数理解析研究所講究録
第322巻 1978年 99-116

並列プログラム図式に関するいくつかの性質

名大・工　山下 雅史
三重大・工　稲垣 康善
名大・工　本多 波雄

1. はじめに

並列プログラム，OS等の研究が進むにつれて，並列処理プログラムを記述し，解析するための，いくつかのモデルが提案され試みられてきた．しかし，それらのいずれも，つぎの諸点に対して十分に答えているとは思えない．

(1)実際にインプリメントされている，例えば Concurrent-PASCAL の様な高級言語との関係を容易に示すことができるか．

(2)制御構造，及び，プロセスの記述を含むプログラム全体の記述が十分に可能であり，且つ形式的な取扱いにも耐えうるか．

(3)各プロセスのスケジューリング方式をも記述することが可能か．

本報告では，以上の諸点に十分答えうるモデルを提案し，デッドロック，相互排他という並列処理プログラムの基本的な問題を考察する．

2. 並列プログラム図式（PPS）の定義と実行

（定義1）PPSは以下で定義される．

PPS = (control part , program part)

control part = (scheduler$_1$, ⋯, scheduler m)

scheduler = (K, Σ, δ, q_0)

ここに，K：状態の有限集合，Σ：入力シンボルの有限集合，δ：$K \times \Sigma$ から K への関数，$q_0 \in K$：初期状態である．

program part = (process$_1$, ⋯, process n)

process = (S_1, ⋯, S_p)

S_i = i . statement

statement = assignment statement / test statement /
halt statement / communication statement

ここに．assignment statement：$Y \leftarrow f(Y)$，test statement：if P then goto i else goto j，halt statement：halt，communication statement：$c \in \Sigma$ である． ∎

PPSの働く仮想計算機（VM）には，2つの関数 exec と next が備っている．next は次の時点で実行するプロセスを定め

る関数であり，exec はTMによる計算を定める関数である．次にPPSの計算状況を定義し，その推移によって，PPSの実行を定義する．

（定義2）ppsの計算状況は4項組（NP, LC, Q, Y）である．ここに，NP：実行されるプロセス番号（$1 \leq NP \leq n$；nはプロセス数），LC：各プロセスのロケーション・カウンタの値を示すベクトル（$LC \triangleq (loc_1, \cdots, loc_n)$），Q：各スケジューラの状態を示すベクトル（$Q \triangleq (q_1, \cdots, q_m)$；mはスケジューラ数），Y：プログラム変数の値を示すベクトル（$Y \triangleq (y_1, \cdots, y_l)$）である． ∎

（定義3）計算状況の推移 $(NP_t, LC_t, Q_t, Y_t) \vdash (NP_{t+1}, LC_{t+1}, Q_{t+1}, Y_{t+1})$ を以下で定義する．

$(LC_{t+1}, Q_{t+1}, Y_{t+1}) = exec(stat_{NP_t loc_{NP_t}}, LC_t, Q_t, Y_t)$

（$stat_{ij}$ はプロセス i ロケーション j のステートメント）．

関数 exec：$(LC', Q', Y') = exec(stat_{ij}, LC, Q, Y)$ は，$stat_{ij}$ が $Y \leftarrow f(Y)$ なら $LC' = LC^{i}_{loc_i+1}$；$Y' = f(Y)$，$stat_{ij}$ が if P then goto k else goto l なら $LC' =$ if P then LC^{i}_{k} else LC^{i}_{l}，$stat_{ij}$ が $c \in \Sigma_k$ なら if $\delta_k(q_k, c) = q'_k$ then $(LC' = LC^{i}_{loc_i+1}$；$Q' = Q^{k}_{q'_k})$，$stat_{ij}$ が halt なら $LC' = LC^{i}_{n}$

（A^{i}_{j} はベクトルAの i 番目の要素に値 j を入れることを示す）．

$NP_{t+1} = next(NP_t, LC_{t+1}, Q_{t+1}, Y_{t+1})$ $(loc_{NP_{t+1}} \neq h)$ である．
$\vdash$, $\vdash\!\!\!\vdash$, $\vdash\!\!\!\vdash\!\!\!\vdash$ は普通の方法で$\vdash$から拡張して定義されるものである． ∎

（定義4）ppsの実行は計算状況の推移列
$(NP_0, LC_0, Q_0, Y_0) \vdash (NP_1, LC_1, Q_1, Y_1) \vdash \cdots$ である．ここに，$LC_0 \triangleq (1, \cdots, 1)$，$Q_0 \triangleq (q_0^1, \cdots, q_0^m)$（$q_0^i$ はスケジューラiの初期状態），$Y_0 \triangleq (\omega, \cdots, \omega)$（$\omega$は未定義記号），$NP_0$: 初期プロセス番号である．
$LC_h \triangleq (h, \cdots, h)$，$NP_h \triangleq (NP, LC_h, Q, Y)$ とすると，計算状況が(NP_h, LC_h, Q, Y)に到達するとppsは停止する． ∎

ppsの例を図1に示す．以降で考察するのは図1のppsに対するデッドロック並びに相互排他問題である．

3. ppsのデッドロック並びに相互排他問題

3.1 デッドロックと相互排他

（定義5）ppsのある計算状況(NP, LC, Q, Y)が任意の$n \geq 0$に対して$(NP, LC, Q, Y) \vdash^n (NP', LC, Q, Y)$となるとき，pps はデッドロック状態にあるという． ∎

2節で述べたpps及びその実行の定義より，関数 exec は，communication statement（com. stat.）以外のステート

process$_1$

1. $\overline{y_1} \leftarrow \overline{x_1}$
2. if $P_1(\overline{y_1})$ then goto 3 else goto $3(\alpha+1)+1$
3. a_1
4. $\overline{ys} \leftarrow f_{11}(\overline{y_1},\overline{ys})$
5. $\overline{y_1} \leftarrow f_{21}(\overline{y_1},\overline{ys})$
:
3α. a_α
$3\alpha+1$. $\overline{ys} \leftarrow f_{1\alpha}(\overline{y_1},\overline{ys})$
$3\alpha+2$. $\overline{y_1} \leftarrow f_{2\alpha}(\overline{y_1},\overline{ys})$
$3(\alpha+1)$. goto 2
$3(\alpha+1)+1$. halt

process$_2$

1. $\overline{y_2} \leftarrow \overline{x_2}$
2. if $P_2(\overline{y_2})$ then goto 3 else goto $3(\beta+1)+1$
3. b_1
4. $\overline{ys} \leftarrow g_{11}(\overline{y_2},\overline{ys})$
5. $\overline{y_2} \leftarrow g_{21}(\overline{y_2},\overline{ys})$
:
3β. b_β
$3\beta+1$. $\overline{ys} \leftarrow g_{1\beta}(\overline{y_2},\overline{ys})$
$3\beta+2$. $\overline{y_2} \leftarrow g_{2\beta}(\overline{y_2},\overline{ys})$
$3(\beta+1)$. goto 2
$3(\beta+1)+1$. halt

scheduler $\mathcal{S}=(K,\Sigma,\delta,q_0)$ $\Sigma=\{a_i\text{'s},b_j\text{'s}\}$

(図1) 本稿で考察する PPS の例

(注) goto i は if true then goto i else goto i の省略形として使用する.

メントに対しては, LC の値を変える. したがって, デッドロックは com. stat., scheduler, 及び関数 next のみに依存する現象であることが分る. デッドロックは next に依存して生起するので, 実際ある種の next に対しては, ナンセンスな状態でデッドロックする. そこで本報告では, next は次の意味で公平 (fair) であるとする.

(定義6) 関数 next が公平 (fair) であるとは. 任意の計算状況 (NP, LC, Q, Y) $(stat\ NP \wedge loc\ NP \neq halt)$ に対して, ある $n \geq 1$ が存在して $(NP, LC, Q, Y) \vdash^n (NP, LC', Q', Y')$ となることである. ∎

(補題1) next が公平であるとき，ppsがデッドロック状態にあることと，ppsの計算状況(NP,LC,Q,Y)が任意の，$loc_i \neq k$ であるiに対して exec(stat$_i$loc$_i$,LC,Q,Y)=(LC,Q,Y)であることとは同値である．

(証明) ppsの実行の定義，公平の定義より明らか． ∎

特定のいくつかのステートメントまたはステートメントの列が同時には実行されないとき，それらは互いに相互排他されているという．2節で述べたいくつかの定義より明らかなように，ppsでは各ステートメント単位での相互排他は実現されている．本報告では少し大きな範囲での相互排他を考える．すなわち，2つの com.stat. にはさまれその間には com.stat. を含まない一連のステートメントの列をブロックと呼び，そのようなブロック毎の相互排他を考える．ブロックは1つのプロセスに含まれるステートメントの集合であるので，B_{iI} (i:自然数; I: 自然数の有限部分集合) で process$_i$ の I の中の数をロケーションとして持つステートメントよりなるブロックを表現することにする．

(定義7) $B_{i_1I_1}, \cdots, B_{i_nI_n}$ の中の任意の2つのブロック B_{iI}, B_{jJ} について，任意に $k \in I$, $l \in J$ をとるとき，ppsの実行の任意の計算状況(NP,LC,Q,Y)において，$loc_i = k \wedge loc_j = l$ となることがないとき，$B_{i_1I_1}, \cdots, B_{i_nI_n}$ は互いに相互排他

されているという. ∎

相互排他の問題を扱う場合にもデッドロック問題のときと同様に関数 next は公平であるとする.

3.2 問題と解法

図1はプロセス間通信を含む Concurrent-PASCAL プログラムに対応して現われる pps の一例である. しかも, その $process_1$ のループ回数と $process_2$ のループ回数の比が $n: f(n)$ (f: 多項式) で表現できる場合が少なくない. 以下では, このような場合に pps がデッドロックに陥る可能性があるか否か, そして相互排他が指定したブロックの集合に対して実現されているか否かは決定可能であることを示す.

(定義8) 2-テープ有限オートマトン (2-TFA) T は5項組 $(K, \Sigma, \delta, q_0, F)$ である. ここに, K: 状態の有限集合, Σ: 入力シンボルの有限集合でその中の特殊な元 $\$$ をエンドマークとして使用する, $\delta: K \times \Sigma \times \Sigma \to 2^{K \times \{0,1\} \times \{0,1\}}$, $q_0 \in K$: 初期状態, $F \subseteq K$: 最終状態の集合, である. ∎

(定義9) 2-TFA T の計算状況は3項組 (q, r, s) である. ここに, q: 現在の状態, r: 第1テープの端からヘッドのある位置までの距離, s: 第2テープの端からヘッドのある位置までの距離, である. ∎

(定義10) 計算状況の推移$\vdash$は以下で定義される.

$\delta(q,(a_r,b_s))\ni(q',(D_1,D_2))$ (a_rはテープ1のr番目のシンボル, b_sはテープ2のs番目のシンボル, $D_1,D_2\in\{0,1\}$)のとき, $(q,r,s)\vdash(q',r+D_1,s+D_2)$ である.

$\vdash^*,\vdash^+,\vdash$ は普通の方法で$\vdash$から拡張されて定義されるものとする. ∎

(定義11) $a_1\cdots a_{r-1}, b_1\cdots b_{s-1}\in(\Sigma-\{\$\})^*$, $T_1=a_1\cdots a_{r-1}\$$, $T_2=b_1\cdots b_{s-1}\$$ とするとき, 計算状況の列 $(q_0,1,1)\vdash^*(q_F,l,m)$ ($q_F\in F$, $l\le r$, $m\le s$) が存在すれば, $\mathcal{T}$は語(T_1,T_2)を受理するという. ($\forall q_F\in F$に対するδは常にϕであり, δの値は$(q_F,(0,0))$であるとする.) ∎

(補題2) $W_D=\{((a_1\cdots a_\alpha)^n\$,(b_1\cdots b_\beta)^{f(n)}\$)\}$ ($n\ge1$)とする. このとき, 図1のppsがデッドロックする可能性を持たないことと $\mathcal{L}(\mathcal{T}_D)\cap W_D=\phi$ とが同値となる2-TFA $\mathcal{T}_D$が存在する.

(略証) 図1のscheduler $\mathcal{S}=(K,\Sigma,\delta,q_0)$ に対して, 2-TFA $\mathcal{T}_D=(K',\Sigma',\delta',q_0',F')$ を以下のように構成する.

・$K'=K\cup\{q_F\}$, $\Sigma'=\Sigma\cup\{\$\}$, $q_0'=q_0$, $F'=\{q_F\}$,

$\delta(q,a)=q'$ ($a\in\Sigma$)のとき, $\forall b\in\Sigma'$に対して $\delta'(q,(a,b))\ni(q',(1,0))$ 且つ $\delta'(q,(b,a))\ni(q',(0,1))$, $\delta(q,a)=\phi$ ($a\in\Sigma$)のとき, $\delta'(q,(a,\$))=\delta'(q,(\$,a))\ni(q_F,(0,0))$, $\delta(q,a)=\delta$

$(q,b)=\phi$ $(a,b\in\Sigma)$ のとき，$\delta'(q,(a,b))=\delta'(q,(b,a))\ni(q_F,(0,0))$

このとき，$\mathcal{T}_\Phi$ はppsの制御をシミュレートして，ppsがデッドロックに入るとき，そのときに限り受理に至る．■

$M_1=\{B_1^1,\cdots,B_{i_1}^1\},\cdots,M_\ell=\{B_1^\ell,\cdots,B_{i_\ell}^\ell\}$ とそれぞれ互いに相互排他が期待されるブロックの集合とする．ここに任意の j,k に対して，$M_j\cap M_k=\phi$ である．ストリング $S_1=a_1\cdots a_\alpha$ に対して，その中の a_j と a_{j+1} の間のブロックが M_i に含まれるとき，シンボル M_i を a_j と a_{j+1} の間にはさんだストリングを S_1' とする．$S_2=b_1\cdots b_\beta$ に対して同様の操作を行い S_2' とする．$W_M=\{((S_1')^n\$,(S_2')^{f(n)}\$)\}$ $(n\geq 1)$ とする．

(補題3) 図1のppsがデッドロックする可能性がなく，且つ $M_1,\cdots,M_\ell$ の各集合に含まれるブロック間での相互排他が実現されていることと $\mathcal{L}(\mathcal{T}_M)\cap W_M=\phi$ とが同値となるような2-TFA $\mathcal{T}_M$ が存在する．

(略証) 図1のscheduler $\mathcal{S}=(K,\Sigma,\delta,q_0)$ に対して，2-TFA $\mathcal{T}_M=(K'',\Sigma'',\delta'',q_0'',F'')$ を以下のように構成する．

・$K''=K\cup\{q_F\}$，$\Sigma''=\Sigma\cup\{\$\}\cup\{M_1,\cdots,M_\ell\}$，$q_0''=q_0$，$F''\ni q_F$

$\delta(q,a)=q'$ $(a\in\Sigma)$ のとき，$\forall b\in\Sigma''$ に対して $\delta''(q,(a,b))\ni(q',(1,0))$ 且つ $\delta''(q,(b,a))\ni(q',(0,1))$，$\delta(q,a)=\phi$ $(a\in\Sigma)$ のとき，$\delta''(q,(a,\$))=\delta''(q,(\$,a))\ni(q_F,(0,0))$，$\delta(q,a)=\delta(q,b)=\phi$ $(a,b\in\Sigma)$ のとき，$\delta''(q,(a,b))=\delta''(q,(b,a))\ni(q_F,(0,0))$

$\forall i, \forall q$ に対して、$b \in \Sigma'' - \{M_i\}$ とするとき、$\delta''(q,(M_i,b)) \ni (q,(1,0))$ 且つ $\delta''(q,(b,M_i)) \ni (q,(0,1))$ 且つ、$\delta''(q,(M_i,M_i)) \ni (q_F,(0,0))$. ∎

補題2.3により、ここでの問題は2-TFAオートマトンと語の集合 $W=\{(a_1 \cdots a_\alpha)^n \$, (b_1 \cdots b_\beta)^{f(n)} \$ \mid (n \geq 1)$ に対して、$L(\mathcal{T}) \cap W = \phi$ か否かの決定問題に帰着された.

2-TFAオートマトン $\mathcal{T}$ と W に対して、2-TFAオートマトン $\mathcal{T}'=(K', \Sigma', \delta', q_0', F')$ ($\Sigma=\{a, b, \$\}$) を以下のように構成する.

- $K-F \ni q_i$ のとき、$K' \ni q_{ijk}$ $(1 \leq j \leq \alpha, 1 \leq k \leq \beta)$
- $\Sigma' = \{a, b, \$\}$, $q_0' = q_{011}$, $F' = \{q_F'\}$
- $\delta(q_i,(a_j,b_k)) \ni (q_\ell,(D_1,D_2))$ $(D_1, D_2 \in \{0,1\}, q_\ell \notin F)$ のとき、$\delta'(q_{ijk},(a,b)) \ni (q_{\ell j'k'},(D_1,D_2))$ (j' = if $(D_1=1)$ then $(j+1 \bmod \alpha)+1$ else j, k' = if $(D_2=1)$ then $(k+1 \bmod \beta)+1$ else k)

  $\delta(q_i,(a_j,b_k)) \ni (q_F,(D_1,D_2))$ $(q_F \in F, D_1, D_2 \in \{0,1\})$ のとき、$\delta'(q_{ijk},(a,b)) \ni (q_F',(D_1,D_2))$ (a_j, b_k の位置に $\$$ があれば、δ' の対応する位置に $\$$ が入る)
- $W' = \{(a^{\alpha n}\$, b^{\beta f(n)}\$)\}$ $(n \geq 1)$ とする.

(補題4) $L(\mathcal{T}) \cap W = \phi$ と $L(\mathcal{T}') \cap W' = \phi$ とは同値である.

(証明) 構成法より明らか. ∎

補題4により $\Sigma=\{a, b, \$\}$, $W=\{(a^{\alpha n}\$, b^{\beta f(n)}\$)\}$ $(n \geq 1)$ である2-TFAオートマトン $\mathcal{T}$ と W についてのみ考察すればよい. 以下では、2-TFA

及びWはこのようなものとする.

(定義12) 2-TFAℑのある計算状況(q, r, s)において,(q, r, s)⊢$(q, r+u, s+v)$且つこの列の中には同じ状態を持つものが両端のqを除いては現われないとき,この列をループといい,このループを$lp(Q, u, v)$(Q:ループの中に現われる状態の集合)で表現する. ∎

(定義13) 2-TFAℑの計算状況の列$(q_{p_1}, r_{p_1}, s_{p_1})$⊢$(q, r, s)$⊢$(q_{p_2}, r_{p_2}, s_{p_2})$に対して,ループ$(q, r, s)$⊢$(q, r+u, s+v)$を挿入するとは,計算状況の列$(q_{p_1}, r_{p_1}, s_{p_1})$⊢$(q, r, s)$⊢$(q, r+u, s+v)$⊢$(q_{p_2}, r_{p_2}+u, s_{p_2}+v)$を作ることをいう.逆の操作をループの除去という. ∎

(補題5) 任意の2-TFA$ℑ=(K, Σ, δ, q_0, F)$とWに対して,$L(ℑ)∩W≠φ$ならば,以下の条件をみたす$i, j≤|K|$が存在する. (1) $L(ℑ)∋(a^i c, b^j d)$

(2) $c=a^x$は$, d=b^y$は$ で $c=d=$ でない.

(証明) $L(ℑ)∩W≠φ$だから そのようなWの元を$(a^{αN}$$, b^{βf(N)}$$)$とする.受理に至る計算状況の列を,$(q_0, 1, 1)$⊢$(q_F, r, s)$ $(q_F∈F)$とする.

(i) $t≤|K|$のとき,$w=(a^{r-1} c, b^{s-1} d)$ ($αN=r-1$のとき,$c=$,$βf(N)=s-1$のとき$d=$とする)とすれば.wは条件をみたす.

(ii) $t \geq |K|+1$ のとき（図2参照）

この計算状況の列の中にループが存在する．すなわち，

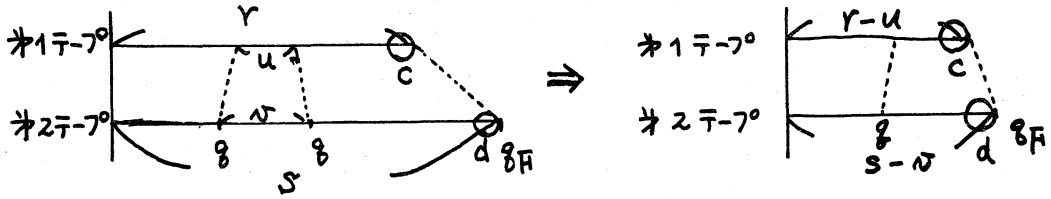
$(q_0,1,1) \vdash^i (q,j,k) \vdash^p (q,j+u,k+v) \vdash^m (q_F,r,s)$

列よりループを除去した計算状況の列

$(q_0,1,1) \vdash^i (q,j,k) \vdash^m (q_F,r-u,s-v)$ を作る．

$i+m \geq |K|+1$ であれば，この中に再びループを見出すことが出来る．この操作を繰り返した結果得た計算状況の列を $(q_0,1,1) \vdash^n (q_F,p,q)$ とする．$(n \leq |K|)$ このとき，

$w = (a^{p-1}c, b^{q-1}d)$ ($\alpha(N)=r-1$ のとき，$c=\$$，$\beta f(N)=s-1$ のとき，$d=\$$ とする）とすれば，w はこの計算状況の推移に従って受理され，$p-1, q-1 \leq |K|$ である． ∎



（図2）最終状態に至る計算状況の列からのループの除去

補題5によって語 (a^ic, b^jd) $(i,j \leq |K|)$ で $L(T)$ の元であるものが存在することが，$L(T) \cap W \neq \Phi$ の必要条件であることが分った．したがって，"語 (a^ic, b^jd) の受理に至る計算状況の列に，許される適当なループを挿入したとき，あるn

に対して $(a^{\alpha n}\$, b^{\beta f(n)}\$)$ の受理に至る計算状況の列とすることが出来るか否かを』を問題Aと書くとき，問題Aが可解であれば，$\mathcal{L}(T) \cap W = \phi$ か否かは可解となる．

(注) 問題Aで使用されるループの種類は，定義12より，2-THA Tより有限個に定まる．これらのループを $lp_i(Q_i, \alpha_i, \beta_i)$ で示し，その集合をLPで示す．

(補題6) 問題Aは可解である．

証明の中で使用する補題を示す．

(補題7) $\forall \alpha_i\ (1 \le i \le l)$：正の整数，$\forall p$：正の値をとる多項式が与えられたとき，

$$\sum_{i=1}^{l} \alpha_i \eta_i = p(n)$$

をみたす整数 n，$\eta_i\ (1 \le i \le l)$ が存在するか否かは，決定可能である．

(証明) $G = G.C.D(\alpha_1, \cdots, \alpha_l)$ とすると $\Sigma \alpha_i \eta_i = Gt$ と書ける．よって，$p(n) \bmod G = 0$ となる n が存在するか否かを決定すればよい．$p(n) \bmod G = p(n \bmod G) \bmod G$ より．この問題は決定可能である． ∎

(補題8) $\forall \alpha_i\ (1 \le i \le l)$：正の整数が与えられているとき，

$\sum_{i=1}^{l} \alpha_i \eta_i = n$ において，n を大きくするとき，

$\forall i$ に対して定数 c_{ij} が存在して $0 < \eta_j < c_{ij}\ (i \neq j)$ となる整数解 $\eta_i\ (1 \le i \le l)$ が存在する．$(G.C.D(\alpha_1, \cdots, \alpha_l) = 1)$

(証明) 先ず, $\forall \alpha, \beta$ ($G.C.D(\alpha,\beta)=1$) が与えられた時 n がいくら大きくとっても, $\alpha x+\beta y=n$ は $0<y<c$ (c:定数) となる解 (x,y) を持つことを示す.

$\alpha x+\beta y=1$ の1つの解を (x_0, y_0) とする. $m=\min(\alpha,\beta)$ とする. このとき n を大きくとると,

$$n=L\alpha+(m|y_0|+1)(\alpha+\beta)+Mm+i \quad (1\le i\le m,\ 0\le Mm<\alpha)$$

と書ける. $\alpha(ix_0)+\beta(iy_0)=i$ であるので.

$$n=(L+m|y_0|+1+ix_0)\alpha+(m|y_0|+1+iy_0)\beta+Mm$$

よって, $0<y<m|y_0|+my_0+M+1$ となる解が存在する.

さて, $\forall i$ に対して,

$$\alpha_1\eta_1+\cdots+\alpha_{i-1}\eta_{i-1}+\alpha_{i+1}\eta_{i+1}+\cdots+\alpha_k\eta_k+\alpha_i\eta_i$$
$$=\alpha_1\eta_1+G.C.D(\alpha_2,\alpha_3,\cdots,\alpha_{i-1},\alpha_{i+1},\cdots,\alpha_k,\alpha_i)\eta$$

と考えると, $0<\eta_1<C_{i1}$ なる解が存在する. ここで

$$\alpha_j / G.C.D(\alpha_2,\alpha_3,\cdots,\alpha_i)=\alpha_j' \text{ とすると}$$

$$\alpha_2'\eta_2+\cdots+\alpha_{i-1}'\eta_{i-1}+\alpha_{i+1}'\eta_{i+1}+\cdots+\alpha_k'\eta_k+\alpha_i'\eta_i=\eta$$

同様のことを繰返すと $\forall j\neq i$ に対して $0<\eta_j<C_{ij}$ となる解がある. ∎

(補題 b の略証)

(i) $c=a$, $d=b$ の場合

問題Aが可解となることは明らかである.

(ii) c_2 は d のいずれかが $ である場合

$(S_1, S_2) = (a^i c, b^j d)$ の受理に至る計算状況の列を

$(q_0, 1, 1) \vdash (q_f, i+1, j+1)$ と書く.

このとき，計算状況の列に挿入可能なループの集合を求めるアルゴリズムを与える.

(Algorithm IR)

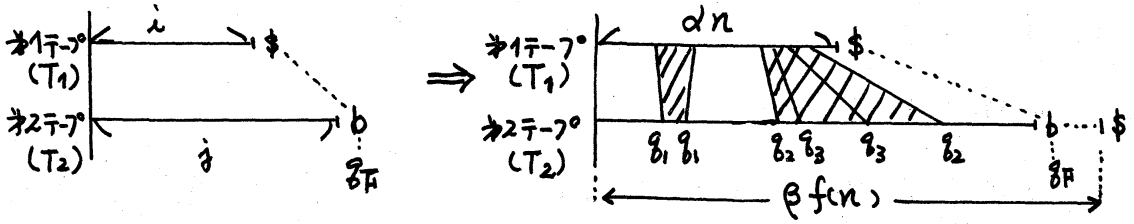
(step 1) 計算状況の列中に現われた q_f 以外の状態の集合を Q, $L = \phi$ とする.

(step 2) LP 中の $\forall_i$ に対して $Q_i \cap Q \neq \phi$ ならば $Q = Q \cup Q_i$, $L = L \cup \{lp_i\}$

(step 3) step 2 を L が増さなくなるまで繰返す. ∎

(a) $\deg(f(n)) > 1$ の場合

(1) $c = $ $ のとき. (図3参照)



(図3) ループの挿入

図3のように T_1, T_2 にいくつかのループを挿入して，T_1 の長さがある n に対して，$\alpha n+1$，T_2 の長さは $\beta f(n)$ 以下となるようにすることが出来るか否かを決定すればよい. Algorithm IR の出力 L を $\{lp_1, \cdots lp_p\}$ とするとき，問題は，

$$\begin{cases} i+\sum_{k=1}^{p}\alpha_k\eta_k=\alpha n & (1) \\ j+\sum_{k=1}^{p}\beta_k\eta_k<\beta f(n) & (2) \end{cases}$$

をみたす非負整数解が存在するか否かを決定すればよい．ここで，$\eta_k=0$ のとき，間接的に $\eta_k=0$ でなければならないものが存在するが，補題7.8より(1)の解 n は非負に無数に存在し，したがって，自然数解 η_k を持つ．逆に，いくつかの η_k を0に限定しても，無限定で(1)に解がなければ，解を有さないので，(1),(2)の自然数解を考察すればよい．

補題7.8より(1)をみたす，自然数解 η_k の存在は決定可能であった．　$\max_k[\beta_k/\alpha_k]=M$ とおく．

$\sum_k\eta_k\beta_k<M\sum_k\eta_k\alpha_k$ である．　よって，

$j+\sum_k\beta_k\eta_k<Mi+M\sum_k\alpha_k\eta_k+j-Mi=M\alpha n+(j-Mi)$．

$\deg(f(n))>1$ より　$M\alpha n+(j-Mi)<\beta f(n)$　となる n は(2)をみたす n の中にある．

(ロ) $d=\$$ のとき

(1)と同様にして，

$$\begin{cases} i+\sum_{k=1}^{p}\alpha_k\eta_k<\alpha n & (3) \\ j+\sum_{k=1}^{p}\beta_k\eta_k=\beta f(n) & (4) \end{cases}$$

をみたす自然数解が存在するか否かを決定すればよい．(4)をみたす自然数 n,η_k が存在するか否かは，補題7.8より決定可能である．

(*) $\min_k[\alpha_k/\beta_k]=m>0$ のとき

$\Sigma\alpha_k\eta_k\geq m\Sigma\beta_k\eta_k$　より，

$\alpha n > i + \Sigma \alpha_k \eta_k \geq m \Sigma \beta_k \eta_k + mj + i - mj = m\beta f(n) + (i - mj)$

したがって，$\deg(f(n)) > 1$ より n に上限 N が存在する．したがって，可解となる．

(**) $\min_k [\alpha_k / \beta_k] = 0$ すなわち $\exists_t$ s.t $\alpha_t = 0$ のとき

(4)の解において n を大きくとっても，補題8より，$\alpha_t = 0$ となる t に対して $t \neq k$ で $0 < \eta_k < C_{tk}$ となる定数 C_{tk} が存在する．そのように解 η_k をとると，

$$i + \Sigma \alpha_k \eta_k = i + \sum_{k \neq t} \alpha_k \eta_k < C \quad (C: \text{定数})$$

となるので解は存在する．

以上より，この問題は可解となる．

$\deg(f(n)) \leq 1$ の場合も同様にして可解であることが証明できる． ∎

(定理) 任意の 2-TFA $\mathcal{T} = (K, \Sigma, \delta, q_0, F)$ ($\Sigma = \{a, b, \$\}$) と $W = \{(a^{\alpha n}\$, b^{\beta f(n)}\$) \mid (n \geq 1, f: \text{多項式})\}$ に対して，

$$\mathcal{L}(\mathcal{T}) \cap W = \phi$$ が否かは可解である．

(証明) 補題5.6による． ∎

(系) 図1の pps に関する上述のデッドロック問題及び相互排他問題は可解である． ∎

4. 結び

本報告で示した結果は次のようにいくらか拡張できる．例

へば，定理は任意の σ，w に対して有効であることから，このような σ，w に変換可能であるような λ-TFA と語の集合に対しては同じ議論が成立する．したがって，$w = \{(a_1 \cdots a_k)(a_{k+1} \cdots a_j)^n \$, (b_1 \cdots b_k)(b_{k+1} \cdots b_m)^{f(n)} \$)\}$ となる PPS に対しても，ここでとりエげたデッドロック と相互排他問題は可解となる．プロセスが3以上，又プロセスの形が複雑な場合については検討中である．

謝辞　日頃，御指導をいただく本学福村晃夫教授及び熱心に御討論くださった研究室の皆様に感謝します．

文献

(1) HANSEN, P.B., Concurrent-PASCAL Report, Information Science, California Institute of Technology, NSF QCA-DCR 74 1733/ cPI (1975)

(2) KARP, R. & MILLER, R., Parallel Program Schemata, JCSS (1969)

(3) ASHCROFT, E. & MANNA, Z., Formalization of Properties of Parallel Programs, Machine Intelligence (1971)

(4) 伊藤貴康，プログラム理論，コロナ社 (1975)