

同期付生成システム (SPS) について

名古屋大	工	山下雅史
三重大	工	稲垣康善
名古屋大	工	本多波雄

1. まえがき

一般に，並行処理システム（あるいは，その同期機構）は，並行に実行される各プロセス（事象の系列）の構成やその出現法則を定める機構と，各事象がその実行時に満足すべき条件を定める機構の対として表現できる．本稿では，上に述べたオの機構を文脈自由文法によって，オの機構をオートマトンによって記述する新しい並行処理システムモデルとして，同期付生成システム（Synchronized Production System-SPS）を提案する．さらに，ペトリネット言語と同じ意味で SPS 言語を定義し，これをを用いて SPS の各部令クラス間の，又，SPS と他のモデルとの間の能力比較を行う．尚，本報告で議論の対象とするのは主に正規文法と有限オートマトンによって定義される SPS である．

2. 同期付生成システム (SPS)

(1)

本章では, SPS の定義を与える. まず, 以下で使用するいくつかの記法を約束しておく.

[記法] i) 記号の集合 Σ 上の正規集合の族を $R(\Sigma)$ で示す. ii) 正規集合 α の系列 x による微分を $\partial_x \alpha$ とする. 即ち, $\partial_x \alpha = \{u \mid xu \in \alpha\}$ である. iii) 非負整数の n 項組 $\sigma = (\sigma_1, \dots, \sigma_n)$ の要素 σ_i を $(\sigma)_i$ で表わす. 又, $\sigma \pm \tau$ ($\tau \in \{0, 1\}$) によって, σ_i に $(\tau)_i$ を ± 1 を代入することによって σ から作られる n 項組 (複号同順) を示す. iv) 集合 A, B に対して, $A - B = A \cap \neg B$ とする. v) λ は空系列, 0 は零ベクトルを示す. □

[定義 1] SPS は 2 項組 (ϕ, A) である. ここで, ϕ は 4 項組 $(\Sigma_N, \Sigma_T, S, P)$ で定められるシステムであり, Σ_N は非終端記号の集合, Σ_T は終端記号の集合 ($\Sigma = \Sigma_N \cup \Sigma_T$ とする), ϕ は Σ_N の特別な要素で開始記号と呼ばれる. 又, P は生成規則の集合であり, 各生成規則は, $A \rightarrow \alpha$ ($A \in \Sigma_N, \alpha \in R(\Sigma), \lambda \neq \alpha$) の形を持つ. A は図 1 に示すようなオートマトンであり, その計算状況は, オートマトンの内部状態と補助テープの状況の 2 項組で表現される. A の入力記号集合は

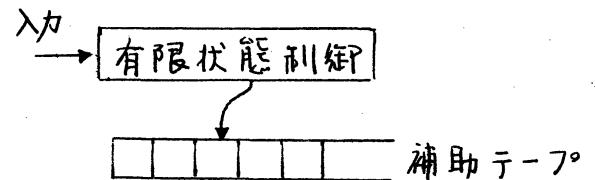


図 1. オートマトン

である。Aが計算状況cにあって、 λ を入力され、計算状況cに推移したとき、この推移を $c \xrightarrow{\lambda} c'$ と表わす。文脈から明らかになると、 λ を省略することもある。又、 τ の推移的閉包を τ^* と書く。□

〔定義2〕SPSの時点表示(id)は2項組 (U, C) である。

ここに、 $\Omega = \{ \sigma \mid \exists (A \rightarrow \alpha) \in P, \exists x \in \mathbb{Z}^*, \sigma = \partial_\alpha \alpha - \{ \lambda \} \cup \{ \lambda \} \}$ とすると、 U は非負整数の $|\Omega|$ 項組、すなわち、 $U = \bigotimes_{\sigma \in \Omega} U_\sigma$ である。又、 C はオートマトンAの計算状況である。□

〔定義3〕時点tのid $I_t = (U_t, C_t)$ から時点 $t+1$ のid $I_{t+1} = (U_{t+1}, C_{t+1})$ への推移は以下のように定義される。

$I_t = (U_t, C_t)$ に対して、 $N_t = \{ (\lambda, \sigma, c) \mid \lambda \in \mathbb{Z}, \sigma \in \Omega, \partial_\alpha \sigma \neq \phi, (U_t)_\sigma \geq 1, c \xrightarrow{\lambda} c' \}$ とする。

このとき、 $I_t = (U_t, C_t)$ から $I_{t+1} = (U_{t+1}, C_{t+1})$ への推移が存在する必要十分条件は、 N_t の要素 (λ, σ, c) が存在して、 $C_{t+1} = c$ かつ、 U_{t+1} が以下に定められるようなものであることである。 N_t の要素 (λ, σ, c) に対して、 $\Delta_1 = U_t - \sigma$ 、 $\Delta_2 = \Delta_1 + (\partial_\alpha \sigma - \{ \lambda \})$ とする。

$$(1) \lambda \in \mathbb{Z}_T \text{ のとき, } U_{t+1} = \begin{cases} \Delta_1: \partial_\alpha \sigma = \{ \lambda \} \text{ のとき} \\ \Delta_2: \partial_\alpha \sigma \neq \{ \lambda \} \text{ のとき} \\ \Delta_1 \text{ 又は } \Delta_2: \text{その他以下} \text{ のとき, である.} \end{cases}$$

(2) $\alpha \in \Sigma_N$ のとき, $(\alpha \rightarrow \alpha) \in P$ に対して, $\Delta'_1 = \Delta_1 + \alpha$, $\Delta'_2 = \Delta_2 + \alpha$ とする.

$$U_{t+1} = \begin{cases} \Delta'_1 : \exists \sigma \in \Sigma^* \text{ s.t. } \alpha \in \sigma \\ \Delta'_2 : \exists \sigma \in \Sigma^* \text{ s.t. } \alpha \in \sigma \\ \Delta'_1 \text{ 又は } \Delta'_2 : \text{それ以外の場合, である.} \end{cases}$$

$(\alpha, \sigma, c) \in N_t$ によって I_t から I_{t+1} への推移 P_t できるとき, これを $I_t \xrightarrow{P_t} I_{t+1}$ と表わす. α を推移 $I_t \xrightarrow{P_t} I_{t+1}$ で読み入れた記号という. α を明示する必要 P_t なければ省略する. $I_1 \xrightarrow{P_1} I_2 \xrightarrow{P_2} \dots \xrightarrow{P_{n-1}} I_n$ であるとき, $I_1 \xrightarrow{P} I_n$ ($\sigma = \alpha_1 \dots \alpha_{n-1}$) と書く. 又, $\exists \sigma \in \Sigma^*$ かつ, $I_1 \xrightarrow{P} I_n$ であるとき, $I_1 \xrightarrow{*} I_n$ と書く. $\square$

[定義4] SPS $\mathcal{S} = (P, A)$ の動作とは, id の推移列, $I_0 \rightarrow I_1 \rightarrow \dots$ である. ここに, $I_0 = (U_0, c_0)$ ($U_0 = 0 + \{S\}$, c_0 は A の初期計算状況) である. $\square$

[定義5] $N_t = \emptyset$ であるような id I_t に到達したとき, SPS は停止する. 持ち $\mathcal{S}$ が $I_h = (U_h, c_h)$ ($U_h = 0$) に到達したとき, $\mathcal{S}$ は完了したという. さらに, c_h が A の受理計算状況であるとき, $\mathcal{S}$ は受理状態で完了したという. $\square$

SPS $\mathcal{S} = (P, A)$ の動作は, 以下のように定義されるマーク付導出木の動作によって, 模型的に表現できる.

[定義6] $\mathcal{P} = (\Sigma_N, \Sigma_T, S, P)$ において, $(A \rightarrow \alpha) \in P$ は, $\forall u (\in \alpha) [(A \rightarrow u) \in P]$ の略記である. 従って, $\mathcal{P}$ は無限個の生成規則を持ち得る文脈自由文法である. $\square$

〔定義7〕 P の導出途中木および導出木の集合を $D(P)$ と書く。導出途中木も単に導出木と呼ぶことにする。 $d \in D(P)$ の項(item)とは、根節点 S , および、同じ親を持つ子節点の文字列(生成規則の右辺に現れる)である。導出木 d の各項に高々1個の"•"(mark)を挿入してできる木をマーク付導出木(Marked Derivation Tree - mdt)と呼び、 P のmdtの集合を $M(P)$ と書く。(図2参照) 四

このとき、SPSの時点表示、および動作は以下のように定義することもできる。

〔定義8〕SPS $\mathcal{S} = (P, A)$ の時点表示(id)は2項組 $I = (m, c)$ である。ここに、 $m \in M(P)$, c は A の計算状況である。 四

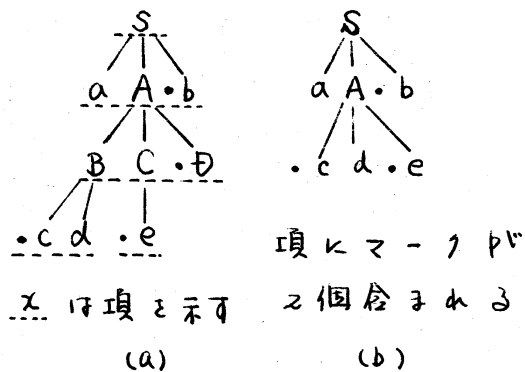


図2 (a) mdt (b) mdtでマーク

〔定義9〕時点 t におけるidは $I_t = (m_t, c_t)$ であるとき、時点 $t+1$ のid $I_{t+1} = (m_{t+1}, c_{t+1})$ は以下のように定義される。 m_t に含みのあるマーク直後の記号 α , I_t の A の計算状況 c_t の α と α に受付けることのできる記号の集合を N_t とする。 $\rho \in N_t$ に対して、 A は ρ を入力して計算状況を c_{t+1} に移すとする。 ρ を含む項を $\rho_1 \cdots \rho_{i-1} \cdot \rho \rho_{i+1} \cdots \rho_k$ とする。

(1) $\lambda \in \Sigma_T$ のとき, 項を $\lambda_1 \dots \lambda_{i-1} \lambda \cdot \lambda_{i+1} \dots \lambda_k$ に置換え m dt を m_{t+1} とする.

(2) $\lambda \in \Sigma_N$ のとき, 項を $\lambda_1 \dots \lambda_{i-1} \lambda \cdot \lambda_{i+1} \dots \lambda_k$ に置換え m dt mdt m_{t+1} とする ($k \geq 1$, $((\lambda \rightarrow n_1 \dots n_\ell) \in P)$ とする.)

(3) 操作 (1), (2) の結果, マークP項の右端にくれば, そのマークは消去される.

SPSは N_t の中の一つの元を非決定的に選択して, 次のidに推移する. このとき, 選択された N_t の要素 λ を推移によって読まれた記号と見做し, この推移を $I_t \Vdash I_{t+1}$ と表わす. 文脈 P から明らかでない λ を省略する. $I_1 \Vdash I_2 \Vdash \dots \Vdash I_n$ であるとき, $I_1 \Vdash I_n$ ($\sigma = \lambda_1 \dots \lambda_{n-1}$) と書く. また, $\exists \sigma \in \Sigma^*$ かつ, $I_1 \Vdash I_n$ のとき $I_1 \Vdash^* I_n$ と書く. $\square$

[定義10] SPS $\mathcal{S} = (P, A)$ の動作は, idの推移列 $I_0 \Vdash I_1 \Vdash \dots$ である. ここで, $I_0 = (\cdot S, c_0)$ (c_0 は A の初期計算状況) である.

SPS $\mathcal{S}$ は, $N_t = \phi$ となる id $I_t = (m_t, c_t)$ に到達したとき停止する. 特に, $m_t \in \emptyset(P)$ のとき (マークP項全てなくなると) 完了したという. さらに, c_t が A の受理計算状況であれば, $\mathcal{S}$ は受理状態で完了したという. $\square$

ここで, 定義2~5で定めるSPSの動作と定義8~10で定める

は SPS の動作が本質的に同等であることを示す.

《補題 1》任意の SPS $\mathcal{S}$ に対して, $[I_0 = (c_0, c_0)] \vdash^{\mathcal{S}} I_* = (c_*, c_*) \Leftrightarrow [I_0 = (c_0, c_0)] \Vdash^{\mathcal{S}} I_* = (m_*, c_*)$ ($c_0 = 0 + \{S\}$) かつ, c_* と m_* は以下の関係をみたす.

m_* のマークの付いた項を $x \cdot y$ とする. $x \cdot y$ の導出に用いられる生成規則 $\vdash A \rightarrow \alpha$ (即ち $x \cdot y \in \alpha$) とする. このとき, $\exists x \alpha - \{x\} = \sigma$ となるようなマーク付きの項 $x \cdot y$ の個数は c_0 に一致する.

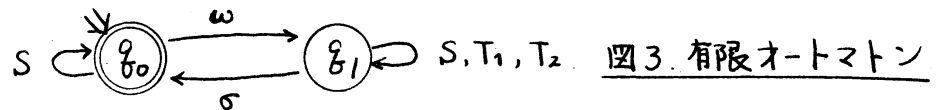
(略証) 推移列の長さに関する帰納法により証明できる. $\square$

補題 1 から, 二つの定義のいずれを SPS の動作の定義としてもよいことが分かる. 以降では, $\vdash$ と $\Vdash$ を区別せず $\vdash$ と書く. SPS は以下のよう解釈することによって並行処理システムを表現する.

- (1) 各項は, 並行に処理されるプロセス
- (2) マークは, 各プロセスに割り付けられたプロセッサの実行位置
- (3) Σ_N の元は, 新しいプロセスを作りプロセッサとそのプロセスに割り付ける命令
- (4) Σ_T の元は, プロセスを構成する事象, を示す.

(例 1) 2 種類のタスク T_1, T_2 を処理するシステムを考える. 無作為に, 任意の個数の T_1, T_2 がシステムに到来するものと

する。各タスクは、到来と同時にプロセッサを割り付けられ、並行に処理されるが、各タスクの実行は互いに相互排除される必要はないものとする。このシステムをSPSに於て表わす。 $\mathcal{S} = (\mathcal{P}, \mathcal{A})$, $\mathcal{P} = (\{S\}, \{T_1, T_2, \sigma, \omega\}, S, P)$, $P = \{S \rightarrow S\omega T_1\sigma \mid S\omega T_2\sigma \mid \omega T_1\sigma \mid \omega T_2\sigma\}$, $\mathcal{A} = (\{q_0, q_1\}, \{S, T_1, T_2, \omega, \sigma\}, \delta, q_0, \{q_0\})$ 図3の有限オートマトンである。



$\mathcal{S}$ の動作の一例を図4に示す。

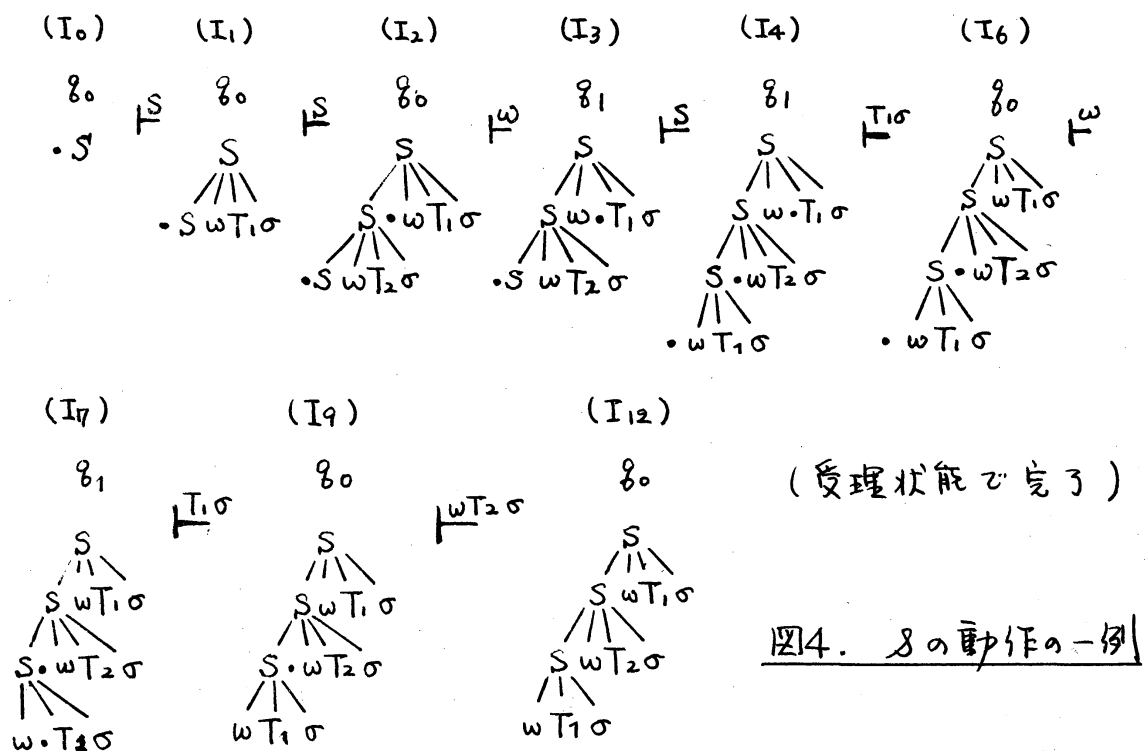


図4. $\mathcal{S}$ の動作の一例

$\mathcal{S}$ は S で二種類のタスクを受付、プロセッサを割り付ける一種のオペレーティング・システムの抽象化と考えられる。 σ と ω

はプロセス間の相互排除を実現するためのセマフォである。I4
では、システムは3つのプロセス w_{T1} , w_{T2} , および w 途中
まで実行の進んだ w_{T1} を持っている。又、I6でマークの直
後の2つの w のいずれを実行するかは選択は非決定になさる。
即ち、一度プロセッサを割付けられればプロセスは、以前
からあるプロセスと同一の取扱いを受ける。□

3. SPS 言語

種々の並行処理システムのモデルの能力比較には、例えば
ペトリネット言語⁽¹⁾ のように、正しい最終状態に至る可能
な計算履歴の集合によって定義される言語 L によって利用さ
れる。そこで、SPS に対しても、SPS $\mathcal{S}$ によって定義される
言語 $L(\mathcal{S})$ を定義する。

[定義11] 準同形写像 $h: \Sigma \rightarrow \Sigma^* \cup \{\lambda\}$ と $h(\alpha) = u$ ($u =$
 $\text{if } (\alpha \in \Sigma^*) \text{ then } \alpha \text{ else } \lambda$) で定義する。

$\sigma = \sigma_1 \dots \sigma_n$ ($\sigma_i \in \Sigma$) に対して、 h と $h(\sigma) = h(\sigma_1) \dots h(\sigma_n)$
によって拡張する。

推移列 $I = I_1 \dots I_n$, $I_1 \vdash I_n$ に対して、 $\pi(I) = h(\sigma)$ と定義する。□

[定義12] SPS において、 $\mathcal{S}$ を受理状態で見たさせる推移列
の集合を $\mathcal{D}(\mathcal{S})$ とおく。このとき、 $L(\mathcal{S}) = \{\pi(I) \mid I \in \mathcal{D}(\mathcal{S})\}$
と $\mathcal{S}$ によって定義される SPS 言語と呼ぶ。□

本稿の以下の部分では、オートマトンが有限オートマトン

で表現される SPS および SPS 言語について考察する。オートマトンが有限オートマトンである SPS のクラスを $\mathcal{M}$, $\mathcal{L} = \{L(s) \mid s \in \mathcal{M}\}$ によって $\mathcal{M}$ に対応する SPS 言語の族を示す。

3.1 $\mathcal{M}_B, \mathcal{L}_B$ の性質 本節では、有限台数のプロセッサによって実行できる（あるいは、同時に並行に実行されるプロセス数に上限がある）という意味で、実際的なシステムモデルに対応する SPS および SPS 言語の族を考察する。

[定義 13] $\mathcal{M}_B = \{s \mid \exists k, \forall I_x, I_0 * I_x = (\cup_x, g_x) \Rightarrow \sum (u_x)_0 \leq k\}$, $\mathcal{L}_B = \{L(s) \mid s \in \mathcal{M}_B\}$ とする。 $\square$

《定理 1》正規集合の族を $\mathcal{R}$ として, $\mathcal{L}_B = \mathcal{R}$

(略証) $\mathcal{R} \subseteq \mathcal{L}_B$ は明らか。 $\mathcal{L}_B \subseteq \mathcal{R}$ を示す。どの $s \in \mathcal{M}_B$ についても、その $\cup$ の種類は有限個しかない。従って、 $L(s) \in \mathcal{R}$ である。 $\square$

このように、システムに現れるプロセッサ数を有限個数に制限すれば、対応する言語は単純な族に含まれる。従って、以下では、プロセッサ数の上限がないシステムを考察の対象とする。(1) $s \in \mathcal{M}$ が $\mathcal{M}_B$ に入るかを決定することによって示しておくことはシステム解析の立場から意味のあることであろう。

《定理 2》 $\forall s \in \mathcal{M}$ に対して、 $s \in \mathcal{M}_B$ かどうかを決定する問題は可解である。

(略証) 次の二つの命題 1), 2) は同値である.

1) $\delta \notin \mathcal{M}_B$

2) $(\sigma_0, \rho_0) \models (\sigma, \rho) \models (\sigma', \rho) \ (\sigma < \sigma')$ となる δ の動作 p が存在する.

2) $\Rightarrow$ 1) は容易. 1) $\Rightarrow$ 2) を示す. 1) p が成立し, 2) p が成立しないと仮定すると, 次のような δ の無限列 p が存在する.

$I_0 \models I_1 \models I_2 \models \dots$ ($I_i = (\sigma_i, \rho) \ (1 \leq i)$), $p \models \forall i, j \geq 1$, $\sigma_i \neq \sigma_{i+j}$, $p \models \sum (\sigma_i)_{\sigma} < \sum (\sigma_{i+j})_{\sigma}$.

$\sigma_i \neq \sigma_{i+j}$ $p \models \sum (\sigma_i)_{\sigma} < \sum (\sigma_{i+j})_{\sigma}$ となる σ_i の列は常に有限列であるから, 仮定と矛盾する. 従って, 1) と 2) p が同値であることが示される.

故に, 可能な δ の推移列の全てを探索すると, $\delta \in \mathcal{M}_B$ ならば, 探索は有限ステップで終了し, 又, $\delta \notin \mathcal{M}_B$ ならば, 2) を満たす δ の推移列を見出すことができて, $\delta \notin \mathcal{M}_B$ が認識できることから, この問題は可解である. $\square$

3.2 $\mathcal{M}_T, \mathcal{L}_T$ の性質 SPS の文法 $p: A \rightarrow \alpha B \mid \beta \ (\alpha, \beta \in R(\Sigma))$ という形を持つクラスは $\mathcal{M}_B$ に含まれるので (実際, 対応する SPS 言語の族は $\mathcal{L}_B$ に等しい) 単純なクラスである. $|p|$, 文法 $p: A \rightarrow B\alpha \mid \beta \ (\alpha, \beta \in R(\Sigma))$ という形を持つ場合には, 上記のクラスとは異なり, その小さい部分クラスにおいても, p がより大きな能力を持つことが示される.

[定義14] SPSを定義する文法 $\mathcal{P}$, $A \rightarrow Bu|v$ ($u, v \in \Sigma_T^*$, $|u|, |v| \leq k$)という形のSPSのクラスを $\mathcal{M}_k$ で示す. $\mathcal{L}_k = \{L(\mathcal{S}) \mid \mathcal{S} \in \mathcal{M}_k\}$ とする. $\square$

本稿の以下の部分では $\mathcal{M}_1$ と $\mathcal{L}_1$ について考察する.

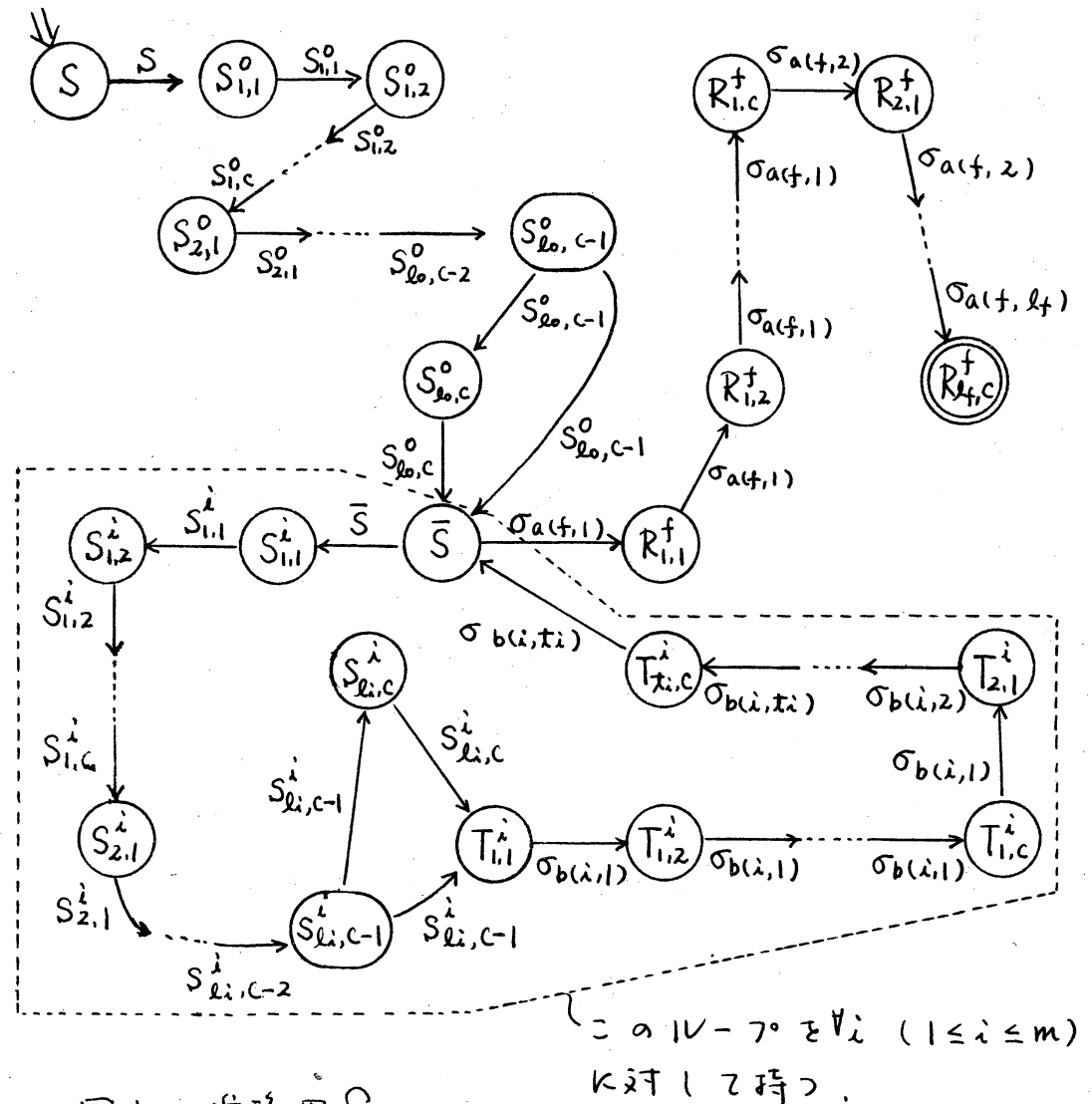
《定理3》ベクトル加算システム(VAS)の到達可能問題と $\mathcal{L}_1$ の空間問題は互いに帰着可能である.

(証明) A) VASの到達可能問題が $\mathcal{L}_1$ の空間問題に帰着できることを示す. 任意に与えられたVASを $\mathcal{V} = (v_0, V)$ とする. v_0 : 非負整数の n 項組, $V = \{v_1, \dots, v_m\}$: 整数の n 項組の有限集合, である. $\mathcal{V}$ の到達可能集合を $\mathcal{R}(\mathcal{V})$ とする. 非負整数の n 項組 v が $\mathcal{P}$ で与えられたとき, v が $\mathcal{R}(\mathcal{V})$ に属するかどうかを決定する問題と, $\mathcal{L}_1$ の空間問題に帰着する. $v_i = (c_1^i, \dots, c_n^i)$ ($0 \leq i \leq m, i$)とする. v_i の中の中要素を集めてベクトル $u \in A_i = (c_{a(i,1)}^i, \dots, c_{a(i, \ell_i)}^i)$, 負の要素を集めてベクトル $u \in B_i = (c_{b(i,1)}^i, \dots, c_{b(i, t_i)}^i)$ とする. このとき, 以下の $\mathcal{S}$ がSPS $\mathcal{S} \in \mathcal{M}_1$ を構成する. $\mathcal{S} = (\mathcal{P}, \mathcal{A})$, $\mathcal{P} = (\Sigma_N, \Sigma_T, S, P)$, $\Sigma_N = \{S, \bar{S} \cup \{S_{a(i,j),k}^i \mid 0 \leq i \leq m, 1 \leq j \leq \ell_i, 1 \leq k \leq c_{a(i,j)}^i\} \cup \{S_{a(i,j),k}^i \rightarrow S_{a(i,j),k+1}^i \sigma_{a(i,j)} \mid 1 \leq i \leq m\} \cup \{S_{a(i,j),k}^i \rightarrow S_{a(i,j),k+1}^i \sigma_{a(i,j)}^{\dagger}, S_{a(i,\ell_i),c_{a(i,\ell_i)}^i-1}^i \rightarrow \sigma_{a(i,\ell_i)} \mid$
 $\dagger) c_{a(i,\ell_i)}^i = 1$ のとき $S_{a(i,\ell_i),0}^i = S_{a(i,\ell_i-1),c_{a(i,\ell_i-1)}^i}^i$ とする.

$0 \leq i \leq m, 1 \leq j \leq l_i, 1 \leq k \leq C_{a(i,j)}^i - 1 \cup \{S_{a(i,j)}^i, C_{a(i,j)}^i\}$
 $\rightarrow S_{a(i,j+1)}^i, \sigma_{a(i,j+1)} \mid 0 \leq i \leq m, 1 \leq j \leq l_i - 1 \cup \{S_{a(i,l_i)}^i, C_{a(i,l_i)}^i\} \rightarrow \bar{S} \mid 0 \leq i \leq m \cup \{S_{a(i,l_i)}^i, C_{a(i,l_i)}^i\}$
 $\rightarrow \bar{S} \mid 0 \leq i \leq m \cup \{S_{a(i,l_i)}^i, C_{a(i,l_i)}^i\}$, である. $\mathcal{A} = (K, \bar{Z}, \delta, q_s, F)$ は, $K = \{q_A \mid A \in \Sigma_N\} \cup \{q_B \mid B = T_{b(i,j)}^i, 1 \leq i \leq m, 1 \leq j \leq t_i, 1 \leq k \leq C_{b(i,j)}^i\} \cup \{q_C \mid C = R_{a(t_f)}^t, k, 1 \leq j \leq l_f, 1 \leq k \leq C_{b(i,j)}^i\}$, $\bar{Z} = \Sigma_N \cup \Sigma_T, F = \{q_{R_{a(t_f)}^t}, C_{a(t_f)}^t\}$, δ は図 5 の推移関数を与えられる. したがって, ①は q_A である. $\sigma \in K, A = X_{y(i,j),k}^i$ ($X \in \{S, T, R\}, y \in \{a, b\}$) $\in X_{j,k}^i$ と, 特 $K, k = C_{y(i,z_i)}^i$ ($y \in \{a, b\}, z_i \in \{l, t\}$) のとき, $X_{j,c}^i$ と略記する.

δ は次の諸性質を持つ.

- i) 初めて $\text{id } p^*(\sigma, q_s)$ に到達したとき, σ は $\forall \sigma_j$ ($1 \leq j \leq n$) に対して $(\sigma)\sigma_j = (\sigma_0)_j$ である.
- ii) どの $\text{id } I = (\sigma, q_s)$ からも I から i ($1 \leq i \leq m$) の U - Γ - Γ - Γ 一周したとき $\text{id } I' = (\sigma', q_s)$ と対応して, $\Delta = \sigma' - \sigma$ であるとき, $\forall \sigma_j$ ($1 \leq j \leq n$) に対して, $(\Delta)\sigma_j = (\sigma_i)_j$ である.
- iii) どの $\text{id } I = (\sigma, q_s)$ に対しても I から U - Γ - Γ - Γ 通らないとき $\text{id } I' = (\sigma', q_{R_{a(t_f)}^t})$ に到達したとき, $\Delta = \sigma - \sigma'$ であるとき, $\forall \sigma_j$ ($1 \leq j \leq n$) に対して, $(\Delta)\sigma_j = (\sigma_f)_j$ である.

図5. 推移図 δ

以上の性質から, $R(\mathcal{V}) \ni v_f \Leftrightarrow L(\mathcal{S}) \neq \emptyset$ である.

B) $\mathcal{L}_1$ の空問題と VAS の到達可能問題に帰着する. $\text{id } I = (\sigma, \tau)$ のベクトル表現とは, $v(I) = \sigma \otimes V$ ($V = \bigotimes_{r \in K} V_r$, $V_g = 1$, $\forall g' \neq g, V_{g'} = 0$) である. このとき, VAS $\mathcal{V} = (\sigma_0, V)$ を以下のよう構成する. $\sigma_0 = v(I_0)$ ($I_0 = (\sigma_0, \tau_0)$, $\sigma_0 = 0 + 1S$, τ_0 はオートマトンの初期状態), $V =$

$\{v \mid v = v(I') - v(I), I = (U, g), \sum_U (U) \sigma = 1, I \vdash I'\}$
 とすれば, SPS の動作の定義より,

$$L(g) \neq \emptyset \Leftrightarrow R(v) \neq \emptyset \quad \blacksquare$$

定理 3 から, $\mathcal{M}_1$ の同期能力は VAS と同等であると考えられるが, SPS 言語の族 $\mathcal{L}_1$ は空系列入のラベル付拡張ペトリネット言語の族 $\mathcal{L}_{PN}$ に真に含まれることが証明できる.

《定理 4》 $\mathcal{L}_{PN} \supseteq \mathcal{L}_1$

A) $\mathcal{L}_{PN} \supseteq \mathcal{L}_1$ を示す. $\forall g \in \mathcal{M}_1$ に対して, $L(g)$ に等しいペトリネット言語を持つペトリネット N を構成する. $g = (\mathcal{P}, A)$ に対し, 以下の要領で $N = (P_N, T_N, E_N, M_0, \mu)$ を作る. μ はラベル付関数である. まず, P に含まれる規則に出現する終端記号に添字を付け, 規則の異なるところに出現する終端記号は互いに区別できるようにしておく. $\mathcal{P} = (\Sigma_N, \Sigma_T, S, P)$, $A = (K, \Sigma, \delta, g_0, F)$ とする.

構成手順 作られるペトリネット N の P_N は, 以下の手順で作られる p_u の形と k 記号の集合, T_N は, t_g^x の形と k 記号の集合, E_N は (p_u, t_g^x) または (t_g^x, p_u) の形と k 2 項組の集合である. $K_\alpha = \{g \mid \delta(g, \alpha) = g' \text{ for some } g' \text{ in } K\} \ (\alpha \in \Sigma)$ とする. (k 記号, 終端記号の添字は無視して考える.)

(I) 開始記号 S に対して p_S を作る. 新しい p 節点は現れる

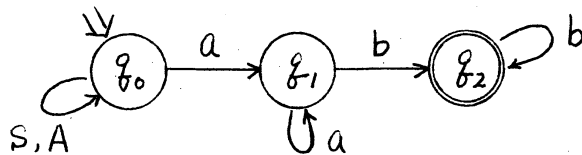
くする。以下の手続を繰り返す。 p 節点を $p_{\alpha_1}, \dots, \alpha_n$ とする。 α_i ($1 \leq i \leq n$) に対して以下の操作を行なう。

i) $\alpha_i = a$ ($a \in \Sigma_T$) とする, $\forall g \in K_a$ に対して, $t_{\alpha_i}^g$ と $(p_{\alpha_1}, \dots, \alpha_n, t_{\alpha_i}^g)$ とを付ける。 ii) $\alpha_i = Aa$ (又は, $A = S, a = \lambda$) とする, $\forall g \in K_A$ に対して, t_{Aa}^g と $(p_{\alpha_1}, \dots, \alpha_n, t_{Aa}^g)$ とを付ける。 $A \rightarrow B_1b_1 | B_2b_2 | \dots | B_kb_k$ 及び P の元であるとする。 $(t_{Aa}^g, p_{B_1b_1}, \dots, b_kb_k)$ とを付ける。 $a \neq \lambda$ とするときは, p_a と (t_{Aa}^g, p_a) とを付ける。 $i \leq k$, $p_{B_1b_1}, \dots, b_kb_k$ P 以前に付けられていない場合は, 二重を付ける。

(II) $\forall g \in K$ に対して, p_g を付ける。 $\forall g, t_x^g$ に対して, $\delta(g, x) = g'$ とおけば, $(p_g, t_x^g), (t_x^g, p_{g'})$ とを付ける。

(III) $M_0(p_s) = M_0(p_{g_0}) = 1$, 他の節点に対しては $M_0(p) = 0$ である。

(例 2) $\mathcal{S} = (\mathcal{P}, \mathcal{A})$, $\mathcal{P} = (\Sigma_N, \Sigma_T, S, P)$, $P = \{S \rightarrow Ab, A \rightarrow Sa | a \in \Sigma\}$, $\mathcal{A} = (Q, \Sigma, \delta, g_0, F)$ は下の推移図で与えられるものとする。



$L(\mathcal{S}) = \{a^i b^i \mid i \geq 1\}$ である。

δ に対して作られるペトリネットは $P \in P' = \{ S \rightarrow Ab, A \rightarrow Sa_1 | a_2 \}$ に対応する。単純に使うと図6のようになる。

(例終り)

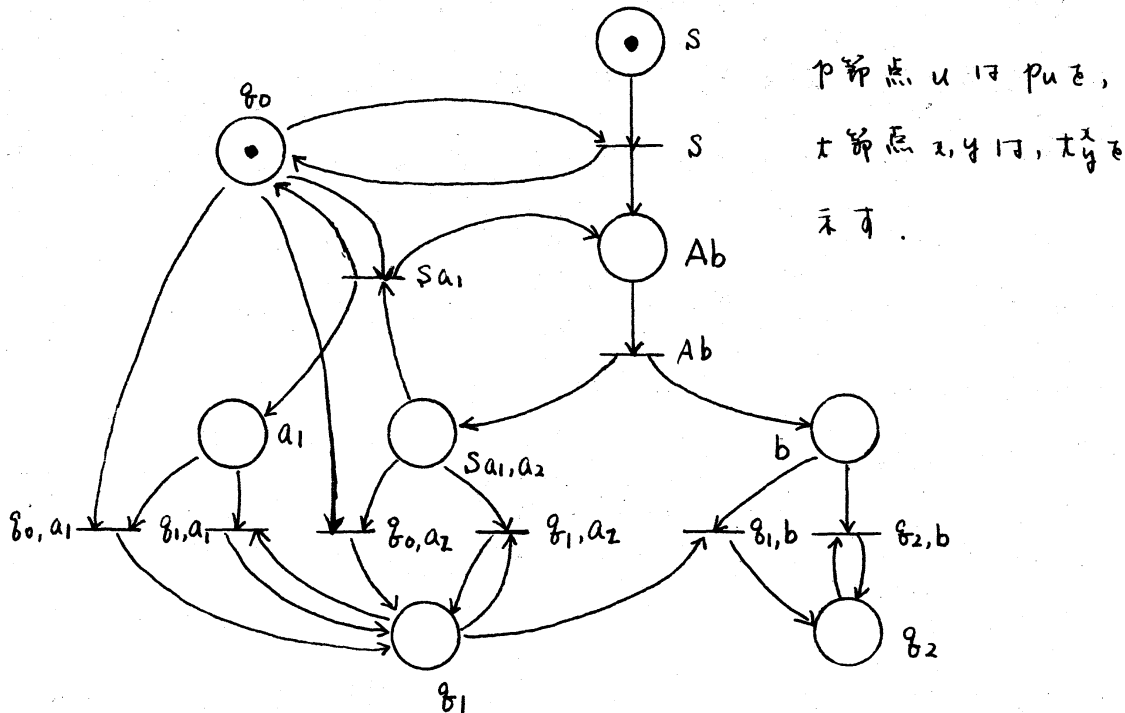


図6 δ に対して作られるペトリネット

(IV) 最終マーカーの集合は $\{ M \mid \sum_p M(p) = 1 \wedge (M(p_0) = 1 \Leftrightarrow p \in F) \}$ である。

(V) μ : ラベル付関数は,

$$\mu(t_{xy}^z) = \begin{cases} \alpha : \alpha \in \Sigma_T \text{ (添字を無視する)} \\ \lambda : \text{その他} \end{cases}, \text{ である. (構成終り)}$$

B) $\exists L$ s.t. $L \in \mathcal{L}_{PN}^\lambda \wedge L \neq \mathcal{L}_1$ を示す. $L = \{ a^i b^i a^i b^i \mid$

$i \geq 1 \}$ とすれば証明できる。証明は、それほど自明ではない

いので、ここでは省略する。(2) $\square$

文脈自由言語の族と示す。

《定理5》 (i) $\mathcal{L}_1 \neq \mathcal{R}$ (ii) $\mathcal{L}_1$ と示す比較不可能である。

(略証) (i) $\mathcal{L}_1 \supseteq \mathcal{R}$ は容易, $\mathcal{L}_1 \neq \mathcal{R}$ は例2より明かす。

(ii) $\mathcal{L}_1$ と示すは, 定理4と文献(1)より明かす。

示す $\mathcal{L}_1$ は $L = \{a^i b^i a^i \mid i \geq 1\} \in \mathcal{L}_1$ と示すこと示す。

$\mathcal{S} = (P, A)$, $P = (\bar{Z}_N, \bar{Z}_T, S, P)$, $P = \{S \rightarrow Aa, A \rightarrow Ba, B \rightarrow sb \mid b\}$, $A = (K, \bar{Z}, \delta, q_0, F)$ は図7の推移図 K であり, と与えられる。 ▢

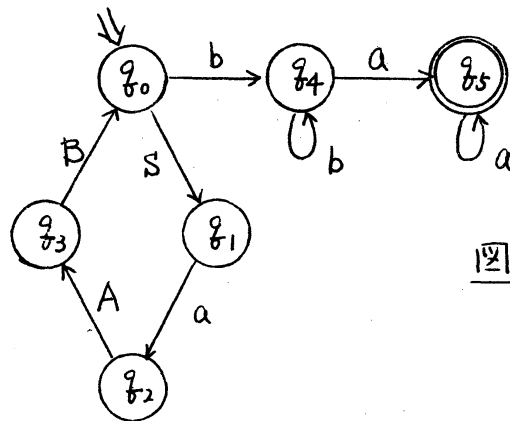


図7 推移図

以上より得られた結果を図8に示す。

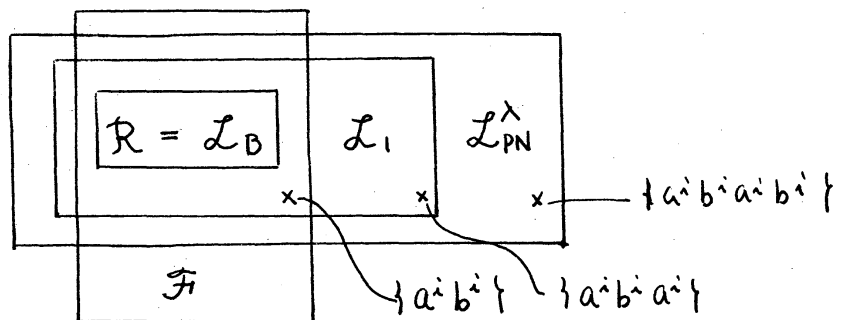


図8. $\mathcal{L}_1$, $\mathcal{L}_B$ と他の言語族との関係

4. あとがき

並行処理システムのモデルSPSを定義し、SPS言語を用いて、SPSのいくつかの部分クラス的能力を考察した。ここでは、特に、左線形文法と有限オートマトンから定義されるSPSを中心に考察した。本稿に含められなかった多くの問題については機会を改めて報告する。

謝辞 御指導を賜る本学福村晃夫教授、三重大学大山に通夫助教授、日頃熱心に御討論を頂く、阿曾弘貝講師をはじめとする福村・本多研究室の方々に感謝する。

(文献)

- (1) M. Hack, "Petri net Languages", Computer Structure Group Memo 124 project MAC
- (2) 山下, 稲垣, 本多, "同期付生成システム(SPS)-新しい並行処理モデル" 信学会オートマトンと言語研究会, 1980.3 発表予定