FIFOキューを同期手段とする並列プログラムについて(II)†

——並列プログラムの待ちなし変換——

有田 五次郎†† 荒木 啓二郎††

オペレーションを点,制御の移動を枝とする非巡回定向グラフで表現される並列プログラムのあるクラスは,各プロセッサが先着順(FCFS)で各オペレーションを実行するとき,自然に同期がとれる.木構造グラフとなるこのような並列プログラムを待ちなし並列プログラム(SPP)と呼ぶ.FCFSはFIFOメモリを用いてハードウェアで容易に実現でき,SPPはMIMD型高多重並列処理における同期問題を解決する一つの手段となる.本論文ではまず並列プログラムをグラフ表現し,確定性,同値性等の幾つかの性質について論じる.ここで用いる定義は並列プログラムの物理的性質,すなわちセグメンテーションやプロセッサ割当てを含んでいる.次に木構造グラフで表現されるSPPを定義してその性質について考察し,最後に一般の並列プログラムをそれと同値なSPPに変換する手順を与える.SPPはデータ依存関係に基づいて構成されており,一種のデータ駆動型プログラムとみなすことができる.しかしこれらはオペレーションとその間のコントロールフローで表現されており,FIFOキューをハードウェアでもったマルチプロセッサシステムによって効率よく実行できる.

1. はじめに

FIFOキューを同期手段とする待ちなし並列プログラム(self synchronizing parallel program, 以後SPPと書く)の概念についてはすでに述べた[1].木構造グラフで表現されるSPPは,各プロセッサが先着順(first come first served, 以下FCFSと書く)に各オペレーションを実行するとき自然に同期がとれる並列プログラムとなっている.高多重並列処理においては多重化の効率を上げるため高速な同期手段を必要とするが,割込み機構を用いてソフトウェアで実装されている従来の同期操作では速度の点で十分ではない.SPPの同期機構であるFCFSはFIFOメモリを用いてハードウェアで容易に実現でき,SPPは高多重並列処理における同期問題を解決する一つの手段となる.

本論文では並列プログラムを非巡回定向グラフで表現し,これからSPPを得る一つの手順を与える.ここで考えているハードウェアモデルはメモリを共有したモジュラ型の高多重並列計算機で,各モジュールは実行制御のためのFIFOキューをもったプロセッサとメモリとから成り,おのおのはシステムアドレス(プロセッサ番号)によって識別される.アドレスはシステムアドレスとメモリアドレスの2次元アドレスで,単一のアドレス空間を構成する.

このようなシステムの上で動作する並列プログラムを表現するためには,プログラムの論理的構造のみならずプログラムのメモリ内での配置等,物理的構造も記述できなければならない.並列プログラムを表現しその動作を記述する研究は多数あるが[2)~5)],これらはいずれもプログラムの論理的構造のみを扱っており,プロセッサ割当てやプログラムのセグメント化などを考える上で十分ではない.本論文では物理的構造を含んだ並列プログラムを定義し,並列プログラムの論理的性質を保存したまま物理的構造を変える変換について述べ,一般の並列プログラムからSPPを得る手順を示す.

2. 並列プログラムのグラフ表現

この章では3章および4章の準備として,並列プログラムをグラフ表現しその性質を考察する.ここで議論される確定性,等価等の概念はKarp[3)],Keller[5)]と同じであるが,この章の目的は並列プログラムにおける諸概念と,本論文で用いている用語・記法の説明であり,厳密な議論は行わない.

2.1 逐次的プログラム

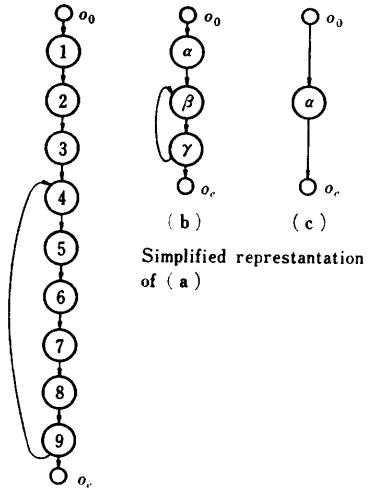
逐次的プログラムSPを次の5組で表す.

$$SP=(M, O, o_0, o_e, S) \qquad (2.1)$$

ここで各要素はそれぞれ次のような意味をもつ.

† On a Parallel Program with Synchronizing Mechanism Using FIFO Queue (II)—A Transformation of a Parallel Program to the Self Synchronizing Parallel Program—by ITSUJIRO ARITA and KEIJIRO ARAKI (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科



(a) Graph representation of program A

図 1　逐次的プログラム
Fig. 1　Sequential program.

M：記憶領域を表す番地空間

O：演算または操作を表すオペレーション o の集合
各オペレーションは入力変数と出力変数をもち，これらを $D(o)$, $R(o)$ と書くと，

$$D(o)\subset M, R(o)\subset M$$

o_0, o_e：O の特別な要素．このプログラムの入口と出口．

S：(o_i, o_j) の形をした O の上の 2 項関係．プログラム上の制御の移動を表す．

SP は一つの定向グラフを表す．たとえば次のプログラムは**図 1**(a) のように表せる．

```
      プログラム　A
①        H=(B-A)/N
②        S=0.
③        X=A
④   10   F1=F(X)
⑤        F2=4.*F(X+H)
⑥        F3=F(X+2.*H)
⑦        S=S+H/3.*(F1+F2+F3)
⑧        X=X+H
⑨        IF(X.LE.B) GO TO 10
```

このとき，

$$O=\{o_0, ①, ②, \cdots, ⑨, o_e\},$$

$$S=\{(o_0, ①), (①, ②), (②, ③), \cdots, (⑧, ⑨), (⑨, ④), (⑨, o_e)\}$$

であり，

$$D(①)=\{A, B, N\}, R(①)=\{H\},$$

$$D(②)=\{0.\}, R(②)=\{S\}, \cdots$$

である．

ただし $A, 0.$ 等は，通常のプログラミング言語が与えている意味と同様，変数 A，定数 0. 等が割り当てられたメモリ領域を表すものとする．

SP をグラフとして見たとき o_i を点，(o_i, o_j) を枝と呼ぶ．$(o_i, o_j)\in S$ のとき $o_i \gg o_j$ と書き，o_i は o_j の親，o_j は o_i の子と呼ぶ．複数の親，複数の子をもつ点はそれぞれ合流点，分岐点である．また o_i から o_j への定向パスがあるとき $o_i \geqslant o_j$，そうでないとき $o_i \not\geqslant o_j$ と書き，閉じた定向パスをループと呼ぶ．

$o_i \geqslant o_j$ に対して次式が成立するとき，o_j は o_i にデータ従属であると呼ぶ．ここに ϕ は空集合を表す．

$$D(o_i)\cap R(o_j)\neq\phi \vee D(o_j)\cap R(o_i)\neq\phi \vee R(o_i)\cap R(o_j)\neq\phi. \quad (2.2)$$

また，$o_i \geqslant o_j$ に対して次の条件が成立するとき，o_i と o_j は互いにデータ独立であると呼ぶ．

(1)　o_j は o_i にデータ従属ではない．

(2)　o_i から o_j へ向かう定向パスが分岐点および合流点を含まない．

(3)　o_i から o_j へ向かう定向パスの上に，o_k が o_i にデータ従属でかつ o_j が o_k にデータ従属であるような点 o_k が存在しない．

データ独立なオペレーションは互いに無関係で並列実行が可能である．

SP は一つの逐次的プログラムを表すが，これをまた一つのオペレーションとみなすことができる．たとえば，プログラム A は {①, ②, ③}, {④, ⑤, ⑥, ⑦, ⑧}, {⑨} の三つのプログラムから成っているとして図 1 (b) のように表すこともできるし，全体を一つのオペレーションとして (c) のように表すこともできる．(c) のように表現したときには

$$D(\alpha)=\{A, B, N, 0., 4., 2., 3.\},$$

$$R(\alpha)=\{S\}$$

と考えることができる．

このように全体のプログラムの断片である SP をプログラム片（以後 P-片と書く）と呼ぶ．

2.2　並列プログラム

SP において互いにデータ独立なオペレーションは並列実行が可能である．SP から得られる，P-片をオペレーションとする並列プログラム，PP を考える．このとき SP は条件分岐およびループをもたないものとする．SP が条件分岐やループをもつときはこれ

らをP-片の内部に入れてしまうか，条件分岐を含まない区間ごとに分割して考える．

プロセッサとメモリから成るn個のモジュールをもつ並列処理システムの，モジュールiに割り当てられたj番目のP-片を${}_ip_j$と書き，P-片をオペレーションとする並列プログラムPPを前と同様，次の5組で表す．

$$PP=(M, P, p^0, p^e, S). \qquad (2.3)$$

記憶領域がモジュールごとに分割されており，モジュールiの記憶領域をM_iとして$M=M_1+M_2+\cdots+M_n$であり，PがP-片${}_ip_j$の集合であることを除いて，各要素の意味はSPと同じである．

PPはプリフィックスiによって類別（ラベル付け）された点から成る非巡回定向グラフである．${}_ip_j$がプロセッサiによって実行されるとするとPPは並列プログラムを表す．

類別された点をもつすべての非巡回定向グラフが並列プログラムとして意味をもっているわけではない．次の性質をもつPPを整形並列プログラムと呼ぶ．

（1） 一つの始点と一つの終点をもち，始点から終点に至るすべての定向パスに含まれる点はこの二つ以外に存在しない．

（2） ${}_ip_j \geqslant {}_kp_l$ ならば ${}_kp_l \not\geqslant {}_ip_j$．

（3） ${}_ip_j \not\geqslant {}_kp_l \wedge {}_kp_l \not\geqslant {}_ip_j$ ならば $i \neq k$．

（4） ${}_ip_j \gg {}_kp_l \wedge {}_ip_j \geqslant {}_rp_s \wedge {}_rp_s \geqslant {}_kp_l$ となる点 ${}_ip_j, {}_kp_l, {}_rp_s$ は存在しない．

関係$\geqslant$はPの上の半順序であり，${}_ip_j \geqslant {}_kp_l$または${}_kp_l \geqslant {}_ip_j$のとき${}_ip_j, {}_kp_l$は直列，${}_ip_j \not\geqslant {}_kp_l \wedge {}_kp_l \not\geqslant {}_ip_j$のとき${}_ip_j, {}_kp_l$は並列と呼ぶ．（3）は並列なオペレーションは異なるプロセッサに割り当てられていることを示している．

SPの適当な区間から得られるPPは整形並列プログラムである．たとえばプログラムAからは**図2**に示す三つの整形並列プログラムが得られる．

枝$({}_ip_j, {}_kp_l)$を$i=k$のとき直行枝，$i \neq k$のとき並行枝と呼ぶ．直行枝は同一プロセッサ上の制御の移動，すなわち逐次的動作を表し，並行枝は異なるプロセッサへの制御の移動，すなわち他のプロセッサ上のP-片の実行開始を意味する．

出度数nの分岐を並列度nのAND分岐，入度数nの合流を待ち数nのAND合流と呼び，とくに直行枝，並行枝各1本をもつAND合流を単純合流と呼ぶ．図2においてp, qは並列度3のAND分岐，s, tは待ち数3のAND合流，uは単純合流である．

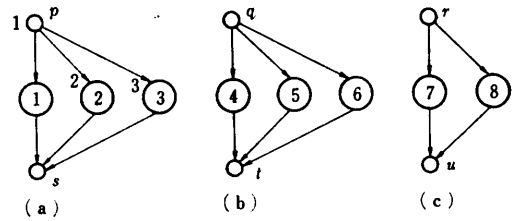


図2 プログラムAから得られる整形並列プログラム
Fig. 2 Well formed parallel programs obtained from program A.

PPを図で表現するときは直行枝，並行枝を区別するため同一プロセッサで実行されるP-片，すなわち同一プリフィックスをもつP-片を縦に並べて書く．また必要があれば，図2（a）のようにプリフィックス（プロセッサ番号）を書き加える．

制御の移動のみに関与するP-片を制御ノード，その他のP-片をオペレーションノードと呼ぶ．前者にはAND分岐，AND合流，無条件分岐が含まれ，後者には演算，入出力操作等が含まれる．

2.3 並列プログラムの確定性

分岐，合流を含まないSPの適当な区間は，互いにデータ独立なオペレーションを異なるプロセッサ番号でラベル付けすることによってPPに変換できる．たとえば次のプログラムはプログラムAを書き直したものであるが，このプログラムの⑦〜⑬は**図3**（a）のようなPPに変換できる．

プログラムB

```
①        H=(B-A)/N
②        S=0.
③        X=A
④        H2=2.*H
⑤        H3=H/3.
⑥   10   CONTINUE
⑦        X1=X+H
⑧        X2=X+H2
```

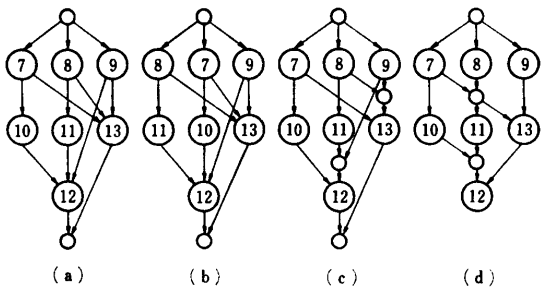


図3 同値および等価な並列プログラム
Fig. 3 Equivalent and equal parallel programs.

⑨ $F1=F(X)$

⑩ $F2=4.*F(X1)$

⑪ $F3=F(X2)$

⑫ $S=S+H3*(F1+F2+F3)$

⑬ $X=X1$

⑭ IF(X. LE. B) GO TO 10

このとき得られた PP においては，すべての並列な P-片 ${}_ip_j, {}_kp_l$ 間に次式が成立する．

$$D({}_ip_j)\cap R({}_kp_l)=\phi\wedge R({}_ip_j)\cap D({}_kp_l)$$
$$=\phi\wedge R({}_ip_j)\cap R({}_kp_l)=\phi \qquad (2.4)$$

このような PP では，並列な P-片の実行開始の順番，実行に要する時間に無関係に実行結果が定まり，これは元の SP の実行結果に等しい．このような PP を確定的 PP と呼ぶ[3]．非確定的 PP は同じ入力データに対して常に同じ結果を与えるとはかぎらない．したがって PP は確定的でないとプログラムとしての意味をもたない．

2.4 並列プログラムの変換

図3(a)の PP は(b)のように書くこともできる．(a),(b)はグラフ的には同型であるが，オペレーションに対するプロセッサ割当てが違っているので異なる PP である．これはまた，適当な制御ノードを加えて(c)のように書くこともできる．(a),(b),(c)の各オペレーションがプログラムBの各文に対応しているとすると，これらはプログラムBの異なる並列表現と考えることができる．ここではこのように一つの SP の異なる表現であり，したがって同じオペレーションノードから成る PP について考察する．

同じオペレーションノードから成る二つの並列プログラム PP, PP′ において，対応するオペレーションノード間の順序が逆になるものがないとき，すなわち ${}_ip_j$ と ${}_{i'}p_j'$, ${}_kp_l$ と ${}_{k'}p_l'$ が対応するオペレーションノードとして，${}_ip_j\geqslant{}_kp_l$ であれば ${}_{k'}p_l'\geqslant{}_{i'}p_j'$, ${}_kp_l\geqslant{}_ip_j$ であれば ${}_{i'}p_j'\geqslant{}_{k'}p_l'$ であるとき，この二つのプログラムは弱等価であるという．

弱等価なプログラムがいずれも確定的であるときそれらは等価であるといい，さらに対応するオペレーションノード間の順序がすべて等しいとき同値という．

図3(a)～(d)がプログラムBを表しているとするとこれらはすべて確定的で，(a)は(b),(c)と同値であり，(d)は(a),(b),(c)と等価である．等価および同値な並列プログラムの実行結果は等しい．

一つのオペレーションに並列であるオペレーションの個数をそのオペレーションの並列度と呼ぶ．逐次的プログラムの各オペレーションの並列度は0であり，これと等価で各オペレーションの並列度が最大になるような確定的 PP が存在する[5]．

図3(d)には(a)と比較して⑦>⑪，⑬>⑫の二つの順序が加わっている．このように新しい順序が加わる変換を順序を強める変換と呼び，(d)は(a)より強い順序をもつという．同様に(a)から(b),(a)から(c)のようにオペレーション間の順序を保存する変換を同値な変換と呼ぶ．同値な変換は確定性，並列性を保存し，順序を強める変換は確定性を保存するが並列性を減少させる．

3. 待ちなし並列プログラム

図4(a),(b)のような，すべての合流が単純合流である同値な並列プログラムを考える．これから合流点に入る直行枝を除くと，(a′),(b′)のように木構造のグラフが得られる．このグラフは待ちを表す合流点を含んでいない．このようなグラフで表されるプログラムを待ちなし並列プログラム (SPP) と呼ぶ．

各プロセッサが FCFS で実行を行うとき (b′) の P-片は常に正しい順に実行されるが，(a′) では ②，④ の実行時間の長さによって ⑤，⑥ の実行順序が変わる．すなわち (b′) は FCFS の仮定のもとで確定的であり，(a′) はそうではない．

図4(a),(b)は単純合流のみをもつ同値な PP であるがこの二つは異なる性質をもつ．この性質を調べるため制御路および対応分岐点の概念を導入する．

単純合流点では流入する並行枝を，その他の点では流入する枝をその点の制御枝と呼び，制御枝のみから成る定向パスを制御路と呼ぶ．n 本の並行枝を含む制御路の長さを，先頭が並行枝のときは n，先頭が直行枝のときは $n+1$ と定義する．${}_ip_j$ から ${}_kp_l$ に至る長さ n の制御路があるとき，${}_kp_l$ の ${}_ip_j$ からのプロセッサ距離（以後，距離と書く）が n であるという．

たとえば図4において ① から ⑥ に至る制御路は，

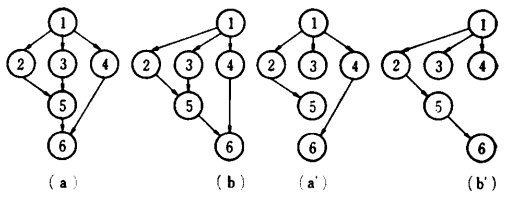


図4 待ちなし並列プログラム
Fig. 4 Self synchronizing parallel programs.

(a)では (①, ④, ⑥) で距離は 2 であり，(b)では (①, ②, ⑤, ⑥) で距離は3である．また，③から⑥に至る制御路は(a)，(b)ともに定義されない．

単純合流点の二つの親に至る定向パスをもち，その合流点の直行枝側の親に至る距離が最も短い分岐点を，その合流点の対応分岐点と呼ぶ．以後簡単のため，対応分岐点と直行枝側の親との距離が n である合流点を距離 n の待ち，距離1の待ちを単純な待ちと呼ぶ．

図 4 (a) の⑤，⑥の対応分岐点はいずれも①であり，⑤ は単純な待ち，⑥ は距離2の待ちである．一方(b)では⑤，⑥ ともに単純な待ちになっている．

単純な待ちでは，合流点の直行枝側の親の制御枝による制御の移動が，合流点の制御枝による制御の移動より必ず時間的に先に起こるが，単純でない待ちではそれは保障されない．たとえば図4(b)では，制御の移動 (①, ④) は制御の移動 (⑤, ⑥) に必ず先行するが，(a)では制御の移動 (②, ⑤) と (④, ⑥) の時間的順序は，オペレーション② および④ の実行時間に依存して変わる．すでに述べたように確定的でない並列プログラムは常に正しい結果を与えるとはかぎらない．したがって SPP も確定的でなければ意味をもたない．

以上の議論により，単純な待ちのみを含む PP から得られる SPP が FCFS の仮定のもとで確定的であることが明らかになった．

4. 待ちなし並列プログラムへの変換

この章では一般の PP を同じ並列度をもつ確定的な SPP に変換する手順を与える．簡単のため手順，場合分け等はすべて図のみを用いて説明する．

4.1 PP 上の基本的変換操作

変換手順の記述を簡単にするため PP 上の次の三つの変換操作を定義する．

(1) move ($_ip_j$; k): P-片列の移動

図 5 (a) に示すように，$_ip_j$ (同図②，以下同様) とそれに直行枝で結ばれている P-片列を異なるプロセッサ上に移動する．

(2) insert ($_ip_j$, $_ip_k$; $_mp_n$, $_rp_s$; $_ip_x$): 制御ノードのそう入

図5(b)に示すように直行枝上に制御ノード(X)をそう入し枝を付け替える操作である．この操作は $m \neq i$ のときにのみゆるされる．また $r=i$ の場合は $_ip_k$ (③) と $_rp_s$ (④) は同一ノードとする．なお，もし ($_mp_n$, $_ip_k$)，($_ip_j$, $_rp_s$) 等の枝 (②から③，①から④への枝) がある場合はこれらは変換の際に除去される．

(3) append ($_ip_j$; $_mp_n$, $_rp_s$; $_ip_x$): 制御ノードの追加

図5(c)のように，直行枝をもたない $_ip_j$ (①) の後に制御ノード $_ip_x(X)$ を加え枝を付け替える操作である．この操作は $m \neq i \neq r$ のときにゆるされる．また不要な枝の除去に関しては(2)と同じである．

4.2 単純合流への変換

SPP はすべての合流が単純合流である PP から得られる．まず待ち数 n の合流を単純合流に変える変換を考える．制御ノードをそう入することによって，待ち数 n の AND 合流を待ち数 $n-1$ の AND 合流に変換でき，この変換が同値な変換であることは明らかである(図3(c)参照)．これを続けることにより一般の PP は待ち数2の AND 合流だけをもつ同値な PP に変換できる．よって以後このような PP についてのみ考察する．

場合分けを行うため PP の上の次のような関係 $\sqcap$ を考える．

(1) $_ip_j \sqcap _ip_j$

(2) $_mp_n \gg _ip_j \wedge _mp_n \gg _kp_l \rightarrow _ip_j \sqcap _kp_l$

(3) $_ip_j \sqcap _kp_l \wedge _kp_l \sqcap _rp_s \rightarrow _ip_j \sqcap _rp_s$

関係 $\sqcap$ は PP の上の同値関係で，これにより PP の点を類別することができる．ある類に属す点の親で

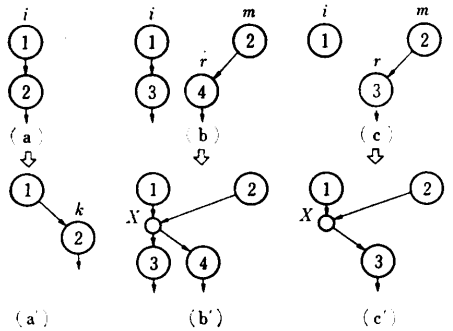


図 5 PP 上の基本的変換操作
Fig. 5 Elementary transformation on PP.

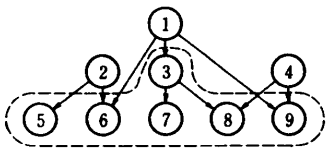


図 6 同じ類に属す点
Fig. 6 Nodes in same class.

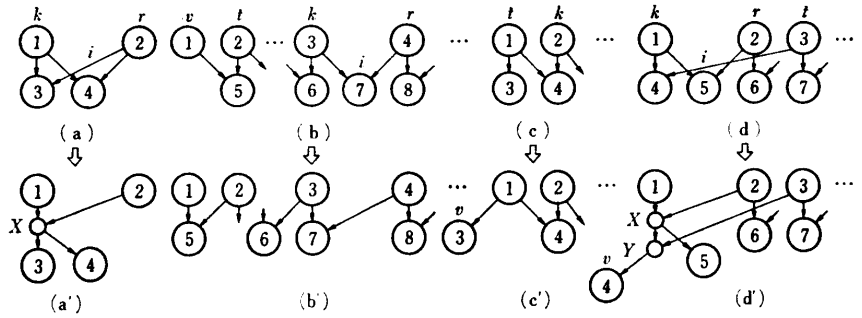


図 7 単純合流への変換
Fig. 7 Transformation to simple join.

その類に属さない点の集合をその類の親と呼ぶ．たとえば図6で点⑤を含む類は {⑤,⑥,③,⑦,⑧,⑨} であり，この類の親は {①,②,④} である．

[変換 I] 単純合流への変換

$_ip_j$ が単純でない合流点である，すなわち枝 $(_kp_l, _ip_j)$, $(_rp_s, _ip_j)$ が存在し $k \neq i \wedge r \neq i$ とする．このとき $_ip_j$ を含む類の性質に従って次の変換操作を考える．

(A)図7(a)のように $_ip_j$ (④) を含む類のなかに $_ip_j$ に同じ待ちを表す単純合流点 $_kp_l$ (③) がある場合は，次の操作により制御ノード $_kp_x(X)$ をそう入し，(a′) のように変換する．

insert $(_kp_l, _kp_m; _rp_s, _ip_j; _kp_x)$.

(B)図7(b)のように $_ip_j$ (⑦) を含む類の親が直行枝側に子をもたない点 $_vp_w$ (①) を含む場合は，次の操作によりP-片列を移動し，直行枝側に子をもたない点を変える．

move $(_ip_u; v)$.

これを続けることにより図7の (b′) の⑤,⑥,⑦ のように P-片の移動が行われ，単純でない合流が単純合流に変わる．

$_ip_j$ を含む類の親がすべて直行枝側の子をもつ場合は，以下の二つの場合に分けられる．

(C)図7(c)のように $_ip_j$ を含む類のなかに合流点でない点 $_ip_u$ (③) が存在するときは，v を新しいプロセッサ番号として次の操作で (c′) となり(B)の場合に帰着される．

move $(_ip_u; v)$.

(D)図7(d)のように $_ip_j$ を含む類のうちの異なる待ちの数がこの類の親の数より多い場合は，(A)～(C)の操作を行った後にも単純でない合流が残る．この場合は $_ip_j$ (⑤) の一つの親 $_kp_l$ (①) およびその子 $_kp_m$ (④) に対し，引き続いた次の操作を行い (d′) のように変換する．ただし v は新しいプロセッサ番号とする．

insert $(_kp_l, _kp_m; _rp_s, _ip_j; _kp_x)$,
insert $(_kp_m, _kp_x; _ip_u, _kp_m; _kp_y)$,
move $(_kp_m, v)$.

(A)～(C)は同値な変換であるが，(D)は順序を強める変換になっている．しかしこの強められた順序は，(D)の形の変換の際にそう入された制御ノード (図7 (d′) の X と Y) の間の順序としてのみ表現されており，これは後の変換の際に考慮される．

単純でない合流点を含む類の異なる待ちの数が，その類の親の数より少なければ(A)～(C)の変換で，多い場合は(D)の変換を加えることによって単純でない合流を単純合流に変換することができる．よってすべての PP を単純合流のみをもつ等価な PP に変換することができる．

4.3 待ちの距離を短くする変換

合流点を移動することによって待ちの距離が変わる．

[変換 II] 待ちの距離を短くする変換(I)

図8(a)，(b) のように $_ip_j$ (⑤) が単純合流点

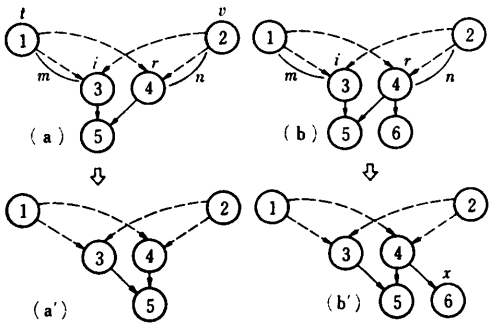


図 8 制御路の長さを短くする変換
Fig. 8 Transformation to minimize the control path length.

で，直行枝側の親 ${}_ip_k$ (③) および並行枝側の親 ${}_rp_s$ (④) に至る定向パスをもつ分岐点 ${}_ip_u$ (①)，${}_vp_w$ (②) があるとする．ただしおのおのの定向パスの1本はそれぞれ長さ m, n の制御路であるとし，また ${}_iP_u$ (①) と ${}_vP_w$ (②) は同一の点であってもよいものとする．${}_ip_u$ (①) は ${}_ip_j$ (⑤) の対応分岐点である．このとき，$m>n$ であれば，次の変換を行って対応分岐点を ${}_vp_w$ (②) に変える．

(A) 図8(a)のように ${}_rp_s$ (④) が直行枝側に子をもたない場合は，

move $({}_ip_j; r)$.

(B) (b)のように ${}_rp_s$ (④) が直行枝側に合流点でない子 ${}_rp_l$ (⑥) をもつ場合は，x を新しいプロセッサ番号として，

move $({}_rp_l; x)$,

move $({}_ip_j; r)$.

変換Ⅱは明らかに同値な変換である．次に制御ノードをそう入して距離の短い待ちを作る変換を考える．

[変換Ⅲ] 待ちの距離を短くする変換(Ⅱ)

図9 (a)，(b)，(c) のように ${}_ip_j$ (②) が距離 n の単純合流点であり，親 ${}_ip_k$ (①)，${}_rp_s$ (④) をもっているとする．このとき ${}_ip_k$ (①) への制御路の長さが1である点 ${}_ip_u$ (③) について，次の変換を行って制御ノード ${}_ip_x(X)$ をそう入する．

(A) 図9(a) のように ${}_ip_u$ が直行枝側に子をもたない場合は次の操作により (a′) のように変換する．

append $({}_ip_u; {}_rp_s, {}_ip_j; {}_ip_x)$.

(B) (b)のように ${}_ip_u$ (③) が直行枝側に合流点でない子 ${}_ip_l$ (⑤) をもつ場合は，v を新しいプロセッサ番号として次の操作で(A)の場合に帰着される．

move $({}_ip_l; v)$.

(C) (c)のように ${}_ip_u$ (③) が直行枝側に合流点である子 ${}_ip_l$ (⑤) をもつ場合は，${}_wp_m$ (⑥) を ${}_ip_l$ (⑤) の並行枝側の親，v を新しいプロセッサ番号として，次の引き続いた操作によって (c′) のように変換する．

insert $({}_ip_u, {}_ip_l; {}_rp_s, {}_ip_j; {}_ip_x)$,

insert $({}_ip_x, {}_ip_l; {}_wp_m, {}_ip_l; {}_ip_y)$,

move $({}_ip_y; v)$.

(A)，(B)は同値な変換であり(C)は順序を強める変換である．しかしここでも変換Ⅰの場合と同様，強められた順序はそう入された制御ノード (図9(c)の X, Y) 間の順序としてのみ表現されている．

この変換により距離 n の待ちはなくなり距離 $n-1$ の待ちに変えられる．

4.4 待ちなし並列プログラムへの変換

変換Ⅰ，Ⅱ，Ⅲを用いて PP から SPP を得る手順を示す．変換手順を適用する点の順序は適当でよいが，たとえば入口 p^0 からの制御路の短い順に，制御路の長さが同じであればプリフィックス，サフィックスの小さい順に行う．なお変換途中で無意味な制御ノードが生じた場合は，**図 10** に示すようにこれらを除去する．

[変換Ⅳ] 待ちなし並列プログラムへの変換

(A) 変換Ⅰによってすべての単純でない合流を単純合流に変換する．

(B) 変換Ⅱによって待ちの距離を最小にする．

(C) 変換Ⅲをくり返し適用し単純でない待ちを単純な待ちに変換する．このとき変換Ⅰの(D)および変換Ⅲの(C)の操作によって付加された制御ノードに関

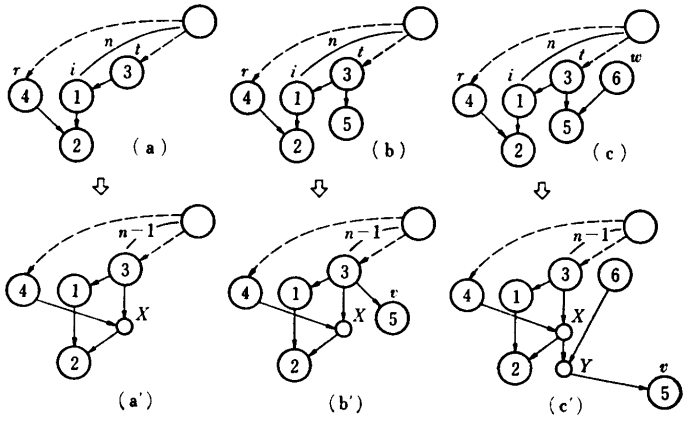


図 9 制御路の長さを1減らす変換
Fig. 9 Transformation to decrease the control path length.

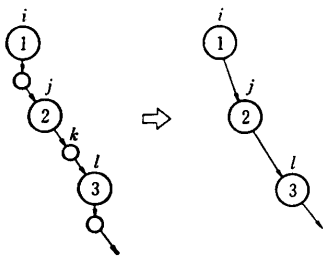


図 10 無効な制御ノードの除去
Fig. 10 Deletion of the noneffective control nodes.

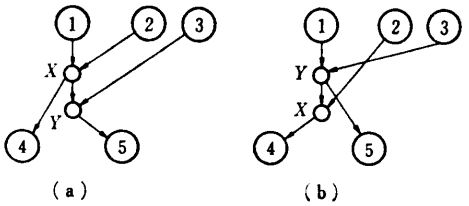


図 11 そう入された制御ノードの入れ替え
Fig. 11 Transposition of the inserted control nodes.

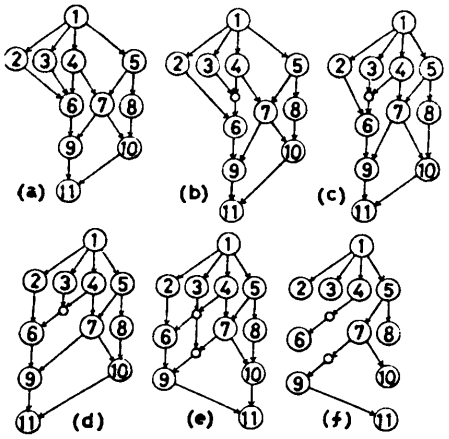


図 12 変 換 例
Fig. 12 An example of the process to obtain SPP.

しては，**図 11** に示すように順序を入れ替えて，1番上の待ち（図11(a)の X および(b)の Y）についてのみ単純な待ちに変換する．

(D) すべての単純でない待ちに(B)，(C)を適用した後，待ちを構成している直行枝をすべて除き，制御枝だけをもつ木構造プログラムに変換する．

変換IVによって得られた SPP は元の PP と同じ並列度をもつ．なぜなら変換の各ステップで強められた順序はすべて付加された制御ノード間の順序としてのみ表現されていたが，(C)ではこの強められた順序によって生ずる待ち（図11(a)の Y および(b)の X）は単純な待ちに変換していない．したがって(D)で直行枝を除くと，これらの制御ノード間の順序は消え，元のプログラムと同じ順序をもつことになる．

図 12 はこの変換の各ステップを簡単な例について示している．(a)は変換前の PP，(b)は待ち数を2にした PP である．これに変換 IV の(A)を適用して(C)が得られ，これの⑥に対して(B)を適用して(d)が，さらに⑨に対して(C)，⑪に対して(B)を適用して(e)が得られる．(e)から待ちに入る直行枝を除くことにより SPP，(f)が得られる．

FCFS の仮定の下で(f)の各オペレーションの実行順序が(a)のそれと同じであることは容易に確かめられる．

5. むすび

以上並列プログラムをグラフ表現し待ちなし並列プログラムを定義して，一般の並列プログラムを待ちなし並列プログラムに変換する手順を与えた．ここで示した手順は待ちなし並列プログラムを得る一つの方法を与えているが，使用プロセッサ数を最小にする変換，入口 p^0 から出口 p^e に至る制御路の長さを最短にする変換等を与えることも重要であろう．また文献1)に述べたように，効率よい並列実行を行うためには並列化されるオペレーションはなるべく大きく，同程度の実行時間をもつことが望ましい．逐次的プログラムから効率のよい待ちなし並列プログラムを作成する並列コンパイラの研究等も必要となろう．

謝辞 本論文をまとめるに当たり九州大学工学部牛島和夫教授，宇津宮孝一助教授には種々のご助言をいただいた．ここに記して謝意を表す．

参考文献

1) 有田五次郎：FIFO キューを同期手段とする並列プログラムについて(I)，一待ちなし並列プログラム一，情報処理学会論文誌，Vol. 24, No. 2, pp. 221-229 (1983).
2) Bear, J. L.: A Survey of Some Theoretical Aspects of Multiprocessing, *Comput. Surv.*, Vol. 5, No. 1, pp. 31-80 (1973).
3) Karp, R. M. and Miller, R. E.: Parallel Program Schemata, *J. Comput. Syst. Sci.*, Vol. 3, No. 2, pp. 147-195 (1969).
4) Karp, R. M. and Miller, R. E.: Properties of a Model for Parallel Computations: Determinacy, Termination, Queuing, *SIAM J. Appl. Math.*, Vol. 14, No. 6, pp. 1390-1411 (1966).
5) Keller, R. M: On Maximally Parallel Schemata, IEEE 11-th Annual Simposium on Switching and Automata Theory, pp. 32-50 (1970).

(昭和57年7月6日受付)
(昭和57年10月4日採録)