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 2: Dominios y problemas de planificación clásica en PDDL.

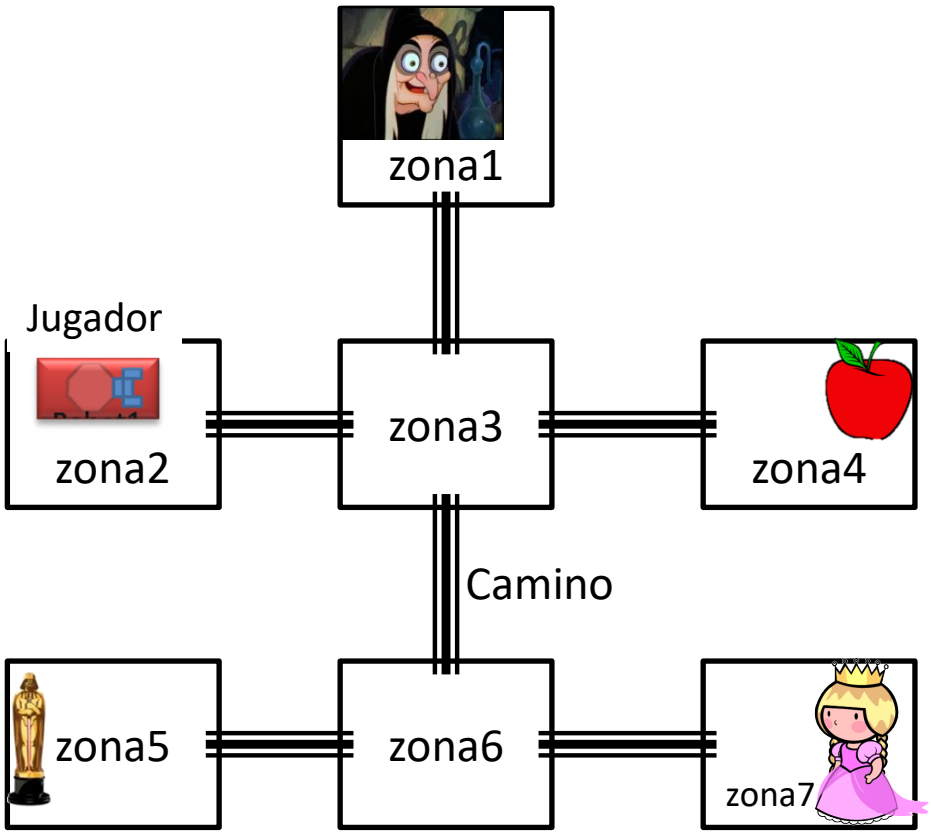
Objetivo

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, los problemas entregados por el alumno y un conjunto de problemas distintos definidos por el profesor.

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. 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 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 5. Teniendo en cuenta las características del jugador de la anterior relación, considerar ahora que cada personaje tiene un bolsillo mágico que admite recibir un máximo número de objetos (configurable en el estado inicial).

a. Modificar las acciones, predicados y funciones necesarios para adecuarse a la nueva característica de los personajes.
b. Extender el problema del anterior ejercicio para que el planificador pueda encontrar un plan con estas nuevas condiciones. Comprobar experimentalmente mediante la propuesta de varios problemas que el nuevo dominio es correcto y que puede resolver problemas en distintas situaciones.
c. Extender el script de generación de problemas de manera que se pueda representar el número de objetos admitido por cada personaje. Se representará en una línea comenzando con la palabra clave "bolsillo" seguida de":" y una lista de elementos separados por espacios, donde cada elemento tendrá la forma <nombre de objeto>:<número de objetos>. Por ejemplo:

```
numero de zonas:7

bolsillo:[bruja1:4 princesa1:5]

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 6. Considerar ahora que hay dos jugadores cooperantes, con las mismas características que en el último ejercicio, y considerar el siguiente escenario:

a. Entre los dos tienen que obtener una cantidad de puntos dada (en el estado inicial) y cada uno tiene que entregar objetos para obtener un mínimo de puntos.
b. Modificar las acciones, predicados y funciones necesario para adecuar el dominio a las nuevas características. Plantear varios problemas en los que se muestre que el dominio es correcto y puede usarse para resolver problemas en distintas situaciones.
c. Extender el script de generación de problemas para poder expresar esta nueva propiedad en el problema. En una línea se indicará el número de puntos a obtener con "puntos_totales:<número>", y en otra línea los puntos de cada jugador, con el formato "puntos_jugador:[<nombrejugador>:<puntosminimos> <nombrejugador>:<puntosminimos> …]"

```
numero de zonas:7

bolsillo:[bruja1:4 princesa1:5]

puntos_totales:50

puntos_jugador:[player1:20 player2:20]

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 7. Considerar un escenario nuevo, en el que los robots son de distinto tipo y cada uno tiene distintas capacidades: uno solo puede recoger objetos (y entregárselos a otro robot) y otro solo puede entregar objetos a personajes (y recibirlos del otro robot cooperante).

a. Modificar las acciones, predicados y funciones necesario para adecuar el dominio a las nuevas características.
b. Comprobar experimentalmente con varios problemas que el dominio diseñado es correcto y puede resolver problemas en distintas situaciones.
c. Extender el generador de problemas para que pueda admitir distintos tipos de “players”, de igual forma a como se hizo en el Ejercicio 1 con el resto de tipos de objetos. Se indica el número de jugadores con la línea “numero jugadores: x”. Asumir que el campo “puntos_jugador” solo se aplica a jugadores que pueden entregar objetos a personajes. El resto de los jugadores hasta completar el número de jugadores indicado, son los encargados de coger objetos.

```
numero de zonas:7

numero de jugadores: 2

bolsillo:[bruja1:4 princesa1:5]

puntos_totales:50

puntos_jugador:[player1:20]

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]
```

Material a entregar

Entregar en una carpeta comprimida los siguientes ficheros, con el siguiente contenido

- **Una carpeta de nombre "Ejercicios"** de manera que por cada ejercicio de la relación 1 y 2 contenga
    a. Un fichero de dominio con nombre EjXdominio.pddl.
    b. Al menos un fichero de problema con nombre EjXproblema.pddl. Si se entregan varios problemas por cada ejercicio (lo cual se considerará muy positivo para la calificación), numerar los problemas: EjXproblema1.pddl, EjXproblema2.pddl, etc.
    c. Una carpeta adicional por cada Ejercicio con el nombre "EjXsoluciones" que contenga archivos correspondientes a la salida en texto producida por ff como resultado final de su ejecución (usando la redirección de salida >> de Linux). La relación de archivos en esta carpeta cada archivo en esta carpeta se llamará "Plan_problema<i>. txt" haciendo referencia al plan generado para cada problema propuesto.
- **Un fichero memoria.pdf** en el que se describan las principales decisiones en el diseño del dominio y por qué se han tomado. Hacer la descripción indicando el ejercicio y apartado, por ejemplo:
    a. Ejercicio1.a: <descripción de qué decisiones se han tomado y por qué..>
    Ejercicio1.b:<descripción…..>
    etc…
    b. En general, para cada ejercicio hay que **explicar las decisiones de diseño** tomadas para representar los objetos, propiedades y relaciones de objetos así como las acciones. Explicar **qué problemas se han utilizado** para comprobar que cada dominio es correcto, **indicando las características de cada problema** (número de objetos, tipos, valores iniciales de funciones, etc.). Puede añadirse como material adicional a la memoria tantos ficheros pddl de problema como se desee, para justificar que se ha experimentado con distintos problemas. Este aspecto se valorará positivamente en la evaluación.
- **Una carpeta de nombre "Generador"** que contenga el código fuente necesario para compilar (si está escrito en C++) y ejecutar el programa que genera problemas pddl a partir de la descripción de texto que se ha propuesto en cada ejercicio El generador programado tiene que ser llamado desde la línea de comando con la sintaxis

  `generaproblema <arg1> <arg2>`

  donde <arg1> es el path del fichero de texto que contiene la descripción del estado inicial y <arg2> es el path del fichero pddl de salida del generador

Calificación

La entrega se calificará de 0 a 10. Los ejercicios 1 a 5 valen 1,6 puntos cada uno. Los ejercicios 6 y 7 valen 1 punto cada uno. Tener en cuenta que se considerarán criterios de calidad de una buena representación del conocimiento un número reducido de acciones y predicados en el dominio, lo cual significa que las acciones representadas son lo suficientemente generales. Observar también que se valorará especialmente la experimentación realizada en cada ejercicio, por lo que el desarrollo del generador de problemas es esencial para una buena evaluación de la práctica. En concreto, los profesores utilizarán varios problemas definidos a partir de ficheros de texto y usarán el generador entregado por el alumno para evaluar las capacidades de los dominios entregados.

Fecha de Entrega

Hasta el 13 de Mayo de 2019,a las 14:00