目次

セルペトリネットツール

1.はじめに

表計算ソフトの WorkSheet のCellに入力して、各々の思うとおりの様々な処理を行うことが出来る。又はマクロを使えば、静的な挙動ならばほとんどのことは表計算ソフトでこと足りる。だが、動的な動きを目で確認したいときや、並列処理などのシミュレーションはプログラムを書かなければならないことが多い。

そのようなときに、表計算ソフトのように手軽に試すことの出来るものを作ることが出来ないかと考えた結果セルペトリネットツールというものを作ってみたら、どのようなことが出来るのかと考察し作った。

処理手順はファイルを読み込みN行M列のセルペトリネットを自動で生成し、Token が Cell 内に存在すると、その Cell に対応する画面上の点(1dot)として表示される。そして処理結果をファイルへ書き出す。

分散処理を行う上で必要な機能を盛り込んでみた。そして、このセルペトリネットでは様々なシミュレーションを目で見ながら、誰でも簡単に使うことが出来るようになるようにした。

2.Cellの定義

2.1 Petri Net の例

Color Petri Net の例を使って説明したいと思います。この場合は Token に色が付いており、それぞれ、blue,red,yellow の色を持っています。Place p_0 には blue,red があり、Place p_1 には yellow の Token が入っています。この状態で Transition t_0 が発火すると blue,red が Place p_1 に移動することになります。

P(Place)
　P={p_0,p_1}

T(Transition)
　T={t_0}

I(InputPlace)
　I={[p_0,t_0]}
O(OutputPlace)
　O={[t_0,p_1]}

CS(ColorSet)
　CS={red,yellow,blue}

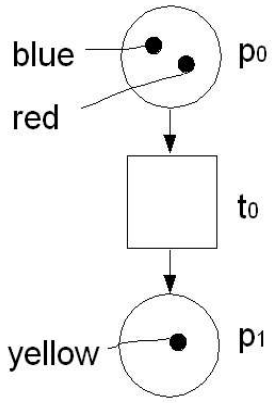

Figure 1: Example.

Token は CS の値を持ち、移動する。
m は各 Place の Token(mark)の数である。

　　m(p_n)： Place p_n の中に入っている Token の数

Fig.1の例では
　　m={m(p_0),m(p_1)}={2,1}

p_0 から t_0 への矢印を Arc と呼び、矢印1本で重みを1とする。1本の矢印を通して Token が1個移動します。

p_0 から t_0 への矢印の重みを A([p_0,t_0])と記します。

発火条件は
　m(p_0) ≧ A([p_0,t_0])
を満たします。

2.2 Cellとは

1つの Cell について考える。まず、どのような場合にも、Place と Transition の組み合わせによって処理できるようにするための最小単位を Cell と考える。情報と処理を組み合わせることにより、将来的に擬似神経細胞集合体のような働きをするものと考える。

まず、Place と Transition を一組にして、Place から Transition へのアークを加える。

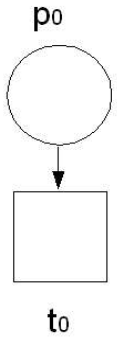

Figure 2: A Petri net cell.

PTN=(P,T,I,O)

$P=\{p_0\}, T=\{t_0\}, I=\{[p_0,t_0]\}, O=\phi$

この一組で1 Cell（ c_0)とする。

$c_0=\{p_0,t_0,[p_0,t_0], \phi\}$

通常1 Cell だけでは処理できない。Arc をもう1本$[t_0,p_0]$と接続させ $c_0=\{p_0,t_0,[p_0,t_0],[t_0,p_0]\}$ を考えるか、2 Cell を使い、それぞれつながないとデッドロックしてしまう。

2.3 Cell Petri Net の定義

定義 1 PTN(Place Transition Net)

P :Place , T :Transition , I :InputPlace , O :OutputPlace と置くとする。

$n,m \in NaturalNumber$

$P = \{p_1,p_2,\ldots,p_n\}$

$T = \{t_1,t_2,\ldots,t_m\}$

$P \cap T= \phi, I \subseteq P \times T, O \subseteq T \times P$

この今回開発した Cell Petri Net Tool の場合には n = m とし同数とする。

$C = \{c_1,c_2,\ldots,c_n\} = \{\{p_1,t_1,f_{1\ in},f_{1\ out}\},\{p_2,t_2,f_{2\ in},f_{2\ out}\},\ldots,\{p_n,t_n,f_{n\ in},f_{n\ out}\}\}$

f_{in}, f_{out} は定義2で説明する Arc のことである。

定義 2 Arc について

定義1で示した P,T を仮定し、f_i を i 番目のアークとすると、

アークの集合:$F \subseteq (P \times T) \cup (T \times P)$

$F=\{f_{1in},f_{2in},\ldots,f_{kin},f_{1out},f_{2out},\ldots,f_{kout}\}$

$f_{in} = [p,t] \in P \times T$ ：Place から Transition へ Cell 内でのアーク

$f_{out} = [t,p] \in T \times P$ ：Transition から Place への Cell 外に出るアーク

アークの重み（本数）は

$A:F \rightarrow \{1,2,\ldots\}$

この Cell Petri Net Tool の場合は Arc の重みは1なので、常に Place に Token が入り次第、その出力先にある Transition が発火(fire)可能となる。

$A:F \rightarrow \{1\}$ $A([p,t])=A(f_{in})=1$

通常 Place から Transition へ Arc の本数を Token が超えたら、Arc の数だけ Token を送り出せる。

定義 3 入力 Place と出力 Place の集合

$t \in T$:Transition の集合

$f \in F$:Arc の集合

$^*t=\{p:p \in P \text{ and } (\exists f)(f \in I \text{ and } f=[p,t])\}$

Place から Transition への Arc を有する全ての Place の集合

$t^*=\{p:p \in P \text{ and } (\exists f)(f \in O \text{ and } f=[t,p])\}$

Transition から Place への Arc が指す Place の集合

定義 4 Token と CS(ColorSet)値

$cs_i \in R$　　非負整数
$n,q \in NaturalNumber$
　　Token は以下の CS 値を持つ
　　$CS=\{cs_0,cs_1,\ldots,cs_i,\ldots,cs_n\}$　　例 $cs_0=\{angle\},cs_1=\{color\},cs_2=\{slow\},cs_3=\{wait\}$
　　$cs_i=\{\varepsilon_i : \varepsilon_i$ は要素値を数えるもの$\}$
$0 \leqq \varepsilon_i < q$
　しかし、CellPetriNetTool では $0 \leqq \varepsilon_i < 2^{32}/2-1$ の範囲内とする。

定義 5　Transition t が発火可能
　　マーキング：$m \in M$
　　Transition　：$t \in T$
　　$^{\forall}p \in {}^*t \quad m(p) \geqq A([p,t])$

　　t に関する発火条件は*t に属する Place 中の Token の数が A([p,t])より多い場合に発火可能となる。

定義 6 発火
　　Transition t への入力 Place p_j は s～u 番までのあると仮定すると、

$$p_j \in {}^*t \quad \sum_{j=s}^{u} m(p_j) \geqq A([p,t])$$

　　という発火条件 ρ を満たす。

　　発火直後、Transition t は k～r 番までの Place p_i に出力すると Token は

$$p_i \in t^* \quad \sum_{i=k}^{r} m(p_i) \geqq A([t,p])$$

　　t が発火するとその後 Place のマーキング m'は
　　$m'(p) = m(p) - A([p,t]) + A([t,p])$

　　また、発火する際に Transition の処理によって Token の CS 値が変わる場合がある。

定義 7　CPN(CellPetriNet)
　　PTN　を PlaceTransitionNet とする、CS を集合とする。
　　ρ　　：発火条件 (References[1])
　　$CPN=(PTN,CS,\rho)$

　　1 Cell の中にはp,tが一つずつ、それに接続する p から t の Cell 内でつながる Arc f_{in} と t から p の Cell どうしをつなげる Arc f_{out} を合わせて Cell と呼ぶことにする。
　　$C=\{c_1,\ldots,c_i,\ldots,c_n\}$
　　定義1を拡張し　i 番目の Cell の接続と処理を考えると
　　$^{\exists}i \quad c_i = \{ p_i,t_i,f_{i\,in},f_{i\,out}\}$

Cell Petri Net は p,t が多数で 1Cell としても良いのだが、構造が複雑になることと、p,t を1Cell として多数組み合わせれば、p,t が多数の場合と同じ働きが可能なのであえて p,t で1Cell とした。

3．1次元配列の Cell について

3.1 Cell の接続

次に Fig.2 で表した、一つの Cell Petri net を直線に接続することを考える。

$P = \{p_1,\ldots,p_n\}$

$T = \{t_1,\ldots,t_n\}$

$I = \{[p_1,t_1],\ldots,[p_n,t_n]\} = \{f_{1\,in},\ldots,f_{n\,in}\}$

$O=\{[t_1,p_2],\ldots,[t_{n-1},p_n],[t_2,p_1],\ldots,[t_n,p_{n-1}]\} = \{f_{1\ Rout},\ldots,f_{n-1\ Rout},f_{1\ Lout},\ldots,f_{n-1\ Lout}\}$

$f_{i\ out} \equiv \{f_{i\ Rout},f_{i\ Lout}\}$ と置くとする。
ただし　　i=1　$f_{1\ out} \equiv \{f_{1\ Rout}\}$　,　i=n　$f_{n\ out} \equiv \{f_{n\ Lout}\}$

$C=\{c_1,\ldots,c_n\}=\{\{p_1,t_1,f_{1\ in},f_{1\ out}\},\ldots,\{p_n,t_n,f_{n\ in},f_{n\ out}\}\}$

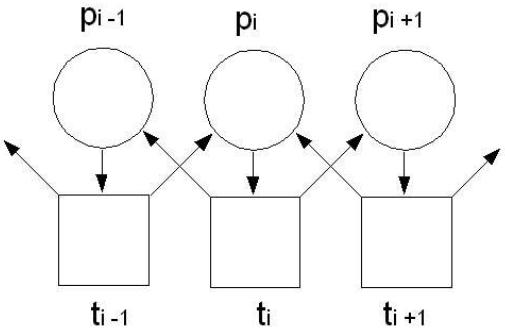

Figure 3: Linear Petri net cell.

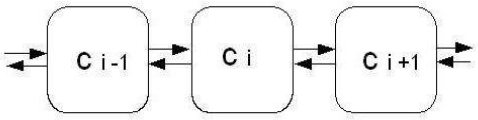

Figure 4:Linear Petri net cell Upside.

例えば、左側から c_{i-1} に CS 値が 0 の Token が入ってきたとする、

c_{i-1} は無条件に c_i に Token を渡す
c_i は 1 加算して c_{i+1} に Token を渡す
c_{i+1} では CS 値が 100 より小さければ c_i へ、そうでなければそのまま右側へ

上記の分岐条件を与えると、$c_{i-1}\rightarrow c_i\rightarrow c_{i+1}\rightarrow c_i\rightarrow c_{i-1}\rightarrow$ … と 100 回繰り返すことになる。
これにより、Cell Petri net は c_{i-1} , c_i , c_{i+1} があればループ条件が作り出せることが分かり、このように、Transition に分岐条件、Token に値を与えると様々な処理ができるようになります。

4. 2次元配列の Cell について

一つのセルには
$^{\exists}$ i,j∈NaturalNumber

$P=\{p_{ij}\},T=\{t_{ij}\},I=\{[p_{ij},t_{ij}]\},$

$$O_{ij}=\begin{cases}
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j-1}],[t_{ij},p_{i\ j+1}]\} & 0<i\pm1<N,\ 0<j\pm1<M \text{ 中心部}\\
\{[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j+1}]\} & i=1,j=1 \text{ 左上端}\\
\{[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j-1}],[t_{ij},p_{i\ j+1}]\} & i=1,0<j\pm1<M \text{ 上端}\\
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j+1}]\} & 0<i\pm1<N,j=1 \text{ 左端}\\
\{[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j-1}]\} & i=1,j=M \text{ 右上端}\\
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i\ j+1}]\} & i=N,j=1 \text{ 左下端}\\
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i+1\ j}],[t_{ij},p_{i\ j-1}]\} & 0<i\pm1<N,j=M \text{ 右端}\\
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i\ j-1}],[t_{ij},p_{i\ j+1}]\} & i=N,0<j\pm1<M \text{ 下端}\\
\{[t_{ij},p_{i-1\ j}],[t_{ij},p_{i\ j-1}]\} & i=N,j=M \text{ 右下端}
\end{cases}$$

それ以外の i,j は
$P=\phi$, $T=\phi$, $I=\phi$, $O=\phi$

簡略の為、N 行 M 列のセルペトリネットの場合には以下の様に記す。

$[p_{nm}]= p_{11}, \ldots, p_{1M},$
$\vdots \quad \vdots$
$p_{N1}, \ldots, p_{NM}$

以下[ ]で括ってあるものは同様
$^{\exists}n,m \in NaturalNumber$
$0<n\pm1<N,\ 0<m\pm1<M$
$P=\{[p_{nm}]\}$
$T=\{[t_{nm}]\}$
$I=\{[p_{nm},t_{nm}]\}$
$O=\{[t_{nm},p_{n\pm1\ m\pm1}]\}$ ただし $p_{0m},p_{n0},p_{Nm},p_{nM}$ の場合は ϕ
$C=\{[c_{nm}]\}$
$n,m \leqq 0 \quad N,M<n,m$
$P=\phi$, $T=\phi$, $I=\phi$, $O=\phi$, $C=\phi$

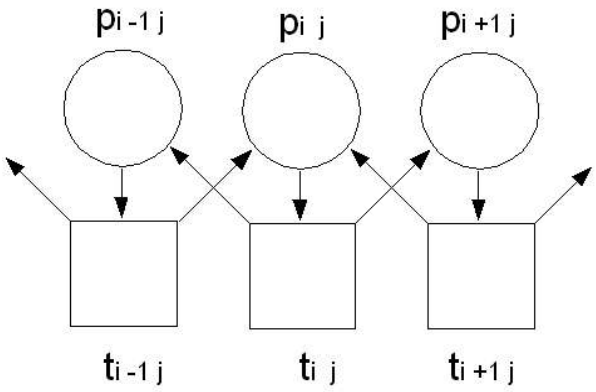

Figure 5: Sheet Petri net cell column j

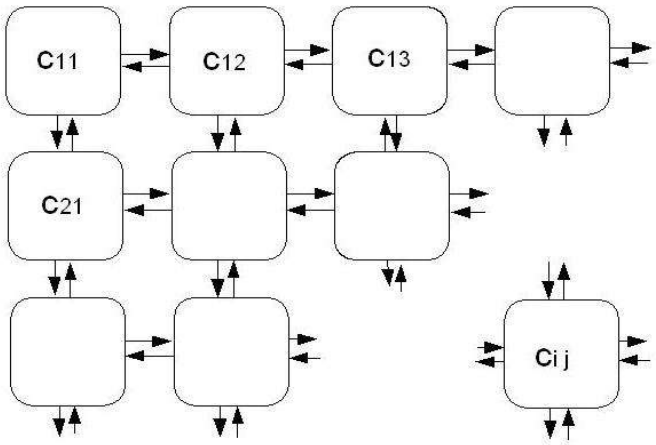

Figure 6: Sheet Petri net cell.

通常 Cell 内の Place に Token 1つ入ることが FiringRule を満たすので、すぐに、Cell 内の Transition が発火し、Token の入った方向と反対方向の Cell 内の Place へ Token をわたす。

Firing rule は ρ : $^{\forall}p_{ij} \in {}^{*}t_{ij}\ \ m(p_{ij}) \geqq A(p_{ij},t_{ij})$を満たすことである。

そして発火した場合
$t_{ij} \in T_{ij}$

$$O_{ij\,kj}(t_{ij}) = \begin{cases} \{p_{i+1\,j}\} & \cdots \text{ if } k=i+1 \\ \{p_{i-1\,j}\} & \cdots \text{ if } k=i-1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

$$O_{ij\,ik}(t_{ij}) = \begin{cases} \{p_{i\,j+1}\} & \cdots \text{ if } k=j+1 \\ \{p_{i\,j-1}\} & \cdots \text{ if } k=j-1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

角度を、画面上で

$$\begin{array}{ccc} & \uparrow 90^\circ & \\ 180^\circ & \leftarrow \quad \rightarrow & 0^\circ \\ & \downarrow 270^\circ & \end{array}$$

だとすると、

$$\rho_{\,ij\,ik}(t_{ij},(p_{ij},0^\circ)) = \begin{cases} \{(p_{i+1\,j},0^\circ)\} & \cdots \text{ if } k=i+1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

$$\rho_{\,ij\,ik}(t_{ij},(p_{ij},180^\circ)) = \begin{cases} \{(p_{i-1\,j},180^\circ)\} & \cdots \text{ if } k=i-1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

$$\rho_{\,ij\,ik}(t_{ij},\phi) = \phi$$

$$\rho_{\,ij\,kj}(t_{ij},(p_{ij},270^\circ)) = \begin{cases} \{(p_{i\,j+1},270^\circ)\} & \cdots \text{ if } k=j+1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

$$\rho_{\,ij\,kj}(t_{ij},(p_{ij},90^\circ)) = \begin{cases} \{(p_{i\,j-1},90^\circ)\} & \cdots \text{ if } k=j-1 \\ \phi & \cdots \text{ otherwise} \end{cases}$$

$$\rho_{\,ij\,kj}(t_{ij},\phi) = \phi$$

また、角度として、Token の進む方向に対して左が+90° 右が+270° 前方+0° 後方+180°と指定することができる。

また、これを拡張してn次元のペトリネットツールを作ることも考えられるが、多次元にした場合の表示方法が困難である。従って、現在2次元までしか作っていない。

5. CellPetriNetTool の条件について

それぞれの Cell に様々な発火条件を指定することが出来るので、CS 値のちがった Token の流れを制御することができる。従って、このことにより、ペトリネットの処理がシミュレーション可能となる。

5.1 Token(mark)

この CellPetriNetTool では Token は Place 数次元の非負整数ベクトル R
Token は CS(ColorSet)という R 値を4個もつ。
従って、定義 4 で使っている n を n=3 とおいて考えるものとする。

$CS=\{cs_0,cs_1,cs_2,cs_3\}$ $cs_0 \in R, cs_1 \in R^3, cs_2 \in R, cs_3 \in R$
$cs_0=\{\text{angle}\}$, $cs_1=\{\text{rgb}\}$, $cs_2=\{\text{slow}\}$, $cs_3=\{\text{wait}\}$

cs_0は角度 0,90,180,270,…に対応する。
$\forall a \in cs_0 \quad \exists m \in NaturalNumber$
$a = 90m$

cs_1は rgb(赤緑青)に対応させ、R^3 すなわち 0～256×256×256-1=2^{24}-1 までの値とする。
2 進数であらわすと
r g b
00000000|00000000|00000000
なので、
$\forall c \in cs_1 \quad \exists r,g,b \in NaturalNumber \quad 0 \leqq r,g,b < 2^8$
$c = 2^{16}r+2^8g+b$
となる。

cs_2は遅延速度(slow)である。
$s \in cs_2 \quad s \in NaturalNumber \quad 0 \leqq s < 2^{32}/2-1$
本来発火条件を満たしても、s 値が変わらない限り、常にあらゆる Place で Token は s 回処理を待つ。
s 値が高いと Token の進行速度が遅くなる

cs_3は待ち(wait)である。
$w \in cs_3 \quad w \in NaturalNumber \quad 0 \leqq w < 2^{32}/2-1$
本来発火条件を満たしても、1つの Place 内で1度だけ Token は w 回処理を待つ

Token は通常 cs_0に与えられている角度方向(直線的)にしか動かない。Transition で Token のCS値に変化を与えることにより動作が変化する。
CS 比較値は Transition 内にあり、Token の CS 値と比べ、条件が一致すると Token の CS 値を変えて発火する。

Cell Petri Net Tool では初期 marking(Token の配置)は File から読み込む

5.2 Place
Token は Place 内で発火するまで待っている。
32 ビット CPU の場合、Place 内には Token が $2^{32}/2-1$=2147483647 個入れられるものとする。

$P=\{[p_{ij}]\}$
i 行 j 列の Place である p_{ij} 中の Token 個数は m 個である。m の値は 0～$2^{32}/2-1$ までである。
$0 \leqq m(p_{ij}) < 2^{32}/2 \qquad m(p_{ij}) \in NaturalNumber$

5.3 Transition
Firing rule(発火条件)を満たして、かつ、Token の cs_i 値もしくは、Transition の設定によって、Token の進行方向、色、速度は変わる。
Transition 内にも Token の CS 値と比較し、条件を満たしていると、Token の CS 値を変化させる CS 比較値というものがある。
Token の CS 値の変更はセルにおいて Token の cs_i 値と Transition セル内の CS 比較値との比較によってなされる。
Transition を起こすと、必ず、Token が移動する。ただし、Token の cs_3 (待ち)が 0 である場合に限る。

5.4 タイムペトリネット(timed Petri net)の導入
発火遅延時間モデル、無反応(待ち)条件、遅延などの時間的な操作が可能である。
タイムペトリネットの Cell とタイムスタンプ機能をもった Token 両方の設定が可能であり、Cell を設定しておくと、Token が Cell 内での処理を遅らせたり、待っていたりすることが出来る。また、Token 自身に counter が付いていて処理速度を遅く定義できる。

6.Cell Petri Net Tool におけるモジュール化

Cell Petri net tool をいくつか組み合わせることによって、さまざまな論理回路が可能となる。また、1つの Cell にある働きを与えることができる。

6.1 Change Angle Cell

通常は直線的に Token を渡すだけであるが、無条件に、もしくは条件によって Token の進行方向を変えることができる。これによって、Token の動く方向を変えることができる。

6.2 Calculate Cell

Cs の値の計算ができる。現在使えるのは ＋ － ＊ ／ ＾ √ の6つの計算ができる。

6.3 Timer Cell

Token の進行速度や、待ちの時間を付加できる。

6.4 例

(1)Token の静止する場所を作る

2 つの Cell で停止モジュールと考え、それぞれの Cell に移すようにすれば Token を閉じ込めることができる。

(2)ループ回路

(1)の回路を離して設置し、2 つの Cell の間を繰り返し通るようにしておき、その間で決められた演算（例えば C 言語の for 文のようなもの）を行う。

(3)同期、並列処理回路

ある Cell を通ると速度が遅くなるように設定しておけば、別経路を通った Token どうしが同期して同じ Cell に入ることができる。

7. Cell Petri Net Tool の実行ファイルへの記入方法

ツールのデフォルト入力でファイル名は Petri.Dat にしてあり、起動時に自動的に読み込むように作ってあります。
又は、別ファイルを読み込みたいときには、

C:¥> Petri ファイル名

と入力しても良い。

7.1 画面表示

Petri.Dat ファイルの一番最初に書き、セル画面のサイズを変えることができます。また、Token の動いた後を消さずに残すことも可能です。

Cell[X 座標,Y 座標]
Cell[200,300]　　…200×300 のセルを作る、動作経路を表示しない。
Cell[200,300]0　　…200×300 のセルを作り、動作経路を消さない。

7.2 Token の値

初期マーキングの CS 値を設定できます。最初の位置xy、進行方向、色、遅延速度、停止時間を決めることが可能です。

m(X 座標,Y 座標){角度 0～270,値 0～16777215,遅さ 0～,待ち 0～}
m(50,20){90,16777215,1,0}　Token の属性は位置 x50y20,角度 90,最大,遅さ,待ち

位置:x,y
進行角度:0,90,180,270,(停止は-1)
値:0～16777215=(256×256×256)-1
遅さ:0～
待ち:0～

7.3 Transition の判定基準

発火中に Token 値を変えることが出来ます。ここでは各値を演算することによって変化させることが出来ます。また、Token 値と比較し、一致すれば、演算をし、一致しなければそのまま変化させないこともできます。

cell(X 座標,Y 座標){角度 0～270,値 0～16777215,遅さ 0～,待ち 0～}判断

cell(5,10){,+3,,}=6　cs_1が6ならば、Cell(5,10)で cs_1 に3をたす
cell(5,10){90,,,}　Cell(5,10)で絶対角度 90°　方向へ曲がる
cell(5,10){,*3,,}　cs_1を3倍する

以下の演算が出来る

+ － * / ^ √

以下の cs_0,cs_1,cs_2,cs_3 で適応することが出来る。

cell(,){cs_0,cs_1,cs_2,cs_3}

次の判断が出来る。符号 S(“>”,”=”,”<”),条件を d とすると、

> = <

cell(,){cs_0,cs_1,cs_2,cs_3}Sd

ただし、cs_0,cs_1,cs_2,cs_3と記入してある CS 値の最も左側と d の値を S によって比較する。したがって、記入されていないと cs_i 値は飛ばされ次の cs_{i+1} 値との比較になる。

7.4 表記例

下記に Petri.Dat の表記例を示す。

```
Cell[150,150]0  //150×150 のセルを作る。動いた飛跡は消さない。
times(5000)  //5000 回繰り返す

cell(50,100){0,,,}    //セル 50,100 の位置に来たら、無条件に 0 度へ Token の方向を変える
cell(100,50){+90,,,}>0 //セル 100,50 の位置で Token の cs0 が 0 より大きければ 90 度 cs0 に加える。
cell(20,100){,/2,,} //rgb 値を半分にする。

// mark(位置){角度,値,遅さ,待ち}
m(100,50){180,5500000,20,0}
m(50,100){270,860,50,0}
```

8. 結び

このCellPetriNetToolは動的な動きを画面上に描き出せ、どの様な処理をしているかが一目で分かる。
セル各々が独立処理であるので、一部のバグがあり、その部分は止まっても他の部分ではHangしない。
セルペトリネットを使いペトリネットの処理を行う。
シミュレーション(数値計算)を行うことが出来る。
などの利点があります。
まだまだ、CellPetri net の今後の動向が注目されると考えられます。

References

[1] Pauline N.Kawamoto and Yatsuka Nakamura, On Cell Petri Nets, Shinshu University.
[2] 臼田昭司,井上祥史,伊藤敏　Lotus1-2-3 による理工系シミュレーション入門　1993
[3] 平林雅英　C++版 Windows95 プログラムを 10 倍強力に作る　共立出版　1996
[4] 柏原正三　C++効率的最速学習徹底入門　技術評論社　2002
[5] 川端一生　真田良蔵　C/C++言語ハンドブック　ナツメ社　1999.9.18
[6] C/C++300 の技　技術評論社　平成 13.11.1
[7] 森下信　セルオートマトン　養賢堂　2003.3.25
[8] 加藤恭義・光成友孝・築山洋　セルオートマトン法　森北出版　1998.10.30
[9] 臼田昭司・東野勝治・井上祥史・伊藤敏・葭谷安正　カオスとフラクタル　オーム社　1999.2.26

Appendix

プログラム

```
//============================petri.cpp=======================================

// Cell Petri Nets の model 化を使ったシュミレーション
/*
○:Place        (mark)
↓:Arc          (PlaceID,TransitionID)
□:Transition   (fire 条件)
*/
#include <string>
#include <vector>
#include <fstream.h>
#include <iostream.h>
#include <iterator>
#include <algorithm>
#include "strcut.cpp"
#include "fread.cpp"
#include <sstream>

#include <stdlib.h> // EXIT_SUCCESS
#include <math.h>   // cos

//グラフィックライブラリは、BCC では動かない箇所を修正しておき、
//Windows95 プログラムを 10 倍強力に作る[C++版]平林雅英 さんのを使わせて頂きました。
//コンパイラは BorlandC++  BCC
#include "sls¥screen_int.fx" // 画面初期化関数
#include "sls¥CX_CY.m"          // X,Y 軸中心値マクロ CX,CY
#include "sls¥keyread.fx"       // キー読取関数
#include "sls¥pset_mode_set.fx" // 描画モード設定関数
#include "sls¥pset.fx"          // 論理点関数

class Arc_ {
private:
public:
        int id_;     // 出力先 ID
        int angle; // 接続方向
        int setID()    const {return id_;   }
        int arc(int t)
        {
                id_   = t;
                return 0;
        }
        friend inline istream& operator>>(istream&,Arc_&);

};


class Mark {
private:
public:
        int angle;  //方向   当面 0 右,90 上,180 左,270 下だけとする。
        int value;    //値   一応 RGB(0,0,0)～RGB(255,255,255)の範囲内としておく
        int slow;     //遅さ
        int wait;     //待ち

        void clean()
        {       angle = 0;
```

```
		value =0;
		wait =0;
		slow=0;
		return;
	}
	int Num(int n)
	{
		if(n==0) return angle;
		else if(n==1) return value;
		else if(n==2) return wait;
		else if(n==3) return slow;

		return -2;
	}
	void inMarkPoint(int n,int v)
	{
		if(n==0)
		{
			for(;v<=-90;v=v+360);
			for(;v>=360;v=v-360);
			angle=v;
		}
		else if(n==1) value=v;
		else if(n==2) wait=v;
		else if(n==3) slow=v;
		return;
	}
	friend inline istream& operator>>(istream&,Mark&);
};


class Places {
private:
public:
	int	_ID;
	int	_fire;
	int  isMark;   // 0:mark が無い 1～:mark がある。(-1～:mark 処理が 0 になるまで行う)
	vector<Mark> _mark;
	vector<Arc_> ar;

	Places(){}
	Places(int i,int f,int& t){
		_ID   = i;
		_fire = f;
	}
	int places(int i,int f,int& t){
		_ID   = i;
		_fire = f;
		return 0;
	}
	int mark(int t,int n)
	{
		Mark tok;

		tok.angle = t;
		tok.value   = n;
		_mark.push_back(tok);
		return 0;
	}
	void clean()
```

```
        {
                _mark.erase(_mark.begin(),_mark.end());
                _ID = 0; _fire=0;
                return;
        }
        int    setID()    const {return(_ID);  }
        int    setFire()  const {return(_fire);}
        friend inline istream& operator>>(istream&,Places&);
        friend inline bool operator>(const Places&,const Places&);
        friend inline bool operator<(const Places&,const Places&);

        ~Places(){}
};

int IsFugo(string Str)
{
        if(strstr(Str.c_str(),"+")!=NULL)return 1;
        if(strstr(Str.c_str(),"-")!=NULL)return 2;
        if(strstr(Str.c_str(),"*")!=NULL)return 3;
        if(strstr(Str.c_str(),"/")!=NULL)return 4;
        if(strstr(Str.c_str(),"^")!=NULL)return 5;
        if(strstr(Str.c_str(),"√")!=NULL)return 6;
        if(strstr(Str.c_str(),"<")!=NULL)return 10;
        if(strstr(Str.c_str(),"=")!=NULL)return 11;
        if(strstr(Str.c_str(),">")!=NULL)return 12;
        return 0;
}
int delFugo(string Str)
{
string str1,str2;
        if(strstr(Str.c_str(),"+")!=NULL)_strcut_(str1,str2,"+",Str);
        else if(strstr(Str.c_str(),"-")!=NULL)_strcut_(str1,str2,"-",Str);
        else if(strstr(Str.c_str(),"*")!=NULL)_strcut_(str1,str2,"*",Str);
        else if(strstr(Str.c_str(),"/")!=NULL)_strcut_(str1,str2,"/",Str);
        else if(strstr(Str.c_str(),"^")!=NULL)_strcut_(str1,str2,"^",Str);
        else if(strstr(Str.c_str(),"√")!=NULL)_strcut_(str1,str2,"√",Str);
        else if(strstr(Str.c_str(),"<")!=NULL)_strcut_(str1,str2,"<",Str);
        else if(strstr(Str.c_str(),"=")!=NULL)_strcut_(str1,str2,"=",Str);
        else if(strstr(Str.c_str(),">")!=NULL)_strcut_(str1,str2,">",Str);
        return atoi(str2.c_str());
}
string FugoNum(int num)
{
        string str="";

        if(num==1)str="+";
        else if(num==2)str="-";
        else if(num==3)str="*";
        else if(num==4)str="/";
        else if(num==5)str="^";
        else if(num==6)str="√";
        else if(num==10)str="<";
        else if(num==11)str="=";
        else if(num==12)str=">";

        return str;

}

class Transition {
```

```
public:
int ID_; // TransitionNumber
int fire;
vector<Arc_> ar;

int CalPoint;   // -1:計算しない 0:angle 1:value 2:slow 3:wait
int fugo;    // +-*/^√
int num;        // 数値
int hantei;  // > = < の大小判定記号が入る
int V;          // 判定数値

double calc(double a,string c,double b)
{
        if(c=="+")
                return a + b;
        else if(c=="-")
                return a - b;
        else if(c=="*")
                return a*b;
        else if(c=="/")
                return a/b;
        else if(c=="^")
                return pow(a,b);
        else if(c=="√")
                return a*sqrt(b);
}
int calc(int a,string c,int b)
{
        return (int) calc((double)a,c,(double)b);
}
double calc(double a,int c,double b)
{
        if(c==1)
                return a + b;
        else if(c==2)
                return a - b;
        else if(c==3)
                return a*b;
        else if(c==4)
                return a/b;
        else if(c==5)
                return pow(a,b);
        else if(c==6)
                return a*sqrt(b);
}
int calc(int a,int c,int b)
{
        return (int) calc((double)a,c,(double)b);
}


void chg_value(Mark &mrk)
{
        if(CalPoint>=0)//-1 では token の変更はしない
        {
                int v = mrk.Num(CalPoint);//mark に入っている値
                if(v>=0)
                {
                        if(hantei==10){if(!(v<V))return;}
                        else if(hantei==11){if(!(v==V))return;}
```

```
                                else if(hantei==12){if(!(v>V))return;}

                                if(fugo!=0)
                                {
                                        v = calc(v,fugo,num);
                                }else v=num;
                                cout << "in v " << v << endl;
                                mrk.inMarkPoint(CalPoint,v);
                        }
                }
        }

};


//=========================================================================
//  判断
inline bool operator>(const Places& p1,const Places& p2)
{
        return p1._mark > p2._mark;
}

inline bool operator<(const Places& p1,const Places& p2)
{
        return p1._mark < p2._mark;
}

inline bool operator>(const Arc_& a1,const Arc_& a2)
{
        return a1.id_ > a2.id_;
}

inline bool operator<(const Arc_& a1,const Arc_& a2)
{
        return a1.id_ < a2.id_;
}


//=============================================================
//      入出力

inline istream& operator>>(istream& fs,Arc_& A)
{
        fs >> A.id_ >> A.angle;
        return fs;
}
inline ostream& operator<<(ostream& os,Arc_& A)
{
        os << A.id_ << '¥t' << A.angle << '¥n';
        return os;
}

inline istream& operator>>(istream& fs,Mark& T)
{
        fs >> T.value >> T.angle;
        return fs;
}
inline ostream& operator<<(ostream& os,Mark& T)
{
        os << T.value << '¥t' << T.angle << '¥n';
```

```
	return os;
}

inline istream& operator>>(istream& fs,vector<Mark>& p)
{
	vector<Mark>::iterator itS = p.begin();
	vector<Mark>::iterator itE = p.end();

	do{
		fs >> (*itS).value >> (*itS).angle ;
		itS++;
	}while(itS < itE);
	return fs;
}
inline ostream& operator<<(ostream& os,vector<Mark>& p)
{
	vector<Mark>::iterator itS = p.begin();
	vector<Mark>::iterator itE = p.end();

	for(;itS != itE; itS++)
	{
		os << (*itS).value << '¥t' << (*itS).angle ;
	}
	return os;
}

inline ostream& operator<<(ostream& os,vector<Arc_>& p)
{
	vector<Arc_>::iterator itS = p.begin();
	vector<Arc_>::iterator itE = p.end();

	for(int i=0;itS != itE; itS++,i++)
	{
		os << '¥t' << *itS;
	}
	os << endl;

	return os;
}

inline istream& operator>>(istream& fs,vector<Arc_>& p)
{
	Arc_ ig;
	vector<Arc_>::iterator itS = p.begin();
	vector<Arc_>::iterator itE = p.end();
	int i,j,k;

	while(fs >> ig)
	{
		p.push_back(ig);
	}

	return fs;
}

inline istream& operator>>(istream& fs,Places& p)
{
	fs >> p._ID  >> p._fire >> p._mark >> p.ar ;
	return fs;
}
```

```
inline ostream& operator<<(ostream& os,Places& p)
{
	os << p.setID() << '¥t' << p.setFire() << '¥n';
	return os;
}

void MakePlace(vector<Places> &itP,unsigned int x,unsigned int y)
{// x × y 個の Place を作る
	unsigned int i;
	for(i=0;i<x*y;i++)
	{	//Places に対応した Connect を作る
		Places pla;
		pla._ID=i;		// ID を設定入力
		pla.isMark=0;
		itP.push_back(pla);
	}
}

//   ↑    上下左右にアークを作り、つなげる。
// ←P→
//   ↓
//P    P
//↓  ／
//T／
void ConnectArc(unsigned int x,unsigned int y,vector<Transition> &T,vector<Places> &P)
{// place と transition を arc でつなぐ
	vector<Transition>::iterator itT = T.begin();
	vector<Places>::iterator     itP = P.begin();
	Arc_ ar;

	unsigned int i;
	for(i=0; itP<P.end() ;i++,itP++,itT++)
	{	//Arc_を作って Connect と接続する。
		ar.id_=(*itT).ID_; (*itP).ar.push_back(ar); //Places から Transiton への arc
		if( i   %y!=0)
		{	ar.angle=180;
			ar.id_= (*(itP-1))._ID;
			(*itT).ar.push_back(ar);//Transition から左 Places への arc
		}
		if((i+1)%y!=0)
		{	ar.angle=0;
			ar.id_= (*(itP+1))._ID; (*itT).ar.push_back(ar);//Transition から右 Places への
arc
		}
		if(i>x       )
		{	ar.angle=90;
			ar.id_= (*(itP-x))._ID; (*itT).ar.push_back(ar);//Transition から上 Places への
arc
		}
		if(i<x*(y-1) )
		{	ar.angle=270;
			ar.id_= (*(itP+x))._ID; (*itT).ar.push_back(ar);//Transition から下 Places への
arc
		}
	}
}

void MakeTransition(vector<Transition> &T,unsigned int x,unsigned int y)
{
```

```
	unsigned int i;
	for(i=0;i<x*y;i++)
	{
		Transition tra;
		tra.ID_=i;		// ID を設定入力
		tra.CalPoint = -1;
		T.push_back(tra);
	}
}

void MakeCell(vector<Places> &XYp,vector<Transition> &XYt,unsigned int x,unsigned int y)
{
	cout << "Place を製作中" << endl;
	MakePlace(XYp,x,y);

	cout << "Transition を製作中" << endl;
	MakeTransition(XYt,x,y);

	cout << "Place と Transition を Arc で接続中" << endl;
	ConnectArc(x,y,XYt,XYp);
}

void InMark(vector<Places> &P,unsigned int xy,int angle,int value,int slow,int wait)
{
	vector<Places>::iterator itPb = P.begin(),itP;
	itP = itPb + xy;

	Mark tk;
	tk.angle=angle;
	tk.value=value;
	tk.slow = slow;
	tk.wait = wait;
	(*itP)._mark.push_back(tk);
	(*itP).isMark++;		//Place に mark があると印をつける
}


int transition(vector<Places> &P,vector<Transition> &T,unsigned int X,unsigned int Y,int Draw)
{
	unsigned int x,y,x_save=0,y_save=0;
	vector<Places>::iterator itP = P.begin(),itPp;
	vector<Transition>::iterator itTr = T.begin();
	for(;itP!=P.end();itP++,itTr++)
	{//PlaceCell ごとに処理する
		if((*itP).isMark == 0)continue; //mark がないなら次の Place へ

		vector<Arc_> &vAP = (*itP).ar;	//Places から Transition への Arc_
		vector<Arc_>::iterator itAP = vAP.begin();

		vector<Arc_> &vAT = (*itTr).ar;	//Transition から Plases への Arc_
		vector<Arc_>::iterator itAT = vAT.begin();

		vector<Mark> &vM = (*itP)._mark;	//Place の中の mark
		vector<Mark>::iterator itM = vM.begin(),itMmp;

		Transition &tr=(*itTr);

		for(unsigned int k=0;itM != vM.end();itM++,k++)
		{//一つの PlacesCell の中に入っている Mark 群について処理する
			Mark &mrk=(*itM);
```

```
if(mrk.wait!=0){mrk.wait--;continue;}//待ちの場合は次 mark の処理をする
if(mrk.value!=0)
{

        y=(*itP)._ID / X;
        x=(*itP)._ID -X*y;

        //token の値を変更する！----------------------------------
        tr.chg_value(mrk);
        //-----------------------------------------------------

        // mark が Cell の上下左右の端にあたった場合 firing rule 2
        if(mrk.angle==0)
        {       if(x==X-1){mrk.angle=180;}
        }else if((*itM).angle== 90)
        {       if(y==0){mrk.angle=270;}
        }else if(mrk.angle==180)
        {       if(x==0){mrk.angle=  0;}
        }else if(mrk.angle==270)
        {       if(y==Y-1){mrk.angle= 90;}
        }
        //次 Place への mark の移動----------------------------------
        // Transition の中で mark が進む同じ angle の Arc_を探す
        for(;itAT!=vAT.end();itAT++)
                if((*itAT).angle==mrk.angle)break;
        itPp = P.begin()+(*itAT).id_;

        Mark mk;
        mk.angle =mrk.angle; mrk.angle =0;
        mk.value   =mrk.value;   mrk.value   =0;

        if(mrk.angle == 0 || mrk.angle ==270)
        {//1 つのセルが完全分散処理になればここで wait に 2 を足すこと
はいらない
                mk.wait = mrk.wait + mrk.slow + 2 /* 進む速度の調整の為 2
にしておく */;
                mrk.wait =0;
        }
        else
        {       mk.wait = mrk.wait + mrk.slow ; mrk.wait =0;}

        mk.slow = mrk.slow ; mrk.slow =0;
        //次の Place へ token を移す。
        {
                Places &Pp=(*itPp);
                Pp._mark.push_back(mk);
                Pp.isMark++;
        }
        vM.erase(vM.begin()+k); //移した後、残った Mark を消す。
        (*itP).isMark --;
        vM = (*itP)._mark;
        itM = vM.begin();
        COLORREF rgb;
        if(mk.value>RGB(255,255,255)) rgb=RGB(255,255,255);
        else rgb = mk.value ;
        pset(x,y,rgb);
        if(Draw!=0)     pset(x_save,y_save,RGB(0,0,0));
        x_save=x, y_save=y;
break;
```

```
                        }
                }
        }
        if(Draw!=0)pset(x_save,y_save,RGB(0,0,0));
}

int inputline(int MaxX,int MaxY,int x,int y)
{
        int i=y*MaxX+x;
        if(i<MaxX*MaxY)return i;
        else return 0;
}

void InTransition(vector<Transition> &T,unsigned int xy,int calpoint,int fu,int n,int han,int v)
{
        vector<Transition>::iterator itTb = T.begin(),itT;
        itT = itTb + xy;

        (*itT).CalPoint=calpoint;   // -1:計算しない 0:angle 1:value 2:slow 3:wait
        (*itT).fugo=fu;    // +-*/^√
        (*itT).num=n;        // 数値
        (*itT).hantei=han;  // > = < の大小判定記号が入る
        (*itT).V=v;          // 判定数値
}

int setXY(unsigned int &X,unsigned int &Y,vector<Places> &P,vector<Transition>
&T,vector<string> &Str,int &draw)
{
        string str,str1,str2,str3,Cut="",";
        int times=3000;

        vector<string>::iterator itS = Str.begin();
        for(;itS!=Str.end();itS++)
        {
                str = *itS;
                if(strstr(str.c_str(),"//")!=NULL)continue;
                else if(strstr(str.c_str(),"Cell[")!=NULL) //Cell の大きさを読み込む
                {
                        _strcut_(str1,str2,"[",str);
                        if(str1=="Cell")
                        {
                                str=str2;
                                _strcut_(str1,str2,"]",str);
                                if(strlen(str2.c_str())!=0)draw=atoi(str2.c_str());else draw=1;
                                str=str1;
                                _strcut_(str1,str2,",",str);
                                X=atoi(str1.c_str());
                                Y=atoi(str2.c_str());
                                cout << X << " × " << Y << " の Cell 作成" << endl;
                                MakeCell(P,T,X,Y);
                        }
                }
                else if(strstr(str.c_str(),"m(")!=NULL) //token の設定を読み込む
                {
                        _strcut_(str1,str2,"m(",str);
                        str=str2;
                        _strcut_(str1,str2,",",str);
                        unsigned int x=atoi(str1.c_str());
                        str=str2;
                        _strcut_(str1,str2,"){",str);
```

```
                        unsigned int y=atoi(str1.c_str());
                        _strcut_(str,str1,"}",str2);
                        _strcut_(str1,str2,",",str);
                        int angle=atoi(str1.c_str());
                        str=str2;
                        _strcut_(str1,str2,",",str);
                        int value=atoi(str1.c_str());
                        str=str2;
                        _strcut_(str1,str2,",",str);
                        int slow=atoi(str1.c_str());
                        int wait=atoi(str2.c_str());
                        InMark(P,X*y+x,angle,value,slow,wait);
                        cout << "Mark x:" << x << " y:"<< y << " angle:" << angle << " value:" <<
value << " slow:" << slow << " wait:" << wait<< endl;
                }
                else if(strstr(str.c_str(),"cell(")!=NULL) //セル内の処理設定を読み込む
                {

                        int calpoint=0,fu=0,n=0,han=0,v=0;

                        _strcut_(str1,str2,"cell(",str);
                        str=str2;
                        _strcut_(str1,str2,",",str);
                        unsigned int x=atoi(str1.c_str());
                        str=str2;
                        _strcut_(str1,str2,"){",str);
                        unsigned int y=atoi(str1.c_str());
                        str=str2;

                        _strcut_(str1,str3,"}",str);

                        str=str1;
                        for(int i=0;i<3;i++,str=str2)
                        {
                                _strcut_(str1,str2,",",str);
                                if(strlen(str1.c_str())==0)continue;
                                calpoint = i;
                                if((fu=IsFugo(str1))>0){n=delFugo(str1);}
                                else n=atoi(str1.c_str());
                        }
                        if((han=IsFugo(str3))>=10)
                        {
                                v=delFugo(str3);
                        }
                        InTransition(T,X*y+x,/*argm,*/calpoint,fu,n,han, v);
                        cout << "Transition x:" << x << " y:"<< y << " CalPoint:" << calpoint << "
符号:" << fu << " 値:" << n << " 判定:" << han <<" 判定値:" << v << endl;

                }
                else if(strstr(str.c_str(),"times(")!=NULL) // 何回ループするか、回数を読み込む
                {
                        _strcut_(str1,str2,"times(",str);
                        str=str2;
                        _strcut_(str1,str2,")",str);
                        times=atoi(str1.c_str());
                }
        }
        return times;
}
```

```
int outXY(unsigned int &X,unsigned int &Y,vector<Places> &P,vector<Transition>
&T,vector<string> &Str,string filename)
{
	string str;
	unsigned int x,y;
	int times=3000,i;

	ofstream fsO(filename.c_str(),ios::out);
	if(!fsO){cerr < "Error in open out!¥n"; exit(1);}

	fsO << "Cell[" << X << "," << Y << "]" << "¥n" ;

	vector<Transition>::iterator itT = T.begin();
	for(;itT!=T.end();itT++)
	{

	int CalPoint;   // -1:計算しない 0:angle 1:value 2:slow 3:wait
	int fugo;    // +-*/^√
	int num;        // 数値
	int hantei;  // > = < の大小判定記号が入る
	int V;          // 判定数値

		if((*itT).CalPoint==-1)continue;

		y=(*itT).ID_ / X;
		x=(*itT).ID_ -X*y;

	fsO << "cell(" << x<< "," <<y<<"){";
	for(i=0;i<(*itT).CalPoint;i++)fsO << ",";
	fsO << FugoNum((*itT).fugo) << (*itT).num ;
	for(;i<4;i++)fsO << ",";
	fsO << "}";
	if((*itT).hantei!=0){fsO << FugoNum((*itT).hantei) << (*itT).V ;}
	fsO <<"¥n";
	}

	vector<Places>::iterator itP = P.begin();
	for(;itP!=P.end();itP++)
	{
		if((*itP).isMark==0)continue;
		vector<Mark> &vM = (*itP)._mark;   //Place の中の mark
		vector<Mark>::iterator itM = vM.begin(),itMmp;

		y=(*itP)._ID / X;
		x=(*itP)._ID -X*y;
		for(unsigned int k=0;itM != vM.end();itM++,k++)
		{//一つの PlacesCell の中に入っている Mark 群について処理する
			Mark &mrk=(*itM);
			fsO << "m(" << x<< "," <<y<<"){"
<<mrk.angle<<"," <<mrk.value<<"," <<mrk.slow<<"," <<mrk.wait<<"}¥n";
		}
	}
	return times;
}

int main(int argc,char *argv[])
{
	unsigned int  x=200,y=200;
	int times,draw;
	unsigned long color=256*256*256-1;// 白色 RGB(255,255,255)=16777215 と同じ
```

```
	for(int i=0;i<argc;i++)
	{
		vector<Places> XYp;
		vector<Transition> XYt;
		vector<string> Str,Strout;

		if(argc==1)file_read(Str,"Petri.dat");
		else file_read(Str,argv[i+1]);
		times=setXY(x,y,XYp,XYt,Str,draw);

		screen_int(x,y);
		cout << times << "回繰り返し行います。" << endl;
		for(int i=0;i<times;i++)
			transition(XYp,XYt,x,y,draw);

		outXY(x,y,XYp,XYt,Strout,"out.dat");

		cout << "終了しました! [Ctrl+C] を押して下さい" ;
		gend();//	hitanykey();
	}
	return EXIT_SUCCESS;
}


//============================fread.cpp=======================================
#include <string>
#include <vector>
#include <fstream.h>
#include <iostream.h>
#include <iterator>
#include <algorithm>

int  file_read(vector<string> &Str,char *filename)
{
	string pt1;
	char buff[12000];

	FILE *fp;
	if((fp=fopen(filename,"r"))==NULL)return 1;

	while(fgets(buff,sizeof(buff),fp))
	{
		char *p;
		p=strstr(buff,"¥n");
		*p='¥0';
		pt1 = buff;
		Str.push_back(pt1);
	}

	return 0;
}


//============================strcut.cpp=======================================
#include <iostream>
#include <string>
using namespace std;
const int BUFF_SIZE = 256;

string strcut(string str,char ch[])
```

```
{
	int i;
	i =(int) str.length()+1;
	char *const cha = new char[i];
	char *p;
	string st;st="";

	strcpy(cha,str.c_str());
	if((p=strstr(cha,ch))==NULL){delete(cha);return st;}
	i=strlen(ch);
	st=p+i;

	delete(cha);
	return st;
}

string strcut(string str,string st)
{
	char *p;
	int i;
	i =(int) str.length()+1;
	char *const cha = new char[i];
	i =(int) st.length()+1;
	char *const ch = new char[i];
	string st1;st1="";

	strcpy(cha,str.c_str());
	strcpy( ch, st.c_str());
	if((p=strstr(cha,ch))==NULL){delete(cha);delete(ch);return st1;}
	i = strlen(ch);
	st1 = p+i;

	delete(cha);delete(ch);
	return st1;
}

string _strcut(string str,char ch[])
{
	int i;
	i =(int) str.length()+1;
	char *const cha = new char[i];
	char *p;
	string st;st="";

	strcpy(cha,str.c_str());
	if((p=strstr(cha,ch))==NULL){delete(cha);return st;}
	*p=NULL;
	st=cha;

	delete(cha);
	return st;
}

string _strcut(string str,string st)
{
	int i;
	i =(int) str.length()+1;
	char *const cha = new char[i];
	i =(int) st.length()+1;
	char *const ch = new char[i];
```

```
	char *p;
	string st1;st1="";

	strcpy(cha,str.c_str());
	strcpy(ch,st.c_str());
	if((p=strstr(cha,ch))==NULL){delete(cha);delete(ch);return st1;}
	*p=NULL;
	st1=cha;

	delete(cha);delete(ch);
	return st1;
}

int _strcut_(string &pre,string &bot,string cut,string str)
{
	pre=_strcut(str,cut);
	bot=strcut(str,cut);
	int i=pre.length(),j=bot.length(),k=str.length();
	if(i==0 && j==0)return k;
	else	return i;

}

void strdev(vector<string> &Str1,string cut,vector<string> Str)
{
	string word,st;
	vector<string>::iterator itS = Str.begin();

	for(string st;itS != Str.end();itS++)
	{
		st=*itS;
//		st=_strcut(st,"¥n");
		string bot,end;
		for(;;st=bot)
		{
			end=_strcut_(word,bot,cut,st);
			if(end==st){Str1.push_back(end);break;}
			Str1.push_back(word);
		}
	}
}

void strdev(vector<string> &Str1,vector<string> Cut,vector<string> Str)
{
	int id;//=0; //ID 番号
	string word,st,cut;
	vector<string>::iterator itS = Str.begin();
	vector<string>::iterator itC;// = Cut.begin();
	vector<string>::iterator itKey=Cut.begin();


	for(string st;itS != Str.end();itS++)
	{//改行コードまでの文字列を切り出す
		st=*itS;
		string bot,end;
		string tmp= st;
		for(; st.length()!=0 ;tmp=st=bot,itC=itKey = Cut.begin())
		{//文字列から単語を切り出す

//切り出し文字を探す
```

```
                        int len=tmp.length(),len2;
                        for(itC=Cut.begin();itC != Cut.end();itC++)
                        {//最も最適な切り出す文字を探す
                                len2=_strcut_(word,bot,*itC,tmp);
                                if(len2 == 0)goto Bottom;
                                if(len>len2 /* && len2 !=0 */)//一番文字列が短く、0 でないものに
分けられる key を登録する
                                {
                                        len=len2;
                                        itKey=itC;
                                }
                        }
//-------------------------------------------------------------------------------
                        if(_strcut_(word,bot,*itKey,st)==0)//登録されている key で分ける
                        {//key が無い場合
                                Str1.push_back(st);
                                break;
                        }
                        Str1.push_back(word);
                        if(*itKey != " ")
                        {
                                Str1.push_back(*itKey);
                        }
Bottom:
                }
        }
}
```