Universidad
Rey Juan Carlos

Test modificaciones práctica 2

Inteligencia Artificial

Introducción

En este documento se proponen una serie de modificaciones al contenido de la práctica 2. Las modificaciones propuestas deben ir dentro del archivo logicPlan.py, no es necesario hacer ningún cambio en otros archivos. Si se hacen cambios en otros ficheros, adjuntar el fichero modificado dentro del archivo comprimido de entrega.

Para completar los ejercicios propuestos, se deben añadir en la parte inferior del archivo logicPlan.py, respetando el nombrado, las siguientes funciones donde irán las modificaciones:

```
...
...
...


'''
   Modifications
'''
def sentence4():
    "*** MODIFICATION 1 ***"
    ...

def sentence5():
    "*** MODIFICATION 2 ***"
    ...
```

1. Implementar sentencia lógica proposicional

Define el método `sentence4()`. Usa el constructor `logic.PropSymbolExpr` para crear los símbolos `AgenteFlecha_0`, `AgenteFlecha_1`, `AgenteDispara_0` y `WumpusMuerto_1`. Crea una

instancia de `logic.Expr` que codifique las siguientes oraciones en lógica proposicional en este orden y sin ninguna simplificación:

- El Agente tiene flechas en el instante 1 si y solo si tiene flechas en el instante 0 y no dispara flecha en el instante 0.
- Que el Agente dispare en el instante 0 y no tenga flecha en el instante 1 implica que el Wumpus está muerto en el instante 1.
- Que el Wumpus no esté muerto en el instante 1, el Agente no dispare en el instante 0 y el Agente tenga flecha en el instante 0 implica que el Agente tiene flecha en el instante 1.
- El Agente no dispara la flecha en el instante 0 y el Wumpus no está muerto en el tiempo 1 y el Agente tiene flecha en el instante 1.

Devuelve la conjunción de las sentencias anteriores.

Los operadores de Python disponibles y sus significados:

- `~ A`: $\neg A$
- `A & B`: $A \wedge B$
- `A | B`: $A \vee B$
- `A >> B`: $A \Rightarrow B$
- `A % B`: $A \Leftrightarrow B$

2. Implementar sentencia lógica proposicional

Define el método `sentence5()`. Usa el constructor `logic.PropSymbolExpr` para crear los símbolos `Agente[1,1]_0`, `Agente[2,1]_1`, `Norte_0` y `Este_0`. Crea una instancia de `logic.Expr` que codifique las siguientes tres oraciones en lógica proposicional en este orden y sin ninguna simplificación:

1. El Agente está en la posición (2,1) en el tiempo 1 si y solo si estaba en la posición (1,1) en el tiempo 0 y se movió hacia el Este en el tiempo 0 y no se movió hacia el Norte en el tiempo 0.
2. El Agente está en la posición (2,1) en el tiempo 1.
3. El Agente en el tiempo 0 solo se ha podido mover al Este o al Norte.

Devuelve la conjunción de las sentencias anteriores.

Entregables

- Se crearán 2 carpetas (`sin_modificar`, `modificada`) en las que se incluirán los archivos de código Python `logicPlan.py`. En la primera irá el archivo que se ha ido completando para la práctica a lo largo de las sesiones y en la segunda el archivo con las modificaciones propuestas en este documento.

- Ambas carpetas se comprimirán en un único archivo de nombre: practica2_NombreApellidos.zip/tar.gz que se entregará vía Aula Virtual.
- El código tiene que ir obligatoriamente comentado explicando su funcionalidad, incluyendo las modificaciones. Debe ser legible y estar debidamente tabulado.
- Se utilizarán sistemas anticopia y se podrá requerir explicación individual de la práctica en caso de duda.
- Entregar la práctica **sin modificar** y la práctica **modificada** con las propuestas de este documento.

Para comprimir las carpetas:

```
tar -czf practica2_NombreApellidos.tar.gz modificada/ sin_modificar/
```

```
zip -r practica2_NombreApellidos.zip modificada/ sin_modificar/
```