Masiel Villalba C-412

Yamile Reynoso C-412

Ideas

Para la implementacion del robot hemos modelado dos comportamientos reactivo y proactivo.

Por un lado tenemos a nuestro robot proactivo que tiene un plan muy bien definido, e "inquebrantable": recoger primero a todos los ninnos.

Puesto que ellos son los responsables de la suciedad, una vez que esten tranquilos se podra comenzar con la limpieza y que no sea en vano. Esta planificacion

para cuando el intervalo de cambio del tablero es grande, es verdaderamente ideal, pues seria suficiente para que el robot recoja y limpie todo antes de que

se desordene el ambiente de nuevo. Pero para cuendo el t es pequenno, estariamos ante un ambiente bien dinamico, en el que el objetivo de ubicar a los

ninnos nunca seria cumplido, y por tanto el de la limpieza tampoco.

Hemos escuchado algunas veces que el comportamiento reactivo figura en realizar acciones random en cada turno.

El nuestro es aparentemente similar al proactivo, al menos en el inicio del codigo, pero en realidad, en lugar de enfocarse ciegamente en recoger

primero a todos ninnos siempre realiza la accion mas inmediata. Los mas inmediato realmente seria limpiar siempre primero, pero en ninguno de los dos

casos lo consideramos, puesto que el robot es el primero en actuar en cada turno y justo despues de su actuacion algun ninno puede ensuciar la casilla que

acaba de limpiar. Por eso para los dos casos tenemos que siempre que se pueda deje a un ninno en el corral si carga alguno y que recoja a cualquiera

que tenga cerca si no carga ninguno. Entonces, regresando a la reactividad, este ademas de hacer la accion mas inmediata, se mantiene recogiendo ninnos,

o al menos intentandolo, mientras el porciento de suciedad este por debajo de 40, y cuando se alcance este tope, ya empieza a preocuparse por priorizar la limpieza.

Tambien hemos asumido que el robot puede limpiar aunque lleve un ninno cargado. En resumen, creemos que un agente reactivo no es aquel que no se plantee

objetivos, claro que no, sino que es sensible al ambiente, y sabe reordenar sus prioridades segun como sea afectado por las condiciones en que esta.

Esta consciente de que sus planes pueden frustrarse por razones externas y por eso siempre hace lo que es mas seguro, mas urgente.

Caracteristicas de la implementacion

El predicado principal es ....

Para representar nuestro tablero no hemos utilizado ninguna matriz, sido que lo hemos hecho con predicados dinamicos.

Tenemos: corral/1, ninno/1, robot/1, sucio/1, obstaculo/1, cada uno recibe una tupla que representa una casilla y retornan true si en esa casilla hay un elemento

de ese tipo. Por ejemplo, sucio((4,5)) es true si la casilla (4,5) esta sucia. Para ubicar un elemento en una casilla utilizamos assert y para quitarlo, retracall.

De la manera que disennamos el tablero garantizamos que nunca se haga assert a una casilla que no exista en el tablero de N,M.

El predicado carga_ninno/0 es true si el robot lleva cargado algun ninno. Cuando el robot carga un ninno, este "deja de existir" temporalmente, puesto que

ninno(Casilla_del_robot) es false. Tambien tenemos tablero/1, no_ninnos/1, que no serian tan necesarios pero se han agregado por tener mas a mano dos datos

importantes: la cantidad de ninnos y las casillas del tablero N*M, que en una misma ejecucion no cambian. no_ninnos(X) es true si hay X ninnos en el

tablero.

"generar_tablero" recibe N,M (dimensiones del tablero), Ns (cantidad de casillas sucias), No (cantidad de obstaculos). Primero ubicamos los corrales

para que cumplan lo que se plantea en la orientacion, de lo cual se encarga "lograr_K_corrales" y luego por cada corral ubicado se elige una casilla

random para ubicar un ninno. Finalmente se ubican random el robot, las Ns casillas sucias y los No obstaculos.

"lograr_K_corrales" recibe Disponibles (casillas en las que todavia no se ha ubicado ningun corral), K (cantidad de corrales que faltan por construir),

N,M (dimensiones del tablero) Logrados (cantidad de tableros que san construido hasta el momento). Este predicado se apoya en corral/7.

corral(Disponibles,Frontera,[X|Cor_rest],Size,N,M) es true si [X|Cor_rest] es una lista de casillas de Disponibles que representan un corral de tamanno Size

cuya frontera es Frontera y X es una casilla de la frontera de Cor_rest que tambien es un corral.

A la "frontera" de un corral pertenecen todos aquellas casillas que esten en la misma fila o columna de alguna casilla del corral.

Este termino es importante pues los corrales se construyen eligiendo siempre una casilla random de su frontera.

actua_robot_proactivo(Robot_pos,Cuadro) y actua_robot_reactivo(Robot_pos,Cuadro) modelan los dos comportamientos del robot implementados, (Cuadro es la

lista de casillas a las que se puede mover el robot en un paso), y mover_Ninnos(ListNinos) simula las acciones de los ninnos.

mover_robot_mas_cercano(X,Buscadas,Pasos) es true si el robot(que esta en X) puede dar Pasos pasos en direccion a la casilla de Buscadas que mas cerca le queda.

La cercania la medimos en funcion de la distancia euclidiana. Si no puede moverse en direccion de la casilla mas cercana Q (porque hay un obstaculo o un ninno)

entoces se mueve a la casilla libre que mas cerca esta de Q.