) $a, b \in V$ ならば $a \wedge b \in V$
- (4) $a, b \in V$ で $a \ll b$ ならば、ある $i \in H$ について $b = a + \delta^i$ となる。

ここで $a \vee b$, $a \wedge b$ は次のように定義される。

$$\begin{aligned} \mathbb{Z}^H \text{ の点である } (a \vee b)_i &= \max\{a_i, b_i\} \\ (a \wedge b)_i &= \min\{a_i, b_i\} \end{aligned}$$

定義 Z^H の点 a, b と自然数 P について、

$a_i \equiv b_i \pmod{P}$ for all $i \in H$ とするとき $a \equiv_P b$ と書く。 Z^H の部分集合 V と V の点 a について、

$Va = \{b - a, b \in V\}$ と定義する。

V が次の性質をもつとき P 進デジタルという。

(pd): $a, b \in V$ で $a \equiv_P b$ ならば $Va = Vb$.

定理 2 \mathcal{A} を TPOG としその蓄積状態の集合を V とする。 V は P 進デジタルなディストリビューティブ集合である。

附記 定理 2 の V は実際は非常に特殊なディストリビューティブ集合になっている。それにもかかわらず TPOG はイリヤック II で使用された指図書を、特殊な一部のものを除いて、記述できる。定理 2 より TPOG \mathcal{A} を実現する論理回路の存在が帰結されるが、より簡潔な回路の実現をオートマトンを利用する観点から考察中である。