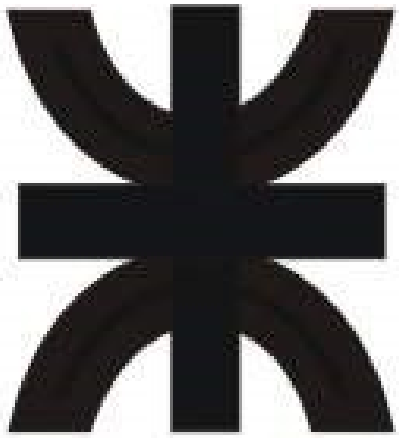
Inteligencia Artificial



Trabajo Práctico N°1

Grupo 4
Integrantes:

- ***Calgaro, María Gabriela***
  <nn_gabita@hotmail.com>
- ***Cravero, Darío Javier***
  <dario.cravero@gmail.com>
- ***Salis, Rafael***
  <rafaelsalis@gmail.com>

2008

Índice de contenido

Etapa 1: Definición del agente-víbora

Objetivo

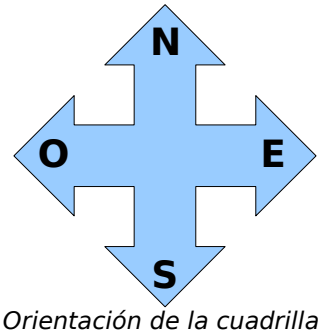
El objetivo del Agente es sobrevivir en el mundo en cual se desenvuelve. Para ello, la idea es que conozca todo el mundo, es decir, que encuentre los cuatro límites del mismo dejando todo el mundo sin valores desconocidos, coma todo el alimento que encuentra y aún permanezca con vida.

Representación de Estado

(composiciónDelAgente, orientaciónDeLaCabeza, mundoConocido, longitud, estaVivo, celdasVisitadas)

composiciónDelAgente: es una lista ordenada con pares de coordenadas (x,y) que indica como está compuesto el Agente, es decir, que casilleros en la grilla conforman su cuerpo. En esta representación, el primer elemento es la cabeza de la víbora y el último la cola.

orientaciónDeLaCabeza: es un valor que indica hacia donde se dirige la cabeza de la víbora. Los posibles valores son: Norte, Sur, Este, Oeste.



Dibujo 1: Orientación de la cuadrilla

mundoConocido: es una matriz que representa lo que el Agente conoce como Mundo, a medida que se va desplazando por el mismo. Los posibles valores son: 0 (no hay nada), 2 (comida), 3 (parte de la víbora), * (no explorado).

longitud: es el tamaño del Agente, es decir, la cantidad de casilleros de la grilla que ocupa el mismo.

estaVivo: variable que indica el estado de la víbora. Los posibles valores son: true (está viva), false (no está viva).

celdasVisitadas: es una lista ordenada con pares de coordenadas (x,y) que indican las celdas del mundo por las que ya pasó el agente, es decir, que casilleros ha visitado desde el comienzo del juego.

Estado Inicial

```
(     {(1,1)},
      Este,
       *     0     *
   [ 0       3     0 ]
       *     0     *
      1,
```

```
      true
      {}          )
```

Estado Final

```
(     -,
      -,
       0 ó 3      0 ó 3      …      0 ó 3
       0 ó 3      0 ó 3      …      0 ó 3
     [ 0 ó 3      0 ó 3      …      0 ó 3 ]
        …          …          …        …
       (0 ó 3)    0 ó 3      …      0 ó 3
      -,
      true
       -                  )
```

Nota: en la matriz que representa el estado final del mundo conocido, se explicita que en cada posición de la misma puede tener dos valores posibles, 0 o 3. Se aclara que en el caso en el cual las posiciones tienen el valor 3, las mismas deben estar contiguas.

Prueba de Meta

Si (noHayMasComida &&
estaTodoConocido &&
estaVivo) **=> Éxito**

Funciones Auxiliares

noHayMasComida:

recorre la matriz en busca de comida (2)

si encuentra comida

=> *false*

si recorre toda la matriz sin encontrar comida

=> *true*

estaTodoConocido:

toma la matriz que representa al mundo conocido y la recorre en busca algo desconocido (*)

si encuentra algún valor desconocido

=> *false*

si no encuentra ningún valor desconocido

=> *true*

estaVivo:

Si *(estaVivo = true)*

=> *true*

sino

=> *false*

Percepciones

El entorno le brinda información al Agente de lo que hay en un par coordenado, siempre que este sea adyacente y este arriba, abajo, a la izquierda o su derecha.

La información que puede retornar es:

0 si no hay nada,

1 si hay un límite del mundo,

2 si hay comida,

3 si hay parte del cuerpo del Agente.

Operadores

GirarIzquierda:

Si Orientacion = NORTE **=>** Orientación = OESTE

Sino Si Orientacion = OESTE **=>** Orientación = SUR

Sino Si Orientacion = SUR **=>** Orientación = ESTE

Sino Si Orientacion = ESTE **=>** Orientación = NORTE

GirarDerecha:

Si Orientacion = NORTE **=>** Orientación = ESTE

Sino Si Orientacion = OESTE **=>** Orientación = NORTE

Sino Si Orientacion = SUR **=>** Orientación = OESTE

Sino Si Orientacion = ESTE **=>** Orientación = SUR

Avanzar:

Si Orientacion = NORTE &
!HayObstaculo(x-1,y) &
!HayComida(x-1,y) &
!PasoNVecesPorEstaCelda(x,y)

=> ComposicionDelAgente.Add(x-1,y)
PuntoABorrar = ComposicionDelAgente.RemoveLast()

MundoConocido.Actualizar(x-1,y,PuntoABorrar)
CeldasVisitadas.Actualizar(PuntoABorrar)

Sino Si Orientacion = SUR &
!HayObstaculo(x+1,y) &
!HayComida(x+1,y) &
!PasoNVecesPorEstaCelda(x,y)
=> ComposicionDelAgente.Add(x+1,y)
PuntoABorrar = ComposicionDelAgente.RemoveLast()
MundoConocido.Actualizar(x+1,y,PuntoABorrar)
CeldasVisitadas.Actualizar(PuntoABorrar)

Sino Si Orientacion = OESTE &
!HayObstaculo(x,y-1) &
!HayComida(x,y-1) &
!PasoNVecesPorEstaCelda(x,y)
=> ComposicionDelAgente.Add(x,y-1)
PuntoABorrar = ComposicionDelAgente.RemoveLast()
MundoConocido.Actualizar(x,y+1,PuntoABorrar)
CeldasVisitadas.Actualizar(PuntoABorrar)

Sino Si Orientacion = ESTE &
!HayObstaculo(x,y+1) &
!HayComida(x,y+1) &
!PasoNVecesPorEstaCelda(x,y)
=> ComposicionDelAgente.Add(x,y+1)
PuntoABorrar = ComposicionDelAgente.RemoveLast()
MundoConocido.Actualizar(x,y+1,PuntoABorrar)
CeldasVisitadas.Actualizar(PuntoABorrar)

<u>Comer:</u>

Si Orientacion = NORTE &
HayComida(x-1,y)
=> ComposicionDelAgente.Add(x-1,y)
MundoConocido.Actualizar(x-1,y)
Longitud += 1

Sino Si Orientacion = SUR &
HayComida(x+1,y)
=> ComposicionDelAgente.Add(x+1,y)

MundoConocido.Actualizar(x+1,y)
Longitud += 1

Sino Si Orientacion = OESTE &
HayComida(x1,y-1)
=> ComposicionDelAgente.Add(x,y-1)
MundoConocido.Actualizar(x,y-1)
Longitud += 1

Sino Si Orientacion = ESTE &
HayComida(x,y+1)
=> ComposicionDelAgente.Add(x,y+1)
MundoConocido.Actualizar(x,y+1)
Longitud += 1

Funciones Auxiliares

MundoConocido.Actualizar(posX, posY): Dado que este procedimiento tiene lugar solamente si no hay obstáculos en el Punto (posX, posY), se marca con un 3 (es decir, que hay parte del Agente en él) la posición. Esta función es llamada por el operador comer ya que luego de avanzar, el agente no debe expandirse debido a que pudo comer, incrementando en 1 su longitud.

MundoConocido.Actualizar(posX, posY, PuntoABorrar): Dado que este procedimiento tiene lugar solamente si no hay obstáculos en el Punto (posX, posY), se marca con un 3 (es decir, que hay parte del Agente en él) la posición y además se indica con un 0 (porque ya no hay nada) en el PuntoABorrar.

ComposicionDelAgente.Add(posX, posY): Agrega un par coordenado (posX, posY) en la primera posición de la lista ComposicionDelAgente.

ComposicionDelAgente.RemoveLast(): Quita el par coordenado al final de la lista ComposicionDelAgente y lo retorna.

CeldasVisitadas.Actualizar(posX, posY): Agrega el par coordenado (posX, posY) a la lista de CeldasVisitadas de manera de poder indicar que el agente ya ha pasado por esa celda en el mundo.

HayComida(posX, posY): retorna *true* en caso de que el valor sea 2, *false* en caso contrario.

HayObstaculo(posX, posY): retorna *true* en caso de que el valor sea 1, 2, o 3, *false* en caso contrario.

PasoNVecesPorEstaCelda(posX, posY): Recorre la lista de CeldasVisitadas de forma tal que si encuentra el par coordenado (posX, posY) 2 veces o más, retorna *true*, en caso contrario retorna *false*.

Descripción de la estrategia de búsqueda

La estrategia de búsqueda implementada fue el Método De Búsqueda Horizontal (o Amplitud) .

El método que sigue esta estrategia es el siguiente:

> Teniendo en cuenta los distintos operadores (comer, avanzar, girar a la izquierda, girar a la derecha) va formando el árbol de búsqueda que utilizará la siguiente acción.
>
> Toma el primer nodo del árbol, evalúa todas las posibilidades en un segundo nivel, y repite esta acción ordenadamente en cada nodo del primer nivel. Esto se efectúa recursivamente hasta que encuentre un camino que lo lleve al éxito, o bien, falle en encontrar una solución.

En caso de que en algún momento deje de ramificar, se dice que el método es completo.