# 世界モデル実習(Sr)/人間の安全保障実験実習 III(Gr) 2006 年度夏学期

## 第四回 エージェントに周囲を調べさせる

前回の復習

・ If 文 (if　else　elseif　endif)
・ ツリーで変数を追加(整数型、実数型)
・ 課題の解説 ① ホシが徐々に下に向かう　② 正方形、三角形の軌跡を描く

今回の内容

・ ツリーの変数と一時的な変数
・ エージェント集合型変数
・ 周囲の調べかた

### ツリーの変数と一時的な変数

前回、ツリーで変数を追加する方法を紹介しました。この変数に値を入れておくことで、次のステップにその値を持ち越すことができました。その必要がない場合には、ルールエディタの中で変数を一時的に作ることができます。たとえば、整数型の変数 i を作りたいのであれば、{ とルールの間に、dim i as integer という風に変数を宣言します。

```
Agt_Step{
dim a as double

a = rnd()
if a < 0.25 then
        forward(1)
elseif a < 0.5 then
        forward(-1)
elseif a < 0.75 then
        turn(90)
else
        turn(-90)
end if
}
```

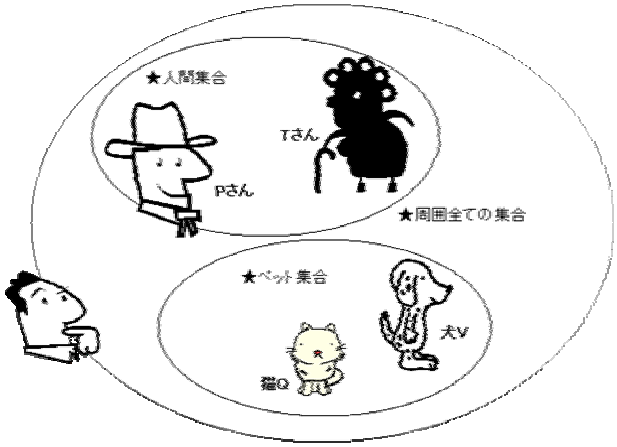
```
Agt_Step{


if rnd() < 0.25     then
        forward(1)
elseif rnd() < 0.5 then
        forward(-1)
elseif rnd() < 0.75 then
        turn(90)
else
        turn(-90)
end if
}
```

こんな風にも使えます。右は失敗してしまった例。変数の宣言は、integer(整数型)、double(実数型)、Agt(エージェント型)、Agtset(エージェント集合型)といった正式な名前を使います。詳しくは、前回のレジュメの表を参照。

**エージェント集合型変数**

周囲にいる他のエージェントを認識したいときなどには、エージェント集合型変数を使います。周囲を認識するということは、下の図で言えば、自分の周囲に「Ｐさん、Ｑ、Ｔさん、Ｖ」がいると判ることです。したがって、周囲にいるエージェントを認識したことを表す変数のとる「値」は、数値ではなく、{Ｐ、Ｔ}、{Ｑ、Ｖ}、{Ｐ、Ｑ、Ｔ、Ｖ}といったエージェントの集合になります。このように、エージェントたち(エージェントの集合)を「値」として格納しておく入れ物をエージェント集合型変数と呼びます。ツリーから作る事もできますし、ルール中で dim syuugou as agtset などと宣言して使う事もできます。




**周りを観察して、変数に集合を格納する**

入れ物は揃いました。つぎに、集合を作ってその中に入れる方法を説明します。これには関数を使います。ただし、「周囲を観察する」ために、「何を」「どこまで」みるかを指定する必要があります。例としては、こんな関数があります。

MakeOneAgtsetAroundOwn(エージェント集合型変数, 範囲, エージェント種, 自分自身を含むか否か)

自分の周りの一種類のエージェントのリストを作る関数です。これをルール内で

MakeOneAgtsetAroundOwn(syuugou, 1, universe.oozora.tori, false)

の様に書くと、距離 1 以内にいる、自分以外の tori を、syuugou の中にリストアップします。このタイプの関数は沢山あります。一部を紹介すると･･･

MakeAgtset(エージェント集合型変数, エージェント種)

→ 指定した種類のエージェント全てをリスト

MakeAgtsetSpace(エージェント集合型変数, 空間名)

→ 指定した空間上のエージェント全てをリスト

MakeAllAgtsetAroundOwn(エージェント集合型変数, 視野, 自分自身を含むか否か)

→ 自分の周囲(視野範囲内)にいるエージェント全てをリスト

MakeOneAgtsetAroundPosition(エージェント集合型変数, 空間, X, Y, Layer,視野,エージェント種)

→ 指定した座標の周囲にいる指定エージェント種をリスト

詳しくは、ヘルプを参照してください。

**エージェント集合を利用する(数を数える)**

リストアップしても、それをどのように行動に結びつけたら良いのでしょうか? artisoc では、集合に含まれる相手の状態(変数)を読んだり、変えたり、抹殺することさえできます。が、ここでは要素の数を数える事からはじめましょう。CountAgtset(エージェント集合型変数)を使います。たとえば tori モデルで、

```
dim kazu as integer
dim syuugou as agtset

MakeOneAgtsetAroundOwn(syuugou, 1, universe.oozora.tori, false)
kazu = CountAgtset(syuugou)
```

と書くと、syuugou の中に、半径 1 以内にいる、自分以外の tori を数えて、kazu の中に入れる事ができます。あとは、「if kazu >= 3 then」のように if 文を用いて、付近の状況を見た行動を記述できます。

**例題**

渋谷駅のコンコースで、山手線から井の頭線に乗る人と、井の頭線から山手線に行く人の流れをシミュレートしてみましょう。50×50 のループした空間に、二種類のエージェント(eastward、westward)を 100 ずつ作ってください。初めは両方ともランダムな場所にいます。eastward は 0、westward は 180 を向いています。毎ステップ、1 ずつ前に進みます。ただし、もし周囲 1 に対向客が居たら、20 度右に進み、進み終わったらまた元の方向を向きます。

eastward のルール

```
Agt_Init{
my.X = rnd()*50
```

```
my.Y = rnd()*50
my.Direction = 0
}
Agt_Step{
dim syuui as agtset
dim a as integer

MakeOneAgtsetAroundOwn(syuui, 1, universe.concourse.westward, false)
a = countagtset(syuui)

if a > 0 then
        turn(-20)
end if
forward(1)
my.Direction = 0
}
```

課題①

例題では同じ方向に行く人は無視していました。1 以内に同じ方向を向いた人がいたら、0.5 ずつ進むようにしてみましょう。

課題②

ここまで作ったモデルでは、自分のいる場所を中心として、半径 1 の範囲を見ていました。これでは後ろも見ている事になります。少し不自然なので、前方 1 から半径 1 の範囲を見るようにしてみましょう。

ヒント: 今まで　　見る → 数える → 判断する → 進む

　　今回　　一度進む → 見る → 数える → 戻る → 判断する → 進む