PRÁCTICA 3 – TÉCNICAS DE LOS SISTEMAS INTELIGENTES

DISEÑO E IMPLEMENTACIÓN DE UN DOMINIO DE PLANIFICACIÓN CLÁSICO

BASADO EN STARCRAFT

USANDO PDDL Y METRIC-FF

Antonio José Blánquez Pérez 3ºCSI
Grupo 2 – Viernes

EJERCICIO 1

El primer ejercicio sirve para plantear la base del dominio y del problema, veamos como se ha hecho. Para el dominio se han establecido 4 tipos de clase objeto: localizacion, recurso, unidad y edificio, y 5 constantes que representarán los subtipos de estos objetos: vce como tipo_unidades, centro_de_mando y barracones como tipo_edificio y minerales y gas como tipo_recurso.

Representados los tipos necesarios se pasa a describir los predicados, habiendose añadido algunos más de los que se pedía en el guión. Para determinar si un edificio o una unidad están en una localización concreta se ha decidido hacerlo con dos predicados diferentes('en' para unidades y 'construido' para edificios) por simple comodidad y simplicidad, ya que si no sería necesario que ambos tipos pertenecieran a la misma superclase. Para determinar si dos caminos están conectados un simple predicado 'conectado' con las dos localizaciones. Para asignar un nodo a una localización (y para comprobar si está en ella) se ha establecido el predicado 'nodo'. Para indicar cuando un VCE está extrayendo un recurso se marcará con un predicado 'ocupado'. Para definir qué tipo de recurso necesita cada tipo de edificio tenemos un predicado 'requiere'. A parte de estos, que son los que pide el ejercicio, se han añadido: 'tenemos' para saber si tenemos ese tipo de recurso concreto(ya que si lo tenemos, al suponer que tenemos infinitos, es solo una pregunta de sí o no y no de cuantos recursos de ese tipo tenemos) y otros tres más para establecer una relación entre un tipo y su constante correspondiente('edificioTipo', 'unidadTipo' y 'recursoTipo', por ejemplo [edificioTipo Barracones1 barracones]). Se han definido también las tres acciones que pide este primer ejercicio a partir de los predicados anteriores con los criterios que se explican a continuación:

- Navegar: Si una unidad no está asginada aún y está en una localización concreta, y esta está conectada con otra, se puede mover la unidad a esta otra localización.

- Asignar: Si una unidad VCE no está ocupada en un nodo y está en una localización concreta donde hay un nodo de un tipo de recurso específico, se podrá asignar la unidad VCE a extraer recursos y asumir que ya tenemos este tipo de recurso.

- Construir: Si una unidad VCE que no está ocupada extrayendo en un nodo está en una localización en la que no se ha construido ningún edificio, se podrá construir un tipo de edificio para el cual tengamos el tipo de recurso que requiere su construccion y asumir que este edificio se ha construido en esa localización concreta. También se debe tener en cuenta que el edificio no este construido en alguna otra localización.

Con el dominio ya definido basta con establecer como objetivo tener construido Barracones1 en una posición arbitraria y establecer los objetos y sus valores iniciales: las posiciones de los 3 VCE con 'en', el centro de mando con 'construido' y los cinco nodos con 'nodo' se han establecido también arbitrariamente(basta con cambiar esta localización al principio de init para ver más casos), las localizaciones son locX_Y siendo X e Y localizaciones en un grid 5x5 y se han introducido todas las combinaciones de posiciones conectadas haciendo uso del plugin *Misc PDDL Generators* del editor web de PDDL teniendo en cuenta que solo podemos movernos a derecha, izquierda, arriba y abajo y no en diagonal; se ha establecido el tipo de cada objeto con los predicados antes indicados y el recurso que requiere cada edificio según lo indicado en el guión.

Si ejecutamos ambos archivos con *Metric-FF* tal cual se ha indicado en el seminario 3 obtenemos una planificación que nos lleva a la construcción de los barracones cumpliendo los requisitos propuestos, por lo que se ha llegado a la conclusión de que el ejercicio se ha resuelto correctamente. Se puede comprobar que sigue funcionando si alteramos las características del mapa o los objetivos.

EJERCICIO 2

Este ejercicio es sencillo, ha bastado con añadir en el dominio el tipo de edificio extractor, añadir a la acción asignar la precondición de que, o el nodo asignado no sea de gas, o deba de tener construido encima un edificio de tipo extractor, y hacer los cambios pertinentes en el archivo de problema: se añaden 2 extractores, Extractor1 y Extractor2(uno para cada nodo gas), se les asigna el tipo de edificio extractor y se añade el predicado correspondiente para plasmar que los extractores requieren minerales para ser construidos.

Con las acciones anteriores ya cumplimos las restricciones mencionadas en este ejercicio, aunque para el objetivo planteado(construir los barracones) no hay diferencia alguna en la solución, por lo que se han dejado algunas líneas comentadas que piden construir otro centro de mando con las que se puede comprobar como se cumplen las condiciones cuando se necesita gas.

Nota: Tener en cuenta que, por como se ha reestructurado asignar, si el recurso asignado no es un gas, el último parámetro de la acción no es relevante y se debe ignorar.

EJERCICIO 3

Para que un edificio pueda requerir más de un recurso se ha cambiado la precondición de que se tenga el tipo de recurso necesario para ese tipo de edificio de la acción construir por la de que para todo tipo de recursos, si lo requiere el tipo de edificio, se debe de tener disponible. Esto se ha hecho con las ordenes forall(para todo tipo de recurso) e imply(si el tipo de edificio requiere el tipo de recurso, hay que tenerlo). Así cumplimos tambien con la restricción de que no haya que especificar el recurso necesario para todos los edificios si no que solo es necesario para todos los tipos de edificios, ya que esa parte no se ha cambiado.

Como en el ejercicio anterior, tenemos el problema de que no se aprecia si la solución es correcta, por lo que se han dejado algunas líneas extra comentadas que basta con descomentar(y comentar el otro objetivo) para tener una situación en la que hay que construir otro centro de mando, donde se puede ver como se construye el extractor para gas, se asignan unidades a ambos recursos y se construye de manera correcta según las condiciones impuestas hasta el momento.

EJERCICIO 4

Para este ejercicio se han introducido en el dominio los dos tipos nuevos de unidad, marine y segador, y los predicados 'genera', que relaciona los tipos de edificio con los tipos de unidades que es capaz de crear, y 'necesita', que es el equivalente a 'requiere' para unidades en vez de para edificios. Además se introduce la acción 'reclutar', para la que hay que replantear la representación de unidades: ya no estarán todas las unidades inicializadas con su tipo y su posición inicial, si no que habrá objetos genéricos(excepto los que sean creados al inicio y no tengan que ser generados por edificios) que no tendrán asignado un tipo ni una localización hasta que no se generen con esta acción. Dicho esto, 'reclutar' asigna a una de estas entidades genéricas un tipo y una localización(lo cual es el equivalente a crear una unidad, ya que la introducimos en el grid) si existe un edificio construido cuyo tipo pueda generar el tipo de la unidad correspondiente, teniendo en cuenta que esta unidad no debe de estar aún inicializada y que se disponga de los recursos necesarios para la creación de la unidad correspondiente(se hace igual que con los recursos necesarios para construir un edificio, explicado anteriormente).

Tener en cuenta que plantear de esta manera las unidades no es lo más correcto ya que se depende del número máximo de unidades planteadas y probablemente haya otro planteamiento más correcto y elegante, aunque creo que para esta práctica es suficiente y más simple hacerlo de esta manera.

En el problema hay que modificar las unidades tal y como se ha dicho anteriormente, se inicializará la única unidad VCE que está desde el inicio y se añadirán 10 unidades genéricas(el número es arbitrario) sin inicializar. Adicionalmente se deben añadir los predicados iniciales para declarar qué recursos necesitan las unidades para crearse y qué edificios las pueden generar, además de eliminar todo lo referente a las unidades anteriores.

Se puede observar como se generan las unidades de forma correcta.

EJERCICIO 5

En este ejercicio se ha tenido que introducir en el dominio el nuevo tipo de edificio bahía de ingenieria y se ha decidido que las investigaciones sean constantes, ya que facilitará el trato que se les dará para investigarlas y para determinar los materiales necesarios para que se lleven a cabo. Por tanto se ha introducido la constante impulsor segador de tipo investigación.

Los predicados que controlarán las investigaciones serán dos, uno de ellos llamado ingrediente relacionará cada constante investigación con un recurso que necesite y el otro, investigado, informará que una investigación ya ha sido realizada.

En las acción reclutar se ha tenido que introducir una estructura or para que, si la unidad a reclutar es un segador, se necesite tener investigado impulsor segador, de manera que es fácil introducir nuevas restricciones de este tipo. Además se ha añadido la acción investigar que, si hay una bahia de ingeniería construida y tenemos todos los ingredientes necesarios para una investigación concreta, marque esta investigación como completada.

En el problema ha sido necesario añadir un objeto bahía de ingeniería y los recursos necesarios para este y para la investigación que, aunque no se detalla en el guión, se ha aclarado que son ambos minerales y gas. Tras ejecutarlo, se puede observar como cumple los objetivos propuestos.

EJERCICIO 6

En este ejercicio se va a cambiar el comportamiento del dominio para que, en lugar de tener o no tener un recurso, se entiendas los recursos por unidades, de manera que cada edificio, unidad o investigación tenga un coste. Para ello, es necesario añadir una acción recolectar para que cada unidad que esté asignada a un nodo recoja un número determinado de ese recurso(en nuestro caso 25). Al tener un límite, se debe comprobar antes que al añadir estos recursos no lo superemos.

Para controlar valores numéricos se han tenido que cambiar todos los predicados que indicaban los recursos necesarios para cada unidad, edificio o investigación y la cantidad de cada recurso que teníamos,que ahora pasarán a ser funciones inicializadas en el problema con los valores indicados en el enunciado. Además de nuevas funciones que controlarán el límite de almacenamiento y los trabajadores que tengamos asignados a cada recurso.

Para que todo funcione correctamente ha sido necesario modificar las acciones construir, reclutar e investigar, además de añadir desasignar para que los trabajadores puedan hacer otras cosas después de recolectar recursos. Para asignar y desasignar ha sido suficiente con eliminar de asignar el efecto de 'tenemos' un recurso, ya que ahora se encargará de ello recolectar, y sumar 1 al número de trabajadores asignados a ese recurso, mientras que desasignar tan solo deberá marcar la unidad como libre de nuevo y restar 1 al número de trabajadores asignados al recurso que estaba recolectando. En el resto de acciones se ha sustituido la condición de 'tenemos' el recurso necesario por la de que para todos los que necesitemos debamos tener un número igual o mayor al necesario, restando este número de nuestras pertenencias en los efectos.

De esta manera hemos adaptado el problema del ejercicio anterior al uso de variables numéricas. Por otro lado, al tener que controlar un límite de almacenamiento, se ha añadido el nuevo tipo de edificio depósito y tres edificios de ese tipo para que, aunque no sea necesario aumentar tanto el límite, se pueda llegar hasta un límite de 400.

Podemos ver en la solución como para construir los barracones necesarios para generar las unidades objetivo, primero construye un depósito, ya que estos necesitan 150 de mineral y el límite inicial es 100, así que el problema se resuelve correctamente.

Se ha elegido un factor de 25 para aumentar el recurso ya que es el que da el tiempo más bajo, unos 11s en mi máquina, ya que con otros conlleva varios minutos. Se ha intentando optimizar el código de las acciones para bajar más el tiempo pero, al menos como se ha afrontado el problema desde el inicio de este documento, ha sido imposible. En el siguiente ejercicio comentaré más sobre esto.

EJERCICIO 7

Para terminar se va a introducir una función tiempo que contará cuantas unidades de tiempo ocuparemos para llegar al objetivo. Para añadir todo lo comentado en el enunciado son necesarias funciones para establecer cuanto tiempo cuesta construir un tipo de edificio, generar un tipo de unidad o investigar, y para establecer la velocidad de cada tipo de unidad. Con estas funciones ya podemos modificar el resto de acciones para que, indicando que hay que minimizar la función tiempo con (:metric minimize (tiempo)).

Las modificaciones en construir, reclutar e investigar son simples, basta con añadir en efectos que incremente tiempo con el coste que representa la función correspondiente de las que se han comentado antes. También se ha impuesto en la acción reclutar un coste de tiempo de 5 unidades, elegido con el criterio de que sean algo proporcionales todos los costes en conjunto.

Basta inicializar en el problema tiempo a 0 y los consumos como se establece en la tabla del enunciado para que se resuelva el problema a falta de establecer coste en el movimiento.

La forma en la que el enunciado propone tratar la duración de la acción de navegar me parece un tanto incoherente: calcular la distancia para cada par de puntos conectados no tiene sentido, ya que desde cada punto solo se puede movar hasta otro con el que esté conectado, donde todas las posibilidades están a la misma distancia, una casilla. Por otro lado sería posible que en el ejercicio 1 se refiriera a que cada punto estuviera conectado con el resto de puntos o con toda la fila y la columna correspondiente, donde tendía sentido realizar este proceso de calcular distancias, pero se pide explícitamente que cada punto esté solamente conectado con las casillas adyacentes exceptuando esquinas. Es por ello que lejos de calcular todas estas distancias se ha normalizado la distancia de cada casilla a otra como 2 quedando guardada en la función consumeDist, de manera que cada vez que una unidad se mueve se suma a tiempo la división de la distancia entre casillas entre la velocidad de la unidad. La velocidad se ha establecido como 1 de base para VCE y Marine y 2 a Segador, ya que debe ser el doble, por lo que los primeros consumirán un tiempo de 2 al moverse, mientras que el segundo hara el movimiento en 1 unidad de tiempo.

La solución devuelta por *Metric-FF* llega a los objetivos y es correcta, por lo que se da el ejercicio como resuelto.