Proyecto De Simulación

Amanda González Borrell C411

Link :https://github.com/amyGB99/Simulation-and-Declarative-Project

Para ejecutar el proyecto: “ Carbal new-run”

El tablero debe ser introducido manualmente en el código del proyecto , abra el documento Main.hs y en el existe un tablero ejemplo.

El proyecto está dividido en los siguientes módulos Main, Board ,Utils, Structure

En Utils algunas funciones que se utilizan en varias partes del código .En Board el pintar el tablero . En Structure están localizado la forma en que guardan los datos:

Tenemos el ambiente el cual debe tener una serie de propiedades: cantidad de filas, columnas, tiempo y de más. También posee listas las cuales guardan a los agentes, esta es la siguiente estructura: Agent la cual tiene un tipo, fila y columna donde se localiza, continuando con el agente el mismo en la lista robots guarda todos los robots en children todos los nenes y así sucesivamente con todos los miembros del ambiente : obstáculos , corrales y suciedad. También tenemos la estructura Action de la cual hablare un poco mas adelante.

La implementación de la simulación está realizada de la siguiente forma la función main en el cual debe crear su tablero este llama a la función simulation el cual se ejecuta las t_final veces que elija , la función simular se encarga de dado un estado inicial de un ambiente y un generador aleatorio se llame a alterRobot encargado de devolver un nuevo estado del ambiente con las acciones realizadas por los robots y luego este estado se le pasa al la función encargada de crear y ejecutar las acciones que ejecutan los niños AlterChildren luego de que se ejecuten ambas funciones debe actualizarse nuevamente el ambiente cambiando el t_final pues se concluyó un turno.

Como las acciones de los Robots se ejecutan primero entonces empecemos explicándolos:

Las acciones de los robots son del siguiente tipo Action name : "move" , "clean up" ,"pick up child","leave child" y una tupla de posición (i,j) q indica el robot va a recoger niño en esa posición o va a limpiar en esa posición y lo mismo con las otras , antes de explicar la manera en que se generan dichas acciones pasemos a lo más fácil que es luego de tenerlas ejecutarlas , cada acción tiene su función de ejecución por tanto el execute de robot general va por cada acción y según el tipo ejecuta la función que ejecuta dicho tipo. En el caso de la de de mover el robot solo sería crear un nuevo ambiente en donde en la lista robot aparezca el robot con la posición cambiada para eso creamos un nuevo ambiente con una nueva lista robot que incluye el cambio, para todas los movimientos se utiliza una función updateEnvironment que se le especifica la lista en la que se va a buscar el agente q va a cambiar de posición inicial y se le pasa la final y esta función te devuelve el estado del ambiente(tablero) con dicho cambio realizado. En el caso de la ejecución de "leave child y ,"pick up child", que son opuestas simplemente como bien mencione anteriormente existe una lista que guarda el ambiente con los niños cargados y robots que cargan seria eliminar o agregar a dicha lista al robot que carga o deja al niño y al niño q es cargado o dejado . En la limpieza seria eliminar el agente suciedad de la posición que se indico eliminándolo de la lista que guarda el ambiente con todas las casillas con suciedad.

Cuando decimos que un robot ejecuta varias acciones es porque en un mismo turno se pueden realizar varias de ellas , para ello explicare el modelo reactivo de agente que implemente una arquitectura Brooks mediante la cual genero decisiones a partir de comportamientos que me permitirán cumplir mi objetivo que será limpiar la habitación . En mi estructura es importante

controlar a los niños para que no continuen ensuciando ya que ellos son la fuente de suciedad por tanto ya explicando cuales son las reglas a seguir de mi arquitectura :.

if hay niño cargado -debo llevarlo a un corral vacío por tanto necesito el camino mas corto hacia un corral vacío luego de tener este camino que calcularé por BFS si la longitud del camino es dos entonces la acción a generar es [Move, Move ,"leave child"] . Si la longitud es mayor a dos entonces genero la acción [Move, Move] ya que quiero acercarme al corral por el camino mas corto. Si el camino tiene longitud uno me muevo y lo dejo [Move,"leave child"] y si es de longitud cero es que estoy sobre un corral vacio lo dejo entonces. En caso de que no exista niño cargado necesito colocar en el corral a los que estén sueltos por tanto buscaré el camino de longitud mínima a un niño que no este en el corral if el camino es de longitud 1 se mueve y lo recoge [Move, pick up child] y el camino es mayor entonces me muevo una vez para cercarme .Luego si no hay niño suelto ni cargado que yo luego de tenerlo pueda llevarlo al corral , buscare el camino mas corto a una suciedad. if el camino es de longitud uno me muevo para en el próximo turno limpiarla porque si el camino es de longitud cero entonces limpio .

Para realizar lo anterior se necesitan ciertas funciones como:

BFS en la cual tendré una lista pendientes que será una lista de tuplas posición [((i,j),way)] y way seria una lista de tuplas donde se guardara el camino desde donde comenzó el BFS hasta la posición i,j

El Bfs se comienza en los robot . Entonces se saca el primer elemento de pendiente como en esa posición pueden localizarse mas de un elemento entonces incluyo al array final que tiene la forma (Agente ,way) es para cada agente que esta en una posición en la tupla se guarda su camino hasta ahí, entonces incluyo a este niño suciedad o corral a la lista final con su camino. Luego de eso decido si expandir o no esa posición que saque de pendiente , el expandir es ir por todos los adyacentes de esa posición y si no están visitados que para eso se tiene una lista de tuplas de posiciones que indican que esa posición ya se visito, si no lo están los agrego a pendientes con su camino . Ahora bien para decidir si se expande se tiene en cuenta : se expande si es casilla vacía, no hay un robot,no hay obstáculo , existe niño o corral vacío. En caso de expandir ya explique q se guardan los adyacentes validos y se actualiza pendiente, y visitado , luego se llama recursivo para sacar otro elemento de la cola así hasta que este completamente vacía , significa que ya los visite a todos y tengo en mi variable result la lista de agente , camino pues cada vez que de pendiente se expulsa una posición con agente que me interesa (niño, corral y suciedad) yo actualizo mi result con dicha información.

El Bfs esta formado por el Innit, el bf, utiliza la función expand, addAdjacentPending, validateAdjacent , entre otras las funciones están comentadas el código su utilidad .

También para generar las acciones necesito saber de todos los ni;os que tengo suelto cual es el mas cercano para eso se creo una función que retorna el min de los caminos a agentes del mismo tipo .

Ahora pasemos a los niños ello solo tienen dos tipos de acciones a generar [Move, dirty] moverse y ensuciar y estas las hace en el mismo turno. En la proceso de generación de la función Move genera las posiciones para arriba , abajo y ambos lados izquierdo y derecho , luego verifica q este vacía o que no este ocupada con ningún otro agente a no ser obstáculo , ente momento de la generación es una suposición puede que la casilla escogida q cumple las características anteriores y que fue escogida random al final a la hora de ejecutar la acción no pueda realizarse de ahí la idea de simular de q un niño puede moverse o no . En el caso de la suciedad escoge de las casillas 3x3 luna random q este vacía, el siempre genera suciedad siempre y cuando existan casillas vacías, puede q no se mueva per si genera suciedad. Luego de tener generadas estas acciones cada una se ejecuta de manera independiente como en el caso del robot .

En la ejecución de las acciones del niño como bien mencione tenemos una casilla a la cual el posiblemente se podrá mover pero esto no es seguro si esta casilla esta vacía entonces se puede mover utilizamos nuevamente la función updateEnvironment para actualizar dicha posición, si lo que hay es un obstáculo entonces empezare un camino recursivo en la misma dirección de movimiento, dicho camino empieza en el obstáculo en la misma dirección q se genero el desplazamiento entre la inicial y el obstáculo , el obstáculo busca una posición final si esta esta vacía entonces se puede realizar el movimiento si hay otro agente q no es obstáculo entonces no se puede realizar el movimiento y si es otro obstáculo vuelvo a llamar recursiva hasta llegar a una posición vacía con otro agente o fuera de los limites del tablero. Luego como guarde una lista con las posiciones recorridas empiezo a ejecutar de dos en dos ,para esto modifico la lista de modo que empiezo a ejecutar los desplazamientos desde el ultimo hasta el primero .En el caso de la ejecución de la acción suciedad se agrega esta casilla generada al ambiente.

Para las implementaciones se utilizaron guardas, cases, let in , tuplas listas , la mezcla de estas , los data y se utilizo la recursividad para crear la mayoría de las funciones.

Otro modelo que se podría implementar para la generación de las acciones de los robots seria tomar el objetivo limpiar primero antes de realizar cualquier otra actividad de las que yo comente con anterioridad . En el modelo presentado para generar las acciones al controlar la fuente de suciedad que son los niños posibilita que la casa quede limpia mas rápido.

Los métodos de ejecución son los executes distintos para cada acción y para cada agente, los mismo que los de la ejecución