UNIVERSIDAD
DE GRANADA

Técnicas de Sistemas Inteligentes

grado en ingeniería informática

PRÁCTICA 3

Representación de dominios y resolución de problemas con técnicas de planificación

Autor
Fernando Vallecillos Ruiz

DNI
77558520J

E-Mail
nandovallec@correo.ugr.es

Grupo de prácticas
TSI 1 Lunes 17:30-19:30

Rama
Computación y Sistemas Inteligentes

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

1. Ejercicio 1

En este ejercicio se pide diseñar e implementar una versión inicial del dominio de planificación. Se sigue las indicaciones y se declaran los siguientes tipos de objetos: Unidades, Edificios, Localizaciones, Recursos. Además se representan como constante los tipos resultando en las clases tipoUnidades, tipoEdificios, tipoRecursos. También se declaran los siguientes predicados:

- {unidadTipo, edificioTipo, recursoTipo} ?obj ?tipo: Asigna tipo a un objeto
- {unidadEn, edificioEn, recursoEn} ?obj ?loc : Establece localización de unidad, edificio o recurso
- hayCamino ?loc1 ?loc2 : Existe camino entre dos localizaciones
- extraeLoc ?Unidad ?loc : La unidad esta extrayendo en la localización
- necesita ?tipoEdificio ?tipoRecurso : Para construir un tipo de edificio se necesita un tipo de recursos

Además se crean tres acciones diferentes:

- Navegar: Mueve una unidad en concreto entre dos localizaciones. Para ello se comprueba que existe un camino entre ambas localizaciones y que la unidad se encuentra en la primera posición. Además se ha de comprobar la unidad no este extrayendo ningún recurso. Tras esto, simplemente se elimina el predicado indicando su posición anterior y se crea uno indicando su nueva posición.
- Asignar: Asigna un trabajador a un nodo de recurso. Primero se comprueba que la unidad sea del tipo VCE (la única que puede extraer). Además se debe de comprobar que la unidad se encuentre en la misma posición que un nodo de recurso. Por último, comprobar que dicha unidad no esté extrayendo previamente (no se puede asignar dos veces). Como resultado simplemente creamos un predicado indicando que la unidad se encuentra extrayendo en la localización indicada.
- Construir: Ordena a un trabajador construir un edificio en una localización. De nuevo, se comprueba que la unidad sea del tipo VCE (la única que puede construir). Se comprueba que la unidad se encuentre en la localización donde se quiere construir el edificio. Esta unidad no puede estar extrayendo nada previamente. Se comprueba que no exista un edificio previamente en dicha localización (límite de un edificio). Por último se necesita comprobar que existen los materiales necesarios. En este caso, los edificios necesitan un solo

material. Por ello basta comprobar que exista un trabajador distinto al actual el cual se encuentre extrayendo el material que se necesita para el tipo de edificio que se quiere construir.

Se pasa entonces al fichero del problema, donde se definen los diferentes objetos declarados inicialmente. Se declaran 25 objetos de tipo Localización para indicar un mapa de 5x5. Se siguen las indicaciones y se crea un centro de mando, tres unidades, 5 nodos de recurso y un barracón. Tras esto se inicializan los predicados. Se inicializan los tipos de todos los objetos asociando los Edificios a tipoEdificio, cada unidad a tipoUnidad y cada recurso a tipoRecurso. Se añade el recurso necesario para construir cada tipo diferente de edificio. También se asocian las diferentes localizaciones para permitir navegar entre vecinas. Con esto se habría creado el mapa y los objetos a colocar. Se debe entonces especificar la localización de los objetos (unidades, edificios, nodos de recurso) en el mapa. De forma arbitraria se elige una localización para el centro de mando con las tres unidades. También se eligen diferentes posiciones para los 5 nodos de recursos. Por último, se establece la meta de construir un barracón.

2. Ejercicio 2

Se añade la restricción de haber un extractor de gas para poder extraer este recurso. Con respecto a la declaración de tipos, se añade este tipo de edificio a tipoEdificios (constante). Solo esto ya permite construir este tipo de edificio en el mapa. Ahora se necesita implementar la restricción en la acción Asignar. Simplemente se añade esta condición a la acción. Es decir, si el nodo de recurso que se va a asignar es de tipo Gas, se comprueba que exista un edificio de tipo ExtractorGas en la misma localización. En el archivo de problema se añaden los datos necesarios respecto al nuevo edificio. Se crea un nuevo objeto de tipo edifico y se le asigna el tipo ExtractorGas (no se coloca en ninguna posición puesto que no está construido). Se añade también que para construir este tipo de edificios se necesita estar extrayendo Mineral.

3. Ejercicio 3

En este ejercicio se permite que los edificios puedan tener 1 o más recursos necesarios para construirlos. Como se ha podido ver, los predicados asocian los materiales necesarios a un tipo de edificio por lo que solo existirá un predicado de este tipo independientemente del número de edificios que se puedan construir de cada tipo. Anteriormente se utilizaba 'exists' para comprobar que el recurso nece-

sario existía. Ya que ahora puede haber más de un recurso necesario esta fórmula pasará a 'forall'. Esto nos permite iterar sobre todos los tipos de recursos y, si el tipo de edificio los necesita, comprobar que haya otro trabajador extrayéndolos. El archivo de problema se mantiene igual. Simplemente se añade un nuevo predicado indicando que el centro de mando necesita gas (previamente solo mineral).

4. Ejercicio 4

En este ejercicio se añade la acción de Reclutar. Esta acción permite crear una nueva unidad si se posee el edificio adecuado y los recursos necesarios. Primero se necesitan añadir los nuevos tipos de unidades en forma de constantes (Marines y Segadores). Luego se añaden dos nuevos predicados:

- puedeReclutar ?tipoEdificio ?tipoUnidad
- necesitaUnidad ?tipoUnidad ?tipoRecruso

Para la nueva acción de Reclutar se puede aplicar una estructura similar al construir. Primero nos aseguramos de que unidad en concreto se quiere crear y el tipo de este. Se comprueba que no ha sido reclutada anteriormente. Luego se pasa a identificar el edificio necesario para crear este tipo de unidad y su localización. Tras esto se aplica un bucle 'forall' igual que al construir para comprobar que se disponen de los recursos necesarios para crear el tipo de unidad. Como resultado se çrea"la unidad situándola en la misma posición que en el edificio donde ha sido reclutada. En el fichero de problema se realizan una serie de cambios para incluir las nuevas condiciones. Primero se tienen que crear las tres nuevas unidades. Luego se comienzan a inicializar los predicados donde se le asigna el tipo a cada una de estas unidades. Se crean los predicados nuevos (descritos anteriormente) para indicar que edificios pueden reclutar que tipo de unidades y que recursos consumen estas. El ejercicio también establece que se comience con un solo trabajador (no con tres) por lo que se eliminan los predicados que crean estos dos últimos trabajadores (aunque es posible reclutarlos). Por último se establece el nuevo objetivo: tener en una localización un marine y en otra un marine y un segador.

5. Ejercicio 5

En este ejercicio se añade la nueva acción Investigar. Esta acción desbloquea la capacidad de reclutar nuevas unidades que hasta entonces estarán bloqueadas. Para investigar se necesita haber construido un edificio de tipo Bahía de Ingeniería.

En nuestro caso solo tendremos una investigación que desbloquea a los Segadores. Se añade el tipo de edificio Bahía de Ingeniería a las constantes. También se añaden los siguientes predicados:

- faltaInvestigar ?tipoUnidad
- necesitaInvestigar ?tipoUnidad ?tipoRecurso

De nuevo la acción Investigar tomará una estructura de precondiciones similar a anteriores. Como se ha explicado lo primero será comprobar que existe un edificio de tipo Bahía de Ingeniería construido. Se comprueba con el predicado faltaInvestigar, que aún no se ha investigado dicha unidad. Este predicado también sirve como bloqueo en el reclutamiento de unidades por lo que se comprueba que no exista para cada tipo de unidad en la acción de Reclutar. Por lo que mientras exista, no se podrán reclutar las unidades especificadas. Tras esto, solo se necesitan comprobar que existen los recursos necesarios para la investigación. Esto se consigue con un bucle 'forall' de la misma forma que en las acciones Reclutar o Construir. Como resultado de esta acción se elimina el predicado faltaInvestigar. Como se ha explicado, esto desbloquea la posibilidad de reclutar unidades de dicho tipo. En el archivo de problema se debe declarar un nuevo edificio Bahía construible y asociarlo a su tipo BahiaIngenieria. Se añaden los recursos necesarios para construir este edificio y para realizar la investigación.

6. Ejercicio 6

En este ejercicio se producen cambios muy grandes con respecto al anterior. Se ira describiendo poco a poco el proceso de transformación y optimización. En este ejercicio se añaden cuantificadores a la cantidad de recursos disponibles. Ya no serán infinitos. Se añade una acción Recolectar la cual añadirá 10 unidades de un tipo de recurso (Mineral o Gas) por cada trabajador que esté extrayéndolo en cualquiera de los nodos. Las acciones de Construir, Reclutar o Investigar llevarán consigo un coste dependiendo de los parámetros (qué es lo que se quiera construir, reclutar...). Se permite desasignar un trabajador de un nodo mediante una nueva acción. También se añade una capacidad máxima de recursos que se puedan almacenar y esta podrá ser aumentada con la construcción de un nuevo tipo de edificio, el Depósito.

Se comienza explicando los cambios necesarios para realizar la cuantificación. Para esto se crean dos predicados numéricos:

- Reserva ?tipoRecurso

- LimiteReserva

El primer predicado guarda la cantidad dispuesta de cada tipo de recurso. El segundo guarda el máximo de cantidad que se puede guardar de cada recurso. Este límite superior es compartido por todos los recursos así que solo hace falta de disponer un solo predicado estableciéndolo. Con esto ya se podría guardar las cantidades necesarias de cada recurso en cada momento. Pasamos ahora a las modificaciones de las acciones ya existentes.

Las acciones de Construir, Reclutar e Investigar se ven modificadas para aplicar las restricciones de recursos. En un principio se creó predicados asociando el coste de cada tipo específico de acción. Esto permitía mantener una estructura similar a la ya implementada anteriormente. Recorriendo todos los tipos de recursos y, en el caso de que se necesitasen, realizar la comprobación (en este caso de que hubiera suficiente). Es una opción viable si se tiene un conjunto grande de predicados, sin embargo, no es nuestro caso. Para agilizar el programa, se 'desenrolla' este bucle. Es decir, se escriben las condiciones que tiene que cumplir para cada uno de los diferentes casos. Por cada acción se han desenrollado dos bucles: precondiciones y efectos. En las precondiciones se comprueba que para cada caso de la acción (tipo de edificio, tipo de unidad o investigación) haya los recursos necesarios. De esta forma, en los efectos ya se ha comprobado que hay los recursos necesarios y por tanto solo hace falta restarlos de la reserva. También se debe mencionar el aumento de capacidad. Si en la acción Construir, se ha construido un edificio de tipo Depósito, provoca un aumento de 100 unidades a LimiteReserva.

Se pasa entonces a las nuevas acciones. La acción Desasignar permite desasignar un trabajador de un nodo en el que estaba extrayendo. Esta acción simplemente borra el predicado extraeLoc. Por eso necesita saber qué trabajador es y dónde estaba extrayendo. La nueva acción Recoletar nos permite añadir recursos a las reservas. En un principio se impusieron dos condiciones que, aunque no fueran obligatorias, podrían descartar situaciones donde no tendría sentido ejecutar la acción. La primera sería que existiese al menos un trabajador que estuviera extrayendo el tipo de recurso que se quiera recolectar. También se imponía que la Reserva no estuviese a su límite (al menos un trabajador pudiese añadir a las reservas). En los efectos se itera sobre todas las unidades y, si la unidad se encontraba extrayendo el recurso a recolectar, se añadía 10 a las reservas. Este bucle provoca que pueda haber un exceso de recursos por lo que luego, se establece que si se ha superado el límite de recursos en la reserva, se quede en el límite de estos. Es decir, la reserva de recursos será como mucho igual al valor de LimiteReserva en cualquier momento. Estos condiciones para aplicar los efectos provocan que las precondiciones no fueran necesarias pero si recomendables. Si no existiese ningún trabajador extrayendo, el efecto no sumaría nada. Si la reserva ya estuviese al límite, tampoco cambiaría luego de la acción.

Por último, se añade Depósito como nuevo tipo de edificio y en el archivo de problema se asocia a un nuevo edificio. Se añade un nuevo edificio y se asocia con Extractor Gas en caso de que haga falta. Ya que al no tener recursos infinitos puede ser necesario. Se añaden los predicados numéricos descritos anteriormente estableciendo un límite inicial de 100 y la cantidad de recursos iniciales a cero. Como se ha comentado, en un principio se añadían predicados para establecer los costes de los diferentes casos de las acciones. Pero se eliminan cuando se desenrollan los bucles.

6.1. Optimización

Los cambios hasta ahora son suficiente para poder encontrar un plan de acción. Sin embargo, el tiempo que toma en esto en encontrarlo es igual de importante. Al comenzar la optimización de este ejercicio, solo se podía encontrar un plan al aumentar el incremento resultante de la acción Recolectar a 50. Por ello se decide buscar como es posible reducir el tiempo tomado. Se puede ver que en el código que en bastante zonas se utilizan 'exists' y 'forall' para comprobar condiciones. Aunque son formas válidas de aplicar una precondición, no son eficientes. Primero se explicará de forma general como se han transformado estas precondiciones y luego se darán los ejemplos concretos.

La transformación es bastante simple. La fórmula 'exists' se usa para comprobar que si hay una cláusula creada que cumpla alguna condición. En los casos anteriores, se ha usado sobre todo para comprobar que no podamos reclutar o construir dos veces con los mismos parámetros. Es decir, aunque se permita reclutar o construir objetos del mismo tipo, no se puede admitir repetirlo con el mismo objeto. Esto se puede arreglar creando nuevas cláusulas asociadas al objeto las cuales especifican si ya ha sido creado o no. Aunque 'exists' también se usaba en los bucles 'forall', desaparece al desenrollarlos. Se pasa ahora a los casos concretos para cada una de las acciones.

Para la acción Asignar se crea un predicado:

- hayExtractor ?loc

Este predicado permite ver si un extractor ha sido construido en una localización. Si el nodo de la localización es de Gas, simplemente se buscará por este predicado en vez de comprobar si existe un edificio de tipo extractor en la localización. Este predicado se crea al construir un edificio de tipo extractor.

Para la acción Recolectar se crea un predicado:

- extraeRecurso ?unidad ?tipoRecurso

La acción de recolectar no necesita saber la localización del recurso, solo cuántos trabajadores se encuentran extrayéndolo. Este predicado nos permite iterar sobre las unidades y comprobar si están extrayendo el recurso en vez de comprobar si existe un nodo del tipo de recurso a recolectar el cual extraen. Este predicado se crea al asignar un nodo a un trabajador y se elimina al desasignarlo.

Para la acción Construir se crean dos predicados:

- hayEdificio ?loc
- construido ?edificio

La optimización de esta acción trajo los mejores resultados. Ya se hablo de los bucles en torno a los recursos para construir un edificio. Sin embargo, existen dos otras precondiciones importantes. Solo puede haber un edificio en cada localización. Y un mismo edificio no puede ser construido dos veces. La primera condición se implementa gracias a hayEdificio. De esta forma se comprueba solamente si ya existe otro edificio en la localización y, por tanto, no hace falta averiguar que edificio es. La segunda condición se implementa gracias a construido. Como se ha descrito anteriormente, permite saber si un edificio ya ha sido construido anteriormente (no puede existir en dos posiciones al mismo tiempo). Ambos predicados también son creados en la acción Construir (o en la inicialización).

Para la acción Reclutar se crean dos predicados:

- reclutada ?unidad
- puedeReclutarEn ?tipoUnidad ?loc

De nuevo, se optimizaron los bucles en torno a los recursos para reclutar una unidad. También existen otras dos precondiciones importantes. Cada unidad solo puede ser reclutada una vez. Y se necesita haber construido el edificio necesario para poder reclutar cada tipo de unidad. De forma similar al anterior ejemplo, la primera condición se implementa con reclutada. De esta forma se permite saber si una unidad en concreto ya ha sido reclutada anteriormente. La segunda condición se relaciona con los tipos de edificios necesarios para crear una unidad. En esencia, para reclutar no necesitamos saber que tipo de edificios pueden reclutar que unidades. Solo se necesita saber dónde se puede reclutar cada tipo de unidad. De esta forma, al crear un Barracón o Centro de Mando, se crean estos predicados indicando que se pueden reclutar ciertos tipos de unidades en esa localización. El predicado reclutada se crea en la propia acción Reclutar (o en la inicialización).

Para la acción Investigar se crea un predicado:

- puedeInvestigar

Aunque solo se dispone de una investigación, se ha 'desenrollado' dicho bucle de recursos de la misma forma que los anteriores. Para poder investigar, se necesita haber creado una Bahía de Ingeniería. En vez de comprobar si existe un edificio de este tipo, simplemente se crea un predicado indicando que se puede investigar. Dado que no existe diferencias entre Bahías se puede asumir que solo se necesita crear un predicado de este tipo. Este predicado se creará al construir una Bahía de Ingeniería.

De esta forma se termina de optimizar alcanzando los resultados de tiempo requeridos ¡2s y cumpliendo todas las condiciones. Se ha querido dejar los parámetros por defecto pero puede surgir que el sistema tarde mas de 2s en encontrar un plan si se cambia las posiciones de los nodos de recursos. Si se aumenta la tasa de recolección (10 por defecto) a 15 o 20 el sistema encuentra el plan en una fracción del tiempo original.