DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Técnicas de los Sistemas Inteligentes.
Curso 2018-19.
Práctica 2: Planificación Clásica

Relación de Ejercicios 1
Dominios y problemas de planificación clásica en PDDL.

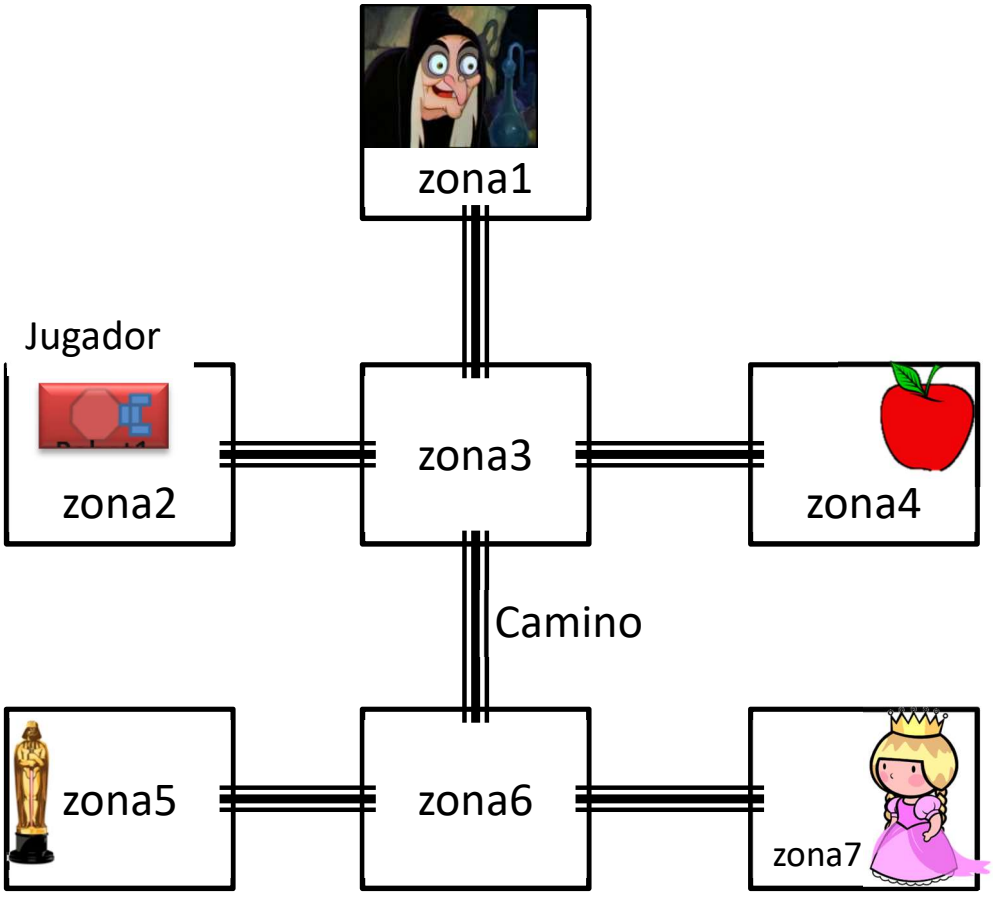
Objetivo

El objetivo de estos ejercicios es experimentar con el lenguaje PDDL y con el planificador Metric-FF explicados en clase. Tener en cuenta que hay que comprobar en todos los ejercicios que los ficheros PDDL de dominio y problema son correctos, mediante la ejecución del planificador y la obtención de un plan válido que resuelva uno o varios problemas. La corrección de cada ejercicio consistirá en ejecutar Metric-FF tomando como entrada el dominio escrito en cada ejercicio y los problemas entregados por el alumno. El profesor de prácticas puede plantear en la corrección un conjunto de problemas distintos definidos por él mismo.

Ejercicios

Los ejercicios a realizar están inspirados en un mundo de aventuras gráficas, ideado por Raúl Pérez llamado ***Los extraños mundos de Belkan.*** El juego en su versión real se desarrolla en un mapa bidimensional discreto en el que un agente puede exhibir comportamientos reactivos o deliberativos y se ha utilizado este curso académico para el desarrollo de las prácticas de la asignatura de Inteligencia Artificial de 2º curso. Para esta práctica de TSI nos centraremos solo en comportamientos deliberativos, en concreto comportamientos definidos mediante un dominio de planificación y especificados como planes de acciones que tiene que llevar a cabo un jugador para poder cumplir misiones en el juego. No utilizaremos el juego implementado

en su entorno real, pero partiremos de una simplificación de los mundos de Belkan, en la que el mundo está formado por zonas cuadradas que están unidas por caminos, como se observa en la siguiente figura.



En el mundo pueden cohabitar 5 tipos personajes (*Princesa, Príncipe, Bruja, Profesor y Leonardo di Caprio*) y pueden existir varios **objetos de distintos tipos**: *óscars (de los de Hollywood), manzanas, rosas, algoritmos y oro*. También existe un agente jugador (que puede considerarse como un robot dotado de una mano) cuyas misiones en esta práctica consisten en entregar objetos (inicialmente distribuidos por zonas del mundo) a personajes (también localizados en distintas zonas). A continuación se proponen varios ejercicios que irán aumentando su nivel de complejidad para poder construir finalmente un dominio de planificación que pueda resolver problemas de planificación que permitan a un agente llevar a cabo un comportamiento deliberativo en un entorno de juegos.

Ejercicio 1. Definir un dominio y problema de planificación considerando que el jugador podrá estar orientado al norte, sur, este u oeste y desplazarse de una zona a otra siempre que esté correctamente orientado. Por ejemplo, podrá desplazarse a una zona al norte de su zona actual, si está orientado al norte. Realizar las siguientes tareas:

a. Representar en el dominio los objetos del mundo (jugador, personajes, tipos de objetos y las zonas del mundo). Hacerlo de la forma más general posible, considerando siempre que puede haber más de un objeto de cada tipo (por ejemplo, puede haber varias princesas o varios "óscars" en el mundo).
b. Representar predicados que permitan describir los estados del mundo, mediante la especificación de aspectos como la relación de conexión entre zonas (representando no solo la relación de conexión, sino también que una zona está al norte/sur/este/oeste de otra), la orientación del jugador, las posiciones de los objetos y cualquier otra relación o propiedad que sea necesaria para la correcta definición de las acciones que puede realizar el jugador.
c. Representar las siguientes acciones del jugador:
    i. girar a la izquierda, girar a la derecha, ir (de una zona a otra correctamente orientado), coger (un objeto), dejar (un objeto) y entregar (un objeto a un personaje).
d. Plantear un problema de planificación con un estado inicial con 25 zonas conectadas arbitrariamente en el que aparezcan situados los 5 personajes en distintas zonas y al menos 5 objetos. El objetivo de este problema consistirá en conseguir que todos los personajes tengan al menos un objeto. Comprobar con Metric-FF que se obtiene un plan para conseguir esta misión. Para escribir el fichero de problema (especialmente las relaciones de conectividad entre zonas) es recomendable utilizar las utilidades que proporciona el editor http://editor.planning.domains.
e. Desarrollar un generador de problemas para este dominio, implementando un programa en c++ o en Python que reciba como entrada un mapa en formato texto y proporcione una representación de un problema utilizando los predicados definidos en este ejercicio. El fichero de texto tiene que contener la información en el siguiente formato:
    i. En la primera línea: información sobre el número de zonas. numero de zonas:<valor>
    ii. En el resto de líneas se indica la relación de conectividad, entre las zonas de manera que en cada línea se describe una lista de zonas adyacentes, asumiendo siempre la relación simétrica entre ellas.
    iii. Cada zona está representada por una constante (z1, z2, etc.)

iv. Cada zona está concatenada con una lista de objetos que puede haber en esa zona, delimitada por corchetes. Si no hay objetos se representará como lista vacía []. El separador de elementos de la lista es el carácter espacio en blanco.
v. Cada objeto en la zona se representa con un par <objeto>-<Tipo>
vi. Un ejemplo de un fichero de mapa sería el siguiente

```
numero de zonas:7


V -> z1[bruja1-Bruja] z3[] z6[]

H -> z2[player1-Player] z3[] z4[manzana1-Manzana]

H -> z5[oscar1-Oscar] z6[] z7[princesa1-Princesa]
```

Ejercicio 2. Una vez comprobado que el dominio descrito es correcto, considerar que la acción de desplazamiento entre zonas tiene un coste igual a la longitud del camino entre cada zona.

a. Modificar el dominio del anterior ejercicio para adecuarlo a esta nueva característica.
b. Extender el problema definido en el anterior ejercicio, definiendo distancias entre zonas y comprobar que el planificador obtiene un plan para estas nuevas restricciones.
c. Extender el script de generación de problemas de manera que se pueda representar la relación de conectividad con peso entre las zonas. De la siguiente forma:
    i. Entre cada dos zonas conectadas se usará la cadena "=<distancia>=" para representar el coste del arco que conecta ambas zonas.
    ii. Un ejemplo de un fichero de mapa sería el siguiente

```
numero de zonas:7


V -> z1[bruja1-Bruja]=10=z3[]=5=z6[]

H -> z2[player1-Player]=10=z3[]=5=z4[manzana1-Manzana]

H -> z5[oscar1-Oscar]=10=z6[]=5=z7[princesa1-Princesa]
```

Ejercicio 3. Considerar ahora que (1) hay distintos tipos de zonas dependiendo del tipo de superficie que contengan, en concreto: *Bosque, Agua, Precipicio, Arena y Piedra*, y (2) hay dos nuevos tipos de objetos: *Zapatilla y Bikini.* Además, considerar también que el jugador, aparte de poder tener cogido un objeto, está dotado de una mochila donde puede guardar otro objeto (solo uno). Realizar lo siguiente:

a. Modificar el dominio del ejercicio anterior para que el jugador pueda desplazarse por el entorno considerando las siguientes restricciones:
    i. Puede moverse a una zona de bosque sólo si tiene una zapatilla (cogida o en la mochila). (Puede definirse un predicado que permita determinar de qué tipo es un determinado objeto).
    ii. Puede moverse a una zona de agua si tiene un bikini (cogido o en la mochila).
    iii. No puede moverse a un precipicio.
b. Modificar el domino para que pueda meter y sacar objetos en/de la mochila. Tener en cuenta que para meter en la mochila lo tiene que tener cogido, solo puede tener cogido un objeto a la vez y uno en la mochila.
c. Extender el problema del anterior ejercicio para poder representar un escenario que contenga zonas de los distintos tipos descritos y también objetos de tipo zapatilla y bikini para que el jugador pueda moverse por todas las zonas. Representar al menos que hay una zapatilla en una zona de agua, o un bikini en una zona de bosque.
d. Extender el script de generación de problemas de manera que se pueda representar el tipo de cada zona. De la siguiente forma:
    i. Cada zona aparecerá concatenada con una nueva lista con un solo elemento que podrá ser uno de los valores: *Bosque, Agua, Precipicio, Arena y Piedra* Un ejemplo de un fichero de mapa sería el siguiente

```
numero de zonas:7


V -> z1[bruja1-Bruja][Bosque]=10=z3[][Arena]=5=z6[][Piedra]

H -> z2[player1-Player][Bosque]=10=z3[][Arena]=5=z4[manzana1-Manzana][Piedra]

H -> z5[oscar1-Oscar][Bosque]=10=z6[][Bosque]=5=z7[princesa1-Princesa][Agua]
```

Ejercicio 4. Considerar ahora que cuando el jugador entrega objetos a un personaje consigue puntos, según la siguiente tabla:

	Leonardo	Princesa	Bruja	Profesor	Príncipe
Oscar	10	5	4	3	1
Rosa	1	10	5	4	3
Manzana	3	1	10	5	4
Algoritmo	4	3	1	10	5
Oro	5	4	3	1	10

a. Modificar el dominio para poder registrar los puntos acumulados por el agente, mediante una función, cada vez que entrega un objeto a un personaje (una función PDDL puede tener dos argumentos).
b. Extender el problema del anterior ejercicio para que, por un lado, se pueda representar la tabla anterior y, además, partiendo de 0 puntos el jugador pueda alcanzar al menos una cantidad arbitraria de puntos (indicándolo en el objetivo), sin especificar a qué personajes hay que entregar objetos. Por ejemplo, plantear un problema en el que con solo objetos de tipo oscar y manzana el jugador pueda alcanzar 50 puntos (asumiendo que solo están los 5 personajes).
c. Extender el script de generación de problemas para incorporar los puntos mínimos que se deben alcanzar. Para ello, se indicará en una línea "puntos_totales:X" siendo X, el número de puntos a alcanzar.

```
numero de zonas:7

puntos_totales:50

V -> z1[bruja1-Bruja][Bosque]=10=z3[][Arena]=5=z6[][Piedra]

H -> z2[player1-Player][Bosque]=10=z3[][Arena]=5=z4[manzana1-Manzana][Piedra]

H -> z5[oscar1-Oscar][Bosque]=10=z6[][Bosque]=5=z7[princesa1-Princesa][Agua]
```