Inteligencia Artificial

Examen Práctico - Grupo 1

7 de Febrero, 2019

Nombre y Apellidos:

En los ejercicios siguientes consideraremos la notación PDDL sin variables. Los átomos se representarán como cadenas que no comiencen por el símbolo "-" (p.e.: P2, Q3, R7...). La negación de un átomo será un átomo con el símbolo "-" delante (p.e.: -P6, -R3,...) Un literal es ún átomo o un átomo negado. Un estado es una lista (sin duplicados) de átomos y un objetivo una lista (sin duplicados) de literales.

Codificaremos un conjunto de acciones como un diccionario de pares clave-valor donde las claves son los nombres de las acciones y los valores son tuplas de 2 tuplas. La primera tupla son los prerrequisitos y la segunda tupla los efectos. De este modo, si consideramos el conjunto de acciones siguiente:

AQUI LOS EJEMPLOS

Ejercicio 1 (0.5 ptos):

Define la función resultado(A,C,E) donde C es un conjunto de acciones, A es el nombre de una de las acciones de C y E es un estado. La función debe devolver el estado resultante de aplicar la acción A al estado E. En esta función se usará más adelante como función auxiliar y en el momento de aplicarla supondremos que ya hemos comprobado que A es aplicable, por lo que en este ejercicio no se pide comprobar la aplicabilidad. Es importante que el estado obtenido no contenga duplicados.

In [ ]:
```
```

In [ ]:
```
resultado('E',C1,['P1','P2','P6'])
# Salida esperada: ['P6', 'P5']
```

In [ ]:
```
resultado('G',C2,['Q1','Q2'])
# Salida esperada: ['Q2', 'Q3']
```

In [ ]:
```
resultado('D',C3,['R1','R2','R4'])
# Salida esperada: ['R1', 'R2', 'R7']
```

Ejercicio 2 (1 pto.):

Escribir una función aplicables(e,C) que tome como entada un estado e y un devuelva la lista de acciones aplicables de C

In [ ]:
```
```

In [ ]:
```
aplicables(['P1','P4','P5','P6'],C1)
# Salida esperada: ['C', 'E']
```

In [ ]:
```
aplicables(['Q1','Q2','Q3','Q4'],C2)
# Salida esperada: ['A', 'B']
```

In [ ]:
```
aplicables(['R2','R4','R5','R6'],C3)
# Salida esperada: ['C']
```

Ejercicio 3 (1 pto):

Escribir una función deltacero(E,G,C) que tome como entada un estado E y un objetivo G que sólo tenga literales positivos y devuelva el valor de la heurística $\Delta_0(E,G)$ usando C como conjunto de acciones.

Nota 1: Para el infinito debes importar el módulo $math$. Infinito se representa como $math.inf$

Nota 2: Se pide la implementación de Δ_0 vista en clase (que es una versión simplificada de una definición general).

In [ ]:

In [ ]:
```
deltacero({'P1','P2'},{'P7','P8'},C1)
# Salida esperada: 11
```

In [ ]:
```
deltacero({'Q1','Q5','Q3'},{'Q9','Q6'},C2)
# Salida esperada: 2
```

In [ ]:
```
deltacero({'R9'},{'R16'},C3)
# Salida esperada: 3
```

Ejercicio 4:

El uso de la heurística Δ_0 en la búsqueda hacia adelante consiste en asignar a cada una de las acciones aplicables un valor h y elegir la acción con valor h más pequeño. La forma de asignar ese valor es la siguiente (ver teoría): _A cada acción A aplicable al estado actual se le asigna el valor heurístico $\Delta_0(e,g+)$, donde $g+$ es el conjunto de literales positivos del objetivo y e el estado que resulta al aplicar la acción A_.

En este ejercicio se pide definir la función adelante(E,G,C), donde E es el estado actual, G es un objetivo y C un conjunto de acciones. La función debe devolver la acción aplicable al estado actual E (entre las acciones del conjunto C) que tenga el menor valor h según el cálculo del párrafo anterior.

Si no hay acciones aplicables la función debe devolver 'NO HAY ACCIONES APLICABLES'

In [ ]:

In [ ]:
```
adelante(['P1','P4','P5','P6'],C1,['P8','-P2'])
# Salida esperada: 'C'
```

In [ ]:
```
adelante(['P1'],C1,['P2'])
# Salida esperada: 'F'
```

In [ ]:
```
adelante(['R7','R4','R11'],C3,['R8','-R7'])
# Salida esperada: 'NO HAY ACCIONES APLICABLES'
```

In [ ]: