Proyecto de Simulación y Programación Declarativa Agentes

Adrian Hernández Pérez

C-411

correo: adrianmatcom@gmail.com.

código: https://github.com/AdrianHP/SimuProDec-Agentes

Ambiente

El ambiente en el cual intervienen los agentes es discreto y tiene la forma de un rectángulo de N × M. El ambiente es de información completa, por tanto todos los agentes conocen toda la información sobre el agente. El ambiente puede variar aleatoriamente cada t unidades de tiempo. El valor de t es conocido.

Las acciones que realizan los agentes ocurren por turnos. En un turno, los agentes realizan sus acciones, una sola por cada agente, y modifican el medio sin que este varíe a no ser que cambie por una acción de los agentes. En el siguiente, el ambiente puede variar.Si es el momento de cambio del ambiente, ocurre primero el cambio natural del ambiente y luego la variación aleatoria. En una unidad de tiempo ocurren el turno del agente y el turno de cambio del ambiente.

Los elementos que pueden existir en el ambiente son obstáculos, suciedad, niños, el corral y los agentes que son llamados Robots de Casa. A continuación se precisan las características de los elementos del ambiente:

Obstáculos: estos ocupan una única casilla en el ambiente. Ellos pueden ser movidos, empujándolos, por los niños, una única casilla. El Robot de Casa sin embargo no puede moverlo. No pueden ser movidos ninguna de las casillas ocupadas por cualquier otro elemento del ambiente.

Suciedad: la suciedad es por cada casilla del ambiente. Solo puede aparecer en casillas que previamente estuvieron vacías. Esta, o aparece en el estado inicial o es creada por los niños.

Corral: el corral ocupa casillas adyacentes en número igual al del total de niños presentes en el ambiente. El corral no puede moverse. En una casilla del corral solo puede coexistir un niño. En una casilla del corral, que est´e vacía, puede entrar un robot. En una misma casilla del corral pueden coexistir un niño y un robot solo si el robot lo carga, o si acaba de dejar al niño.

Niño: los niños ocupan solo una casilla. Ellos en el turno del ambiente se mueven, si es posible (si la casilla no est´a ocupada: no tiene suciedad, no est´a el corral, no hay un Robot de Casa), y aleatoriamente (puede que no ocurra movimiento), a una de las casilla adyacentes. Si esa casilla está ocupada por un obstáculo este es empujado por el niño, si en la dirección hay más de un obstáculo, entonces se desplazan todos. Si el obstáculo está en una posición donde no puede ser empujado y el niño lo intenta, entonces el obstáculo no se mueve y el niño ocupa la misma posición. Los niños son

los responsables de que aparezca suciedad. Si en una cuadrícula de 3 por 3 hay un solo niño, entonces, luego de que él se mueva aleatoriamente, una de las casillas de la cuadrícula anterior que esté vacía puede haber sido ensuciada. Si hay dos niños se pueden ensuciar hasta 3. Si hay tres niños o más pueden resultar sucias hasta 6. Los niños cuando están en una casilla del corral, ni se mueven ni ensucian. Si un niño es capturado por un Robot de Casa tampoco se mueve ni ensucia.

Modelos de Agentes

Agente Niño

Este agente se comporta de manera aleatoria.En cada turno decide si moverse o no,en caso de que se mueva, escoge al azar una de las casillas adyacentes a las que se pueda mover y se mueve.Si un niño esta en un corral o es cargado por un robot no hace nada

Agente Robot de la Casa

El Robot de Casa se encarga de limpiar y de controlar a los niños. El Robot se mueve a una de las casillas adyacentes, las que decida. Solo se mueve una casilla sino carga un niño. Si carga un niño puede moverse hasta dos casillas consecutivas. También puede realizar las acciones de limpiar y cargar niños. Si se mueve a una casilla con suciedad, en el próximo turno puede decidir limpiar o moverse. Si se mueve a una casilla donde está un niño, inmediatamente lo carga. En ese momento, coexisten en la casilla Robot y niño. Si se mueve a una casilla del corral que está vacía, y carga un niño, puede decidir si lo deja esta casilla o se sigue moviendo. El Robot puede dejar al niño que carga en cualquier casilla. En ese momento cesa el movimiento del Robot en el turno, y coexisten hasta el próximo turno, en la misma casilla, Robot y

Características

Para considerar un agente como inteligente, debemos demostrar que posee flexibilidad para la interpretación y resolución de los problemas del ambiente. En el caso de este proyecto, se programa un agente inteligente con tres variaciones en su modelo estratégico.

Reactivo: En cada uno de sus turnos, el robot hace una evaluación exhaustiva del ambiente. Si se han generado cambios, este los registrará y actuará en consecuencia. Toma sus deciciones de una manera intelegente en el estado actual del ambiente

Proactivo: El objetivo del agente es mantener la limpieza de la casa y ubicar a los niños en el corral.

Modelos

Se definieron 3 modelos para el robot con distintas estrategias para lograr su objetivo.El robot tiene dos estados internos que son

1. A = Está libre
2. B = Está cargando un niño

Modelo 1

El objetivo de este es priorizar la limpieza siempre está limpiando o localizando la suciedad más cercana en el ambiente en el estado actual,he intenta moverse hacia la casilla que más lo acerque,aunque si por el camino se encuentra un niño lo toma.

Las reglas basicas de comportamiento de este modelo en orden de priodad son :

Para el estado A

1- Si se encuentra en una casilla sucia entonce limpia

2- Si hay un niño adyacente entonces se mueve para esa casilla y lo toma al niño

3- Busca la suciedad más cercana y se mueve para la siguiente casilla más cercana a esa suciedad.En caso de que sea una de las casillas adyacentes simplemente se mueve para esa casilla

4- En caso de que su único posble movimiento sea un corral lleno entonces entrará al corral y carga al niño

5- Si no se cumple ninguna de las anteriores entonces no hace nada

Para el estado B

1- Si se encuentra en una casilla sucia entonce limpia

2- Si se encuentra en una casilla de corral entonces deja al niño

3- Busca el corral más cercano y se mueve para la siguiente casiilla más cercana a ese corral. En caso de que sea una de las casillas adyacentes simplemente se mueve para esa casilla

4- En caso de que no se pueda mover entonces deja al niño o no hace nada

Modelo 2

El objetivo de este es priorizar la recogida de niños siempre está localizando el niño más cercano en el ambiente en el estado actual,he intenta moverse hacia la casilla que más lo acerque,aunque si por el camino se encuentra un suciedad la limpia.

Nota: cuando ya no hay niños fuera del corral el comportamiento de este modelo pasa a ser el del modelo 1

Las reglas basicas de comportamiento de este modelo en orden de priodad son :

Para el estado A

1- Si se encuentra en una casilla sucia entonce limpia

2- Si hay un niño adyacente entonces se mueve para esa casilla y lo toma al niño

3- Busca el niño más cercano y se mueve para la siguiente casilla más cercana a ese niño

4- En caso de que su único posble movimiento sea un corral lleno entonces entra al corral y carga al niño

5- Si no se cumple ninguna de las anteriores entonces no hace nada

Para el estado B

1- Si se encuentra en una casilla de corral entonces deja al niño

2- Si se encuentra en una casilla sucia entonce limpia

3- Busca el corral más cercano y se mueve para la siguiente casiilla más cercana a ese corral. En caso de que sea una de las casillas adyacentes simplemente se mueve para esa casilla

4- En caso de que no se pueda mover entonces deja al niño o no hace nada

Modelo 3

Este modelo prioriza la recogida de niños antes que nada.Limpia solo en caso de que no tenga más opción en otro caso sigue su camino para lograr su objetivo ignorando las casillas sucias.Esta es una idea interesante ya que los niños no pueden moverse hacia casillas sucia,entonces habría más suciedad y bajaría la probabilidad de que un niño se mueva , y una vez no haya niños en el ambiente no se generaía más suciedad a no ser por un cambio aleatorio del ambiente.Así que se ahorrarían turnos en los que limpiarías y se aprovecharían en la tarea de recoger niños para que no generen más suciedad, en plan voy a limpiar cuando dejen de ensuciar.Esta estrategia puede traer efectos negativos que veremos luego

Nota: cuando ya no hay niños fuera del corral el comportamiento de este modelo pasa a ser el del modelo 1

Las reglas basicas de comportamiento de este modelo en orden de priodad son :

Para el estado A

1- Si hay un niño adyacente entonces se mueve para esa casilla y lo toma al niño

2- Busca el niño más cercano y se mueve para la siguiente casilla más cercana a ese niño

3- Si se encuentra en una casilla sucia entonces la limpia

4- En caso de que su único posble movimiento sea un corral lleno entonces entrará al corral y carga al niño

5- Si no se cumple ninguna de las anteriores entonces no hace nada

Para el estado B

1- Si se encuentra en una casilla de corral entonces deja al niño

2- Busca el corral más cercano y se mueve para la siguiente casiilla más cercana a ese corral

3-. Si se encuentra en una casilla sucia entonces la limpia

4- En caso de que no se pueda mover ni limpiar entonces deja al niño o no hace nada

Experimentos

Se analizaros diversos tipos de ambientes con difentes características.

Se fijan unos pasan unos datos que son

-Cantidad de filas

-Cantidad de columnas

-Cantidad de niños

-Porcentaje de casillas con obstáculos

-Porcentaje de casillas con suciedad

-t el valor de cada cuántos turnos el ambiente varía aleatoriamente

Estos datos se pasan en modo de una lista

Con esos datos se generan n ambientes (n se pasa como parámetro),y en cada uno de esos ambientes se hace una simulacíon de k turnos (k también se pasa como parámetro) para cada unos de los modelos del robot.Aqui se muestra la cantidad de ambientes en las que cada robot tuvo éxito,la cantidad en las que falló y el porcentaje medio de la suciedad que quedó en cada uno de los ambientes.

Támbien hay otra simulación que muestra el entorno en consola, y dice la acción que va a realizar el robot paso a paso mostrando cada uno de los turnos.

Después de varios experimentos con juegos de datos distintos se pudo observar que el modelo 2 es el que mejor se adapta a los ambientes ya que ambientes que superan los 10x10 tiene mejores resultados.El ambientes pequeños el modelo uno tiene resultados muy favorables ,ya que este si se encuentra con un niño por el camino lo toma, y es muy fácil toparse con un niño en espacios reducidos.

Ahora el modelo 3 tiene una cosa negativa debido a que si en algún momento el robot queda trabado y no se puede mover más,entonces todas las casillas que decidió no limpiar cuando tuvo la oportunidad para limpiarlas después quedaron sucias,este modelo funcionaría mejor si se mejora la estrategia de poner los niños en el corral,ya que la que posee solo busca la mas cercana,que es probable que obstruya a la hora de poner otros niños, pienso que si se mejora esta estrategia de poner los niños en el corral tuviera mejores resultados, incluso mejores que los otros dos modelos,pero con la estrategia actual fue el modelo que presentó los peores resultados.El modelos 2 también se ve afectado si el robot se traba,pero mucho menos ya el el si limpió cuando tuvo la oportunidad.El modelo 2 es un intermedio entre el uno y el 3 por eso presentó los mejores resultados,aunque recalco, mejorando algunas estrategias el modelo 3 podría mejorar su resultados en gran media,incluso superar los demás.

Aplicación

La aplicación es un paquete del gestor de Haskell stack. Se usó Haskell 8.10.7 y no la versión más reciente en el momento por problemas de compatibilidad con las herramientas usadas en el desarrollo de esta.

Instalación

1. Instalar stack

- Referirse a la [documentación](https://docs.haskellstack.org/en/stable/install_and_upgrade/) para instalar el gestor en dependencia del sistema operativo que se use.

2. Descargar la versión de Haskell necesaria

- stack setup 8.10.7

Ejecutar aplicación

1. Abrir consola en la carpeta raíz del proyecto
2. Ejecutar stack run

Hay una función que muestra la simulación en consola paso a paso,para la misma la leyenda es la siguient

- @ : Niño
- * : Robot
- \+ : Robot con niño
- _ : Casilla vacía
- \# : Casilla con suciedad
- X : Obstáculo
- o : Corral
- O : Niño en corral

Acerca del código

Para la implementación en haskell se crearon diferentes estructuras, se ha creado que es el que le da forma a todo el entorno en la simulación.Es el tipo Environment que se que tiene la siguiente estructura.

```
data Environment = Environment {
                     size     :: Coord
                   ,  robot   :: Robot
                   ,  kids    :: [Kid]
                   ,  crib    :: [Coord]
                   ,  fullCrib :: [Coord]
                   ,  dirty :: [Coord]
                   ,  obstacle :: [Coord]
                   ,  empty ::[Coord]
                   ,  remaininKids :: Int
                   ,  moveKidProb :: Int
                   ,  dirtProb :: Int
                   ,  randomGen :: StdGen
                   ,  time ::Int
                   ,  turn ::Int
                   ,  kidPriorityRobot::Bool
                   ,  modelRobot::Int
                   , testMode :: Bool
                  } deriving (Eq ,Show)
```

Aquí otros tipos que se han definido

```
data Direction = Up
               | Down
               | L
               | R
               | UpLeft
               | UpRight
               | DownLeft
```

```
type Coord = (Int, Int)
```

Coord es una tupla de (Int,Int)

```
             | DownRight
data Kid = Kid{
            position :: Coord
        ,   isLoaded :: Bool
        ,   inCrib :: Bool

}deriving (Eq ,Show)
```

Este tipo representa a los niños donde position es la coordenada (x,y) donde está el niño en el tablero, isLoaded es un booleano que dice si el niño está siendo cargado por el robot o no y inCrib dice si el niño está en un corral o no.

```
data Robot = Robot{
            pos :: Coord
        ,   isHoldingKid :: Bool
        ,   prevPos:: Coord

}deriving (Eq ,Show)
```

Este tipo representa al robot donde pos es la coordenada (x,y) donde está el robot en el tablero, isHoldingKid es un booleano que indica si el robot está cargando un niño o no y prevPos que es la coordena anterior en la que estaba parado el robot.Este último sirve de ayuda en diversas funciones donde interviene este tipo.

Ahora en Environment

size: es una coordena que indica la cantidad de filas y de columnas en el entorno

robot : representa al robot en el entorno

kids : representa a los niños en el entorno

crib :es un array de coordenas donde está el corral

obstacle : es un array de coordenadas donde están los obtáculos

empty :es un array de coordenadas que representa las casillas vacías

dirty : es un array de coordenadas que representa las casillas con suciedad

moveKidProb : es la probalidad de que un niño se mueva en un turno

dirtProb : es la probalidad de que un niño genere suciedad cuando se mueva

randomGen : es la semilla incial para el generador de números random,se le pasa cualquier número,si se quiere que la generación de los números random sea distinto se le pasa un número distinto,esta variable es modificada automáticamente cada vez que se genera un random con otro generador.

time : representa el tiempo de variación aleatoria del ambiente

turn :el turno actual

kidPriorityRobot :un booleano para modificar el comportamiento del robot,decide si el mismo prioriza la recogida de niños o no.

modelRobot : es para escoger uno de los modelos de robots que está implementado.

La fución **generateEnvironment** recibe como parámetro todos los datos anteriores y devuelve un **Environment** con esas características.

Sobre estas estructuras se crearon diferentes funciones que las modifican para obtener los resultados deseados.La mayoria de las funciones reciben una instancia actualizada de **Environment** para trabajar sobre este.

Principales Funciones

robotAction : esta decide que es lo siguiente que hará el robot dependiendo de si el robot está cargando un niño o no.La acciones que hace el robot son moverse,moverse y cargar niño, limpiar, dejar a un niño, o simplemente no hacer nada.

unloadedAction : si el robot no está cargando cargando un niño entonces esta funcion va a decidir que es lo que va a hacer el robot según el modelo que sea.

loadedAction : si el robot está cargando cargando un niño entonces esta funcion va a decidir que es lo que va a hacer el robot según el modelo que sea.

Las dos funciones anteriores se van a apoyar en la fución **closestObjebts** que devuelve un tupla de 3 elementos, donde cada elemento es un array de coordenadas.El primer elemento representa el camino desde el robot hasta el niño más cercano,el segundo elmento representa el camino desde el robot hasta la suciedad más cercana y el tercer elemento representa el camino desde el robot hasta el corral más cercano,en caso de que esta fución devuelva un arreglo donde el fin del camino sea (-1,-1) fue que no se encontró el niño,corral o suciedad.Para hacer esto la función se apoya en un BFS.

kidsActions : esta funcion recorre todo el array de niños y para cada uno decide si se va a mover o no, generando un número random entre uno y 100 que según la probabilidad que tenga el niño de moverse,ese número decide si se moverá o no.Para recorrer la lista se utiliza la funcion auxiliar fold.

También hay otras funciones como por ejemplo cada una de las acciones del robot menciadas anteriormente es una funcion,también cada accion que realiza un niño dígase moverse,mover un obstáculo y generar suciedad es una función, estas también utilizan otras funciones auxiliares como **fold,map,filter**.Tambien se utilizó mucho las expresiones **let in , where** también se utilizaron las guardas como por ejemplo en las funciones **loadedAction** y **unLoadedAction** para decidir el mejor movimiento del robot .