Proyecto de Simulación y Programación Declarativa
URL Github: https://github.com/Carlitonchin/Agentes-Reactivos

Carlos Alejandro Arrieta Montes de Oca C412

Orden del Problema:

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 varíar 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 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 2niño 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.

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

Modelos Seguidos

La idea general de la solución consistió en implementar una idea greedy en la que cada agente busca cumplir el objetivo de mejor rendimiento en el momento, si en un momento dado 2 agentes diferentes tienen el mismo objetivo, el agente que lo puede conseguir con un menor costo (cantidad de turnos) es el que se queda con la tarea, y el otro pasa a buscar hacer otra cosa

Modelo 1:
A continuación presento los distintos objetivos que se plantean los agentes y sus respectivos costos y condicionantes

1. Limpiar suciedad: para que un agente se plantee este objetivo debe estar en la misma casilla que una suciedad. Costo = 0

2. Dejar niño en la cuna: para que un agente se plantee este objetivo debe estar en una cuna con un niño cargado. Costo = 0

3. Ir a cargar un niño: para que un agente se plantee este objetivo no deben cumplirse ninguna de las condiciones de los 2 objetivos anteriores y, además, el agente no debe tener cargado a un niño (los robots no pueden cargar a 2 niños al mismo tiempo). Costo = n (n: cantidad de pasos que el robot debe dar para llegar hasta la casilla en la que el niño se encuentra en ese momento)

4. Ir a limpiar una suciedad: para que un agente se plantee este objetivo no deben cumplirse ninguna de las condiciones de los 2 primeros objetivos. Costo = s + 1 (s: cantidad de pasos que el robot debe dar para desplazarse hacia la casilla en la que está la suciedad)

5. Ir a dejar a un niño en la cuna: para que el robot se plantee este objetivo no se pueden cumplir ninguno de los 2 primeros objetivos y debe tener un niño cargado. Costo = c/1.5 + 1 ()

Modelo 2:

Es prácticamente igual al Modelo 1 con la única diferencia que en el objetivo #3 el costo es n/2. Funcionan bastante similar, lo que este segundo método está orientado a llevar a los niños a la cuna más rápido descuidando a lo mejor algunas suciedades (Funcionan bastante bien los 2)

Ideas seguidas para la implementación:

Para la implementación separé la lógica en varios módulos:

. *Elementos:* en este módulo se encapsula la estructura de los elementos que conforman el tablero: niños, robots, suciedad, obstáculos, y cuna. Para cada uno de ellos hay un tipo que forman parte de la clase *Posicion*

. *Tablero:* contiene el tipo de dato *Tablero* que encapsula la lógica de la matriz de la simulación. Un tablero está formado por listas de *Elementos.* En este módulo hay funciones útiles para saber que tipo de elemento hay en cierta posición, saber si un robot o un niño pueden moverse hacia una casilla, saber si un robot está cargando o no a un niño, etc

. *Pintar:* este módulo se encarga de llevar a un imprimir un tablero en la consola

. *Escribir:* su única función es escribir y borrar elementos del tablero en una posición que le indiquen

. *Movimiento:* como su nombre lo indica es el encargado el movimiento de los elementos, para ello se auxilia del módulo *Escribir,* pero aquí está toda la lógica de lo movimientos

. *Objetivo:* se encarga de calcular los objetivos de los agentes

. *FunciónAgentes:* se encarga de hacer cumplir los objetivos a los agentes

. *Aleatorio:* módulo encargado de darle un poco de ´no determinismo´ a la solución, es usado para generar el estado inicial del tablero, y luego también para el movimiento de los niños

. *MovimientoNinhos:* este módulo es usado en el turno del ambiente (cada t iteraciones), y se encarga de mover a los niños y generar suciedad siguiendo los requerimientos orientados

. *Listas:* contiene algunas funciones útiles para operar con listas (pertenece, agregar un elemento a una lista, eliminar, indexar, etc)

Detalles de implementación

En cada iteración cada robot hace un bfs para calcular los objetivos, y se queda con el mejor de todos, si un robot descubre que tiene un objetivo compartido con otro robot,

en ese momento revisa quién lo puede hacer con menor costo, en caso de que sea el robot en cuestión lo toma y le avisa al otro robot que debe recalcular sus objetivos, en caso contrario ignora ese objetivo y continúa su búsqueda

Luego de que culmine el bfs, cada robot va a tener claro su objetivo, el costo que le va tomar conseguirlo, y el primer paso que debe dar para hacerlo

Pasos para correr el proyecto:
En el módulo *Main* que está ubicado en la carpeta *app* tiene en la parte superior las 'variables' (en realidad son funciones) que puede modificar para correr una simulación. Los nombres están bastante expresivos en general, excepto las 3 últimas:

sem: esta es una semilla que se va a usar para generar aleatoriedad, si no cambia este parámetro va a obtener la misma simulación una y otra vez (suponiendo que dejó el resto intacto)
tAmbiente: es un valor que indica cada cuantos pasos de los agentes se mueven los niños
cantIteraciones: es la cantidad de iteraciones que se quieren simular, se debe tener cuidado de no ingresar valores ´absurdos´, ya que el sistema no valida que se ingresen datos correctamente (ejemplo, en un tablero de 10X10 no se puden crear 200 niños)

```
17
18  --CAmbiar estas variables para obtener una simulacion diferente
19  cantNinhos = 5
20  cantRobots = 3
21  cantSuciedad = 3
22  cantObstaculos = 5
23  anchoTablero = 10
24  largoTablero = 10 :: Int
25  sem = 3
26  tAmbiente = 1
27  cantIteraciones = 100
28  --------------------------------------------------------------------------
29
```

Si tiene instalado algún paquete que le permita visualizar emojis en la consola podrá ver los elementos del tablero de mejor manera

 Niño  Cuna  Suciedad  Obstáculos
 Robot

En caso que no tenga puede instalar un paquete (-- sudo apt install fonts-emojione)

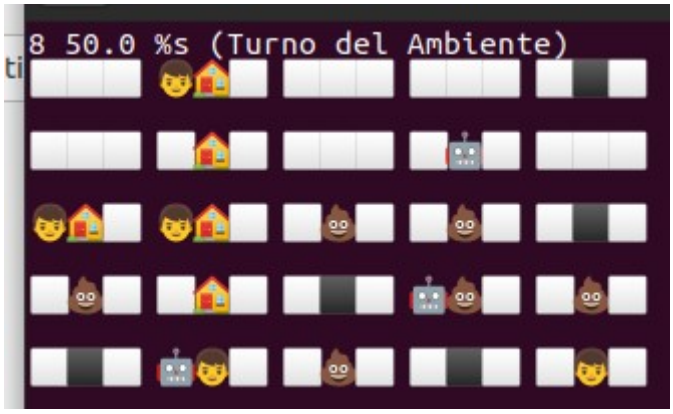
Si no desea instalar ningún paquete cada elemento saldrá con su inicial, niño (n), cuna (c), etc. Pero debe setear la función *emoji* en *falso (*está en el módulo *Main* debajo de las variables de la simulación*).* Por defecto está seteada en *True*

```haskell
--CAmbiar estas variables para obtener una si
cantNinhos = 5
cantRobots = 3
cantSuciedad = 3
cantObstaculos = 5
anchoTablero = 5
largoTablero = 5
sem = 3
tAmbiente = 1
cantIteraciones = 100
--------------------------------------------------------------------------------

emoji = True

tableroSemilla = iniciarTablero largoTablero anchoTablero sem
```

En caso de que haya un niño y un robot en la misma casilla se podrá saber si el robot lo está cargando por su posición, si el robot está a la derecha: cargando. Si está a la izquierda del niño : no lo está cargando

De forma general un tablero se observa de la siguiente manera:

El tablero luce así



El primer número indica el # de la iteración (8), luego se ve el % de suciedad y por último una indicación en caso de que el ambiente haya variado aleatoriamente (turno del ambiente)

Si no tiene un paquete de emojis debe substituir al ninho por una N, al robot por una R, etc

Simulaciones:

Luego de realizar varias simulaciones en un tablero de 10X10, 5 obstáculos, 3 suciedades de inicio, y el cambio del ambiente todos los turnos:

. Con 5 niños se necesitaron 2 robots para mantener más del 60% del tablero vacío
. Con 10 niños se necesitaron 4 robots
. Con 15 niños se necesitaron 8 robots

Más de 15 niños con esta amplitud de tablero fue muy complicado para los agentes mantener la limpieza, porque se llenaba muy rápido

Ampliando la amplitud del tablero a 20X20 y manteniendo el resto de variables

. Con 10 niños se necesitó un solo robot
. Con 20 niños se necesitaron 2 robots
. Con 30 niños se necesitaron 10-11 robots