# 1　スーパーバイザの計算 (CASE1_METHOD2_PRESEN3.m)

プログラム 1: Create G1

```
% Creat G1 (DES)
robot(1).Q = 72; % number of states
       % the initial state q0 is always labeled "0"
robot(1).Qm = (71); % marker state set
robot(1).delta = [0,15,6;
                  ...
                  64,15,71]; % transition triples (exit state, event, enter
                      state)

create('G1', robot(1).Q, robot(1).delta, robot(1).Qm); % create G1
```

オートマトン G1 を作成．G2，H1 も同じように．

プログラム 2: Create object

```
object.Q=2;
object.Qm=(1);
object.delta=[0,108,1];
create('Eobject',object.Q,object.delta,object.Qm);
```

落ちた商品のオートマトンを作成.
行動 108（人間が落ちた商品を棚に戻す行動）をするとき状態が遷移.

```
edge=create_edge(human.state);
statepairsH = create_mutex_pairsH(human.state,edge);
```

## 1.1　(create_edge.m)

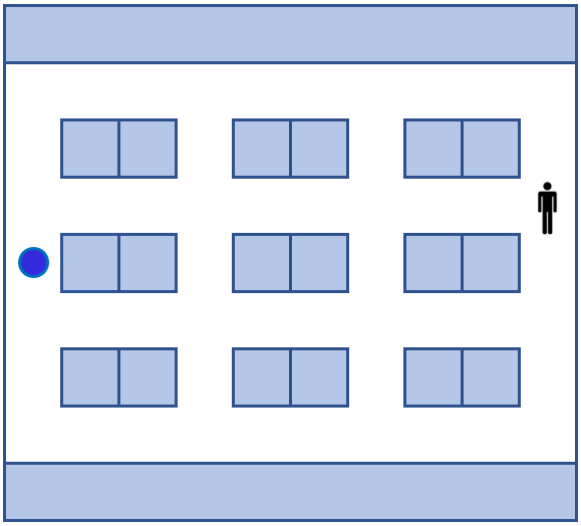
```
function edge=create_edge(human_states)
    for i=1:length(human_states)
        edge(i)=0;

        % スタート地点とゴール地点の判定
        if floor((human_states(i,1)-1)/10)==0 % 北（上）がスタート地点
            edge(i)=edge(i)+1000;
        end
        if mod(human_states(i,1),10)==0 % 東（右）が壁
```

```
10              edge(i)=edge(i)+100;
11          end
12          if floor((human_states(i,1)-1)/10)==6 % 南（下）がゴール地点
13              edge(i)=edge(i)+10;
14          end
15          if mod(human_states(i,1),10)==1 % 西（左）が壁
16              edge(i)=edge(i)+1;
17          end
18      end
19  end
```

H1 と G1，G2 の mutex するペアを定義するときに必要な関数.
H1 の上下左右にロボットがいてはいけないが，H1 の左右に壁がある場合と上下にスタートかゴールある場合は，mutex するペアを減らす必要がある．ここで判定をして，human.state（人間が通る可能性のある位置をピックアップした配列）に対応した，edge 配列を作成.

## 1.2 (create_mutex_pairsH.m)

statepairsH(ロボットと人間が同じ状態にいるペア) に，人間の周りのロボットがいてはいけない場所を追加していく関数.
edge 配列を使って，図のように H1 が 30 に，G1 が 31 に存在するときのような，数字は隣り合っているが実際は隣り合っていないペアを除きつつ，statepairsH に追加していく.



プログラム 4: specifications

```
120 %Specifications
121 mutex('E12', 'G1', 'G2', statepairs);
122 mutex('E1H', 'G1', 'H1', statepairsH);
123 mutex('E2H', 'G2', 'H1', statepairsH);
124 mutex('E1object','G1','Eobject',statepair_object);
125 mutex('E2object','G2','Eobject',statepair_object);
126
127 sync('E', 'E12', 'E1H', 'E2H', 'E1object', 'E2object');
```

101 103 行はロボット同士，ロボットと人間の衝突を回避する specifications.
104，105 行は商品が落ちているとき，その落ちている場所をロボットが通らないように制限する specifications.

プログラム 5: SUP

```
129 allevents('ALLPLANT', 'PLANT');
130 sync('SPEC', 'E', 'ALLPLANT');
131
132 supcon('SUP', 'PLANT', 'SPEC');
133
134 condat('SUPD', 'PLANT', 'SUP');
135
136 printdes('SUP');
137 printdat('SUPD');
```

通常通り，supervisor を計算する．

# 2　シミュレーション (presen3.m)

draw_human_area：ロボットの進入禁止エリアをプロットする
draw_agent：ロボットと人間をプロットする
draw_item：商品をプロットする
draw_goal：ゴール（搬出場所）をプロットする
draw_trouble：商品が落ちた場所をプロットする
agent：現時点のロボットと人間の現在地，スタートとゴールや商品の位置，行動を管理
path_string：初期状態から受理状態までの一連の行動（txt ファイルから）
mutex_statepairsH：mutex のペア（txt ファイルから）
human_area_state：現時点のロボットの進入禁止エリア

```
1  x = 10;
2  y = 9;
3  shelf = [1,2;
4           2,2;
5           4,2;
6           5,2;
7           7,2;
8           8,2;
9           1,4;
10          2,4;
11          4,4;
12          5,4;
13          7,4;
14          8,4;
15          1,6;
16          2,6;
17          4,6;
18          5,6;
19          7,6;
20          8,6];
21
22      for i = 1:x
23          plot([i,i], [1,y-1], 'k');
24          hold on;
25      end
26
27      for i = 0:y
28          plot([0,x], [i,i], 'k');
29          hold on;
30      end
31
32      for i = 1:length(shelf)
33          rectangle('Position', [shelf(i,1),shelf(i,2),1,1], 'EdgeColor', 'r', '
                LineWidth', 3);
34          hold on;
35      end
```

1〜63 行はデフォルト.
x が横のサイズ，y が縦のサイズ.
shelf は棚を表示する位置.
85〜98 行でグラフに倉庫のみの様子をプロットする.

```
167 draw_agent(1) = plot(agent(1).statex, agent(1).statey, 'o', 'MarkerSize', 18,
        'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b');
168 draw_agent(2) = plot(agent(2).statex, agent(2).statey, 'o', 'MarkerSize', 18,
        'MarkerEdgeColor', [0 0.9 0], 'MarkerFaceColor',[0 0.9 0]);
169 draw_agent(3) = plot(agent(3).statex, agent(3).statey, 'd', 'MarkerSize', 14,
```

```
        'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k');
170
171 draw_item(1) = plot(agent(1).itemx, agent(1).itemy, 's', 'MarkerSize', 16, '
        MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b');
172 draw_item(2) = plot(agent(2).itemx, agent(2).itemy, 's', 'MarkerSize', 16, '
        MarkerEdgeColor', [0 0.9 0], 'MarkerFaceColor', [0 0.9 0]);
173
174 draw_goal(1) = plot(agent(1).goalx, agent(1).goaly, 'o', 'MarkerSize', 18, '
        MarkerEdgeColor', 'b', 'LineWidth', 2);
175 draw_goal(2) = plot(agent(2).goalx, agent(2).goaly, 'o', 'MarkerSize', 18, '
        MarkerEdgeColor', [0 0.9 0], 'LineWidth', 2);
176 draw_goal(3) = plot(agent(3).goalx, agent(3).goaly, 'd', 'MarkerSize', 14, '
        MarkerEdgeColor', 'k', 'LineWidth', 2);
177
178 draw_trouble = plot(trouble.statex, trouble.statey, 'x', 'MarkerSize', 18, '
        MarkerEdgeColor', [0.5 0 1], 'MarkerFaceColor', [0.5 0 1], 'LineWidth', 4)
        ; % [0.5 0 1] = purple
```

セットしたいろいろをグラフにプロットする.

ここからロボットと人間が 1 マス進む過程について

```
208         human_state=(7.5-agent(robot_num+1).statey)*10+agent(robot_num+1).statex
                +0.5; % Agent(robot_num+1)は人
                間
209         human_area_state=search_human_area(mutex_statepairsH,human_state); % はロ
                ボットの進入禁止エリア Human_area
210         if length(human_area_state)~=1
211             for i=2:length(human_area_state)
212                 human.area(i,1)=mod(human_area_state(i)-1,10)+0.5;
213                 human.area(i,2)=7.5-floor((human_area_state(i)-1)/10);
214                 draw_human_area(i)=scatter(human.area(i,1),human.area(i,2),500,'
                        s','MarkerFaceColor','r','MarkerEdgeColor','r','
                        MarkerFaceAlpha',.4,'MarkerEdgeAlpha',.4);
215             end
216         end
```

ロボットの進入禁止エリアを探索して保存する.

```
219         for a=1:agent_num % すべての agent・().をリセット act
220             agent(a).act=-1;
221         end
```

すべてのエージェントの行動を，前回の行動が残らないよう-1 にリセットする.

```
223         for a=1:agent_num % すべてのエージェントのを決定 act
224             if count>max
225                 break;
226             end
227             A=floor(path_string(count,1)/10);
228             if A>robot_num
229                 A=A-9+robot_num;
230             end
231             if agent(A).act==-1
232                 agent(A).act=mod(path_string(count,1), 10); % 各エージェントの行
                        動を代入
233                 count=count+1;
234             else
235                 break;
236             end
237         end
```

Event がどのエージェントの行動か見極めて，そのエージェントの行動がリセットされた時の-1 のままなら，event を agent().act に代入.
もし-1 以外であれば，エージェントが 1 回行動する時間内に，2 回行動することになってしまうので break.

```
239         % マス分動かす 1
240         for i=0:n:1
241             for a=1:agent_num
242                 switch agent(a).act
243                     case 1
244                         set(draw_agent(a), 'YData', (agent(a).statey+i));
245                     case 3
246                         set(draw_agent(a), 'XData', (agent(a).statex+i));
247                     case 5
248                         set(draw_agent(a), 'YData', (agent(a).statey-i));
249                     case 7
250                         set(draw_agent(a), 'XData', (agent(a).statex-i));
251                     otherwise
252                 end
253             end
254
255             % も移動 human_area
256             if length(human_area_state)~=1
257                 switch agent(robot_num+1).act
258                     case 1
259                         for j=2:length(human_area_state)
260                             set(draw_human_area(j), 'YData', human.area(j,2)+i);
```

```
261                         end
262                     case 3
263                         for j=2:length(human_area_state)
264                             set(draw_human_area(j), 'XData', human.area(j,1)+i);
265                         end
266                     case 5
267                         for j=2:length(human_area_state)
268                             set(draw_human_area(j), 'YData', human.area(j,2)-i);
269                         end
270                     case 7
271                         for j=2:length(human_area_state)
272                             set(draw_human_area(j), 'XData', human.area(j,1)-i);
273                         end
274                     otherwise
275                 end
276             end
277             drawnow;
278         end
```

1 マス分の移動を 1/n 回に分けてプロットし，動画のように表示する．

```
291         for a = 1:agent_num
292             switch agent(a).act
293                 case 1
294                     agent(a).statey = agent(a).statey+1;
295                 case 3
296                     agent(a).statex = agent(a).statex+1;
297                 case 5
298                     agent(a).statey = agent(a).statey-1;
299                 case 7
300                     agent(a).statex = agent(a).statex-1;
301                 case 8
302                     delete(draw_trouble);
303                 otherwise
304             end
305             if agent(a).statex==agent(a).itemx & agent(a).statey==agent(a).itemy
306                 delete(draw_item(a));
307             end
308         end
```

エージェントの現在地を更新．また，ロボットが商品の場所に到達したら，商品の表示を消す．※目的の商品がある位置でロボットがとまったとき，2 回目の delete() が実行できずにエラーなるから，直した方がいいかも．

```
317         if length(human_area_state)~=1
318             for i=2:length(human_area_state)
319                 delete(draw_human_area(i));
320             end
321         end
```

human_area_state のサイズが 1 のとき（進入禁止エリアがないとき）はスルー.
進入禁止エリアがあるなら，プロットを delete.