Simulación-Programación Declarativa Proyecto 2

Enrique Martínez González C-412

1 Orden del problema asignado:

2 Principales ideas seguidas para la solución del problema:

Para comenzar el simulado del problema planteado se crea una habitación representada por una cuadrícula de N por M, en la que cada celda poseerá un tipo en el entorno de la habitación, estos tipos pueden ser varios y representan todos los posibles estados en los que se puede encontrar una casilla. Esta cuadrícula la llamaremos tablero y encima de este es donde ocurren todos los enventos. En un comienzo se generan encima de este la cantidad de niños, robots, corrales, obstáculos y basura que introduzca el usuario, además selecciona el tipo de modelo de agente que desea utilizar para la realización de la simulación, resaltar que las dimensiones del tablero son escogidas igualmente por el usuario.

Una vez que se posee un tablero inicial, comienza el simulado con las características mencionadas en la orientación. Primeramente se desplazan los niños si le corresponde en ese turno, pues el usuario introdujo la cantidad T de turnos que tienen que suceder para que el ambiente cambie, este cambio de ambiente es el desplazamiento de los niños y la generación de basura si corresponde. Los niños se desplazan con probabilidad $1/2$ y una vez desplazados se genera basura con probabilidad $1/2$ en todas las cuadriculas de 3 por 3 en las que estaba presente justo antes de desplazarse siguiendo las normas de generación de basura planteadas en la orientación.

Con todos los niños en el lugar que le corresponde, se procede a mover los roboces presentes en el tablero, siguiendo el modelo seleccionado por el usuario. En la implementación existen 3 modelos de agentes, los tres se diferencian en la acción que realizan cuando se encuentra el robot solo en una casilla. Si el robot está sobre una casilla sucia, este la limpiará siempre, si está encima de una casilla con un niño, este lo cargara siempre, si está cargando un niño buscará el corral más cercano y si ya está encima de un corral, dejará al niño en este sitio, si ya lo había dejado pues tendrá el mismo comportamiento que si estuviese en una casilla vacía, que a su vez es el mismo comportamiento del robot cuando se encuentra en un corral vacío. Ahora, cuál es el comportamiento de un robot encima de una de estas casillas. Pues es en esta desición en la que varían los modelos implementados. En el primero de ellos, el robot irá tras el niño más cercano, en el segundo irá a limpiar la basura más cercana y en el tercero irá a por el niño o basura más cercana. En los dos primeros, en caso de que no exista su objetivo, buscará la basura más cercana en caso del primer modelo o el niño más próximo en caso del segundo. En caso que no exista basura o niño alcanzable solo mantendrá su posición. En el tercer modelo en caso de que no exista basura o niño alcanzable sencillamente no se moverá. Resaltar el caso que el robot cargue a un niño y no existan corrales alcanzables, el robot limpiará la basura más cercana.

La simulación culmina cuando se alcanza una cantidad de turnos máximas o cuando la habitación posee el 60% de las casillas vacías limpias como indica la orientación del problema.

Para una mejor observación de la simulación se imprime en la consola el estado del tablero a lo largo de los turnos, las casillas son representadas por una matriz, en la que

cada casilla puede poseer un máximo de 3 letras representando el tipo que está ocupando la celda en ese instante. A continuación se muestra el significado de cada posible estado de las celdas de esta cuadrícula:

- vacío = "[ ]"
- robot = "[R]"
- niño = "[C]"
- basura = "[T]"
- corral = "[H]"
- obstáculo = "[X]"
- robot y basura = "[RT]"
- robot y niño = "[RC]"
- robot y corral = "[RH]"
- niño y corral = "[CH]"
- robot, niño y corral = "[RCH]"
- robot, niño y basura = "[RCT]"

3 Modelos de agentes considerados:

4 Ideas seguidas para la implementación:

Para la represetación del tablero se utilizó una lista de listas en la que cada casilla contiene sus coordenadas para una facilidad en su empleo, y el tipo de celda en cuestión, además de dos propiedades más que indican si se está dejando algo o recogiendo algo, estas existen para que puedan coexistir robots y niños en una misma casilla en múltiples estados.

El tablero inicial es en comienzo una cuadrícula de N filas y M columnas en el que todas las casilas son de tipo *empty*. Para la generación de los distintos tipos de celdas en el tablero inicial se utiliza una selección aleatoria de una cantidad igual a la escogida por el usuario del tipo de casilla en específico sobre una lista de todas las celdas de tipo *empty* que se encuentran en el tablero en ese momento. El caso de corrales es especial pues estos deben estar adyacentes. Para esto se crea en comienzo un corral en una posición aleatoria, y luego se van generando corrales en una casilla adyacente vacía a un corral aleatorio de los ya colocados hasta llegar a la cantidad de corrales igual a la cantidad de niños seleccionadas en un comienzo.

Una vez creado el tablero se desplazan los niños. Estos se mueven cada T turnos, con una probabilidad de 1/2 cada niño, esta probabilidad es facilmente modificable en el

código. Si en la dirección que este se va a mover existe un obstáculo, se busca la primera casilla libre en la dirección del movimiento a realizar para poder desplazar todos los obstáculos en esta dirección, en caso de que no exista, el niño mantiene su posición. En caso de que el movimiento del niño sea satisfactorio se procede a generar basura, esta se crea buscando todas las subcuadrículas de 3 por 3 a las que pertenecía el niño, por cada subcuadrícula se cuenta la cantidad de niños que hay dentro de las 9 celdas y la cantidad de basura que hay en estas. Dados estos números se sabe el número máximo de basura que se puede generar, en caso de que sea mayor que 0, se recorre cada celda de la subcuadrícula y se genera basura con probabilidad 1/2, resaltar que esta probabilidad se puede modificar facilmente en la implementación, una vez se acaban las celdas en la subcuadrícula o se llega al máximo de basura permitido, se pasa a la siguiente subcuadrícula.

Con los niños desplazados, se mueven los roboces pertenecientes al tablero resultante. Con este fin cada estado de un robot tiene un objetivo en su movimiento. Este objetivo cambia con respecto al modelo seleccionado en un comienzo por el usuario para los estados en los que un robot está solo, está en un corral vacío o está en un corral con un niño pero ya lo dejó en un turno anterior. Para cumplir el objetivo correspondiente a cada estado en los tres modelos se realiza un búsqueda de la casilla que satisface la condición de ser la celda deseada más cercana, para esto se implementó una matriz de distancia que se expande desde el robot analizado en ese instante, en donde no puede atravesar los tipos de celdas que no pueden poseer un robot encima especificados en la orientación y se toma de los objetivos el que menor distancia con respecto al robot tenga. Con esta misma matriz de distancia se genera el camino que debe tomar el robot y así saber que dirección debe tomar este en el turno. Con la trayectoria que debe serguir el robot se realiza el movimiento para acercar a este a la celda deseada. En caso de que la celda deseada no exista, o no sea accesible, el robot busca otro objetivo secundario de menor importancia, en caso del modelo que prioriza la recogida de niños, su objetivo secundario es la basura, y en el modelo que su casilla deseada es la basura, al no poder llegar a esta, buscará a un niño, el modelo que busca el objeto más cercano no posee objetivo secundario. Si en la trayectoria un robot coincide con una basura, este la limpia, y si coincide con un niño este lo carga. Si un robot tiene un niño cargado, su objetivo principal en todos los modelos es dejarlo en un corral y su objetivo secundario es la basura. En caso de que un robot no pueda cumplir su objetivo primario ni secundario, este mantendrá su posición sin realizar ninguna acción.

La simulación correrá mientras la cantidad de turnos transcurridos sea menor que el máximo fijado, resaltar que esta cantidad máxima de turnos puede ser modificada facilmente en el código en el que por defecto está con un máximo de 1000 turnos, y mientras la cantidad de casillas limpias sea menor al 60% de las casillas ensuciables en el tablero.

5 Consideraciones obtenidas a partir de la ejecución de las simulaciones del problema:

6 Enlace al repositorio del proyecto en Github:

https://github.com/kikeXD/simulation-haskell