Proyecto de Simulación y Programación Declarativa Agentes

David Guaty Domínguez C412
`Link en Github`

1. Marco General

El ambiente en el cual intervienen los agentes es discreto y tiene la forma de un rectángulo de N $\times$ 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 1siguiente, 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 esté 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 está ocupada: no tiene suciedad, no está 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 aparezla 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.

Robot de Casa: El Robot de Casa se encarga de limpiar y de controlar a los niños. El Robot se mueve a una de las casillas adyacentee, las que decida. Solo se mueve una casilla sino carga un niño. Si carga un niño pude 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 niño.

2. Objetivos

El objetivo del Robot de Casa es mantener la casa limpia. Se considera la casa limpia si el 60 % de las casillas vacías no están sucias.

3. Solución

La idea general del programa es mantener un ciclo principal el cual modifica un tipo de datos que representa el ambiente en donde se desarrolla la simulación. En cada turno se genera un nuevo ambiente que sirve de base para el siguiente turno. La simulación se detiene de dos formas: al llegar a una cantidad de turnos máxima y cuando el objetivo de la sección dos no se cumpla, o sea, que se tenga un por ciento de casillas limpias por debajo del 60 %. En el último de los casos de parada se considera que los robots han fracasado y se necesita cambiar de modelo de agente o aumentar la cantidad de robots o realizar otras modificaciones. Se asume que la simulación comienza con una limpieza superior al 60 %.

3.1. Modelando el ambiente

El modelo del ambiente se encuentre en el módulo World.hs.

Para modelar un estado del ambiente se consideró mantener varios conjuntos que representan las diferentes entidades que existen. Por ejemplo, el conjunto de los niños son las posiciones que ocupan los niños en el estado actual del ambiente. Así, en vez de tener una matriz, se tienen diferentes conjuntos de posiciones. Para obtener información sobre el ambiente se realizan operaciones sobre esos conjuntos. Por ejemplo, para obtener los niños que no están en el corral simplemente sería obtener la diferencia ente el conjunto de los niños y el conjunto de las casillas que ocupa el corral.

Entonces, el mundo en el que se desarrolla la simulación no es más que una tupla de conjuntos (Obstáculos, Suciedad, Corral, Niños, Robots, Robots que tienen niños). El último conjunto es solo para diferenciar los robots que tienen cargado un niño de los robots que no.

En el código un mundo no es exactamente la tupla anterior, sino que guarda un poco más de información, en especifíco, las dimensiones del rectángulo.

Modelado del ambiente

```
type Position = (Int, Int)
type Blocks = [Position]
type Dirts = [Position]
type Corral = [Position]
type Kids = [Position]
```

```
type RobotType = Int
type Robot = (Position, RobotType)
type Robots = [Robot]
type Elements =(Blocks, Dirts , Corral ,Kids, Robots, Robots)
type Dimension = (Int, Int)
type World = (Elements, Dimension)
```

En dicho módulo se implementan funciones para obtener y modificar los conjuntos según sea necesario. Además, también tiene funciones para obtener información del ambiente como el por ciento de limpieza.

3.2. Cambio del ambiente

El cambio del ambiente ocurre en dos momentos. El primero es en donde los niños se mueven aleatoriamente. El segundo es el cambio aleatorio cada t unidades de tiempo en donde se genera suciedad por los niños.

El cambio del ambiente está implementado en el módulo Kids.hs.

La función que engloba todo el cambio es allKidsMove. La cual mueve todos los niños que están "libres".

allKidsMove

```
allKidsMove:: World -> Int -> Bool -> World
allKidsMove w  = moveFreeKids w freeKids where
freeKids = getFreeKids w
```

Los niños libres son todos lo niños que no estén en el corral. Los niños que no están en el corral pero están siendo cargados por un robot tampoco se cuentan como libres. Pero como se ve en la definición de la función de getFreeKids

getFreeKids

```
getFreeKids:: World -> [Position]
getFreeKids w = getKids (getElements w)  \\  getCorral(
   getElements w)
```

en ningún momento se hace diferencia de conjunto con el conjunto de los robots para descartar los niños que están siendo cargados por un robot. Esto es debido a que cuando un robot carga un niño se elimina dicho niño de su conjunto, y se agrega un tipo de robot en el conjunto especial *Robots que tienen niños* mencionado anteriormente en la modelación del ambiente.

La función de moveFreeKids hace implícitamente una operación de fold por recursividad. Por lo que va iterando por cada niño llamando a la función kidMove pasando al siguiente niño el resultado de la función. En este caso el resultado de la función es un nuevo ambiente en el cual el niño realizó su movimiento. En cada niño, luego de llamarse a la función kidMove se chequea si es necesario generar nueva suciedad. Se le informa al módulo Kids si se debe generar o no suciedad pasando un argumento de tipo Bool como se ve en la definición de allKidsMove. El ciclo principal se encarga de chequear si es necesario generar nueva suciedad, ya que es el que está consciente del tiempo actual de la simulación.

Moviendo un solo niño: Para esto se toman todos los posibles(no necesariamente válidos) movimientos que el niño podría hacer. Luego de todos estos se filtran por una función que verifica si dicho movimiento es válido. Entonces, de todos los movimientos válidos se toma uno aleatorio mediante una función de utilidad que se encuentra en el módulo Utils.hs que se encarga de escoger un elemento aleatorio de una lista. Toda esta composición de funciones se ve en la definición de la función kidMove. O sea la composición es makeMove pickOne filter moves. La función moves devuelve todas las posiciones adyacentes a la posición del niño que están dentro del ambiente, y que están vacías o poseen un obstáculo(el cual el niño podría mover, de lo cual se encarga la función valid). Para saber si un niño puede mover un obstáculo se calcula el vector de dirección de movimiento del niño y se busca una casilla vacía en esa dirección, si existe entonces el niño puede mover el obstáculo y por tanto es un movimiento válido.

kidMove

```
kidMove :: World -> Position -> Int -> World
kidMove w pos seed = makeMove w pos (pickOne (filter (
    valid w pos) (moves w pos)) seed )
```

Generando suciedad: Para generar la suciedad de un solo niño se toma la posición en la que estaba antes de moverse y se seleccionan las casillas adyacentes vacías. Además se cuentan en todas las casillad adyacentes a dicha posición la cantidad de niños que hay. La posición anterior y las casillas adyacentes forman una cuadrícula de 9x9. De acuerdo a la cantidad de niños es la suciedad que se va a generar. Para esto se toma el mínimo entre la cantidad de suciedad a generar y la cantidad de casillas adyacentes vacías disponibles, este valor mínimo resulta en la cantidad máxima de suciedad que se puede generar. Teniendo la cantidad máxima se selecciona un número aleatorio en-

tre 0 y dicha cantidad y el resultado del número aleatorio decidirá cuántas casillas adyacentes vacías escoger. Para escoger las casillas se usa el método de escoger elementos aleatorios de la una lista implementado en el módulo Utils.hs. Toda esta combinación de pasos se puede ver en la composición de las funciones en la definición de la función generateDirt.

generateDirt

```
generateDirt :: World -> Position -> Int -> Bool ->
    World
generateDirt w pos seed False = w
generateDirt w pos seed True = addDirt w (fst (
    selectItemsFromList (howManyDirt w pos seed) (
    emptyAdjacent w pos) seed))
```

3.3. Agentes

El comportamiento de los robots está en el módulo Robots.hs.

Los robots se comportan de dos maneras. Una manera es más reactiva y la otra manera es más proactiva.

Un tipo de robot en cada turno procesa su ambiente y se encamina a lo más cerca que detecte. Si lo más cerca que ve es un niño tratará de ir a recogerlo, y si es una suciedad tratará de ir a limpiarla.

El otro tipo de robots en cada turno persigue específicamente a los niños, ignorando la suciedad. Este robot tiene el objetivo de llevar a los niños al corral lo antes posible para detener la generación de suciedad y así lograr mantener la habitación por encima del por ciento esperado indefinidamente.

Otra diferencia es que si el tipo de robot más reactivo tiene un niño encima y se encuentra una suciedad tratará de limpiarla. Sin embargo, el más proactivo la ignora y camina lo más que pueda hasta el corral.

Ambos tipos de robots tienen elementos de sociabilidad en el sentido en que un robot decide ir hacia un lugar solo si ningún otro robot ha decidido ir. Por ejemplo, un robot no va a limpiar una suciedad si otro ya decidió ir a por ella.

Para encontrar caminos hacia los objetivos se usa una función bfs que se encuentra en el módulo World. Esta función recibe, entre otras cosas, una lista de destinos partiendo desde una posición y devuelve dos mapas(diccionarios) que especifican las distancias de cada posición con respecto a la inicial y los

padres en el árbol generado por el bfs de cada posición (para obtener los caminos).

La función de llamada para moverse todos los robots en un turno es análoga a la de los niños ya que actúa como un fold.

Para mover un solo robot se toma en cuenta su tipo y en base a eso los objetivos del bfs cambian. Pueden ser niños o puedes ser niños y suciedad. Se filtran por los niños libres.

3.4. Experimentación y resultados

Luego de varios experimentos se llega a la conclusión de que en muchos casos los robots más reactivos tienen un mejor rendimiento que los proactivos. Ambos cumplen los objetivos, pero el reactivo mantiene un promedio de limpieza más elevado que el proactivo. Para casos en que existe mucha variación del ambiente los reactivos se manejan mejor. En caso de tener muchos niños y poca variación aleatoria del ambiente los proactivos al ignorar la suciedad llegan al objetivo más rápidamente de tener asegurado un por ciento de limpieza por encima del 60 %.