Inteligencia Artificial 2018-1
Proyecto1: Representación del Conocimiento

IIMAS-PCIC

Luis Alejandro Lara Patiño
Roberto Monroy Argumedo
Alejandro Ehécatl Morales Huitrón

12 de octubre de 2017

1. Funcionamiento del proyecto

Este proyecto contiene un intérprete de comandos escrito en Prolog, el cual es la base del funcionamiento del mismo.
Para iniciar la aplicación, consultar el archivo `iniciar.pl` y realizar una consulta al predicado `iniciar`, ejecutando los siguientes comandos dentro del *listener* de SWI-Prolog:

```
?. [main].
?. iniciar.
|:
```

O bien, el intérprete puede ser ejecutado directamente desde una terminal, con el siguiente comando:

```
swipl -f main.pl -g "iniciar, halt."
```

Una vez iniciado el intérprete, el *prompt* usual de Prolog cambiará a '`|:`'. Esto indica que ya pueden introducirse comandos.
Inicialmente, se crea una base de conocimiento vacía, la cual puede ser modificada con los comandos presentados en la siguiente sección.

2. Referencia de comandos

2.1. Para agregar información

nuevaClase(Nombre, Padre)
Crea una nueva clase en la base de conocimiento, con el nombre y la clase padre especificados. Este comando verifica que no exista una clase con el mismo nombre previamente.
Argumentos:

- Nombre. Nombre de la nueva clase.
- Padre. Nombre de la clase padre. Se verifica que esta clase exista. En el caso de la clase raíz, este parámetro debe ser `nil`. El sistema verifica que solamente exista una clase raíz.

nuevoObjeto(Nombre, Padre)
Crea un nuevo objeto en la base de conocimiento. El comando verifica que la clase padre exista.
Argumentos:

- Nombre. Nombre del nuevo objeto. Puede ser un nombre único, una lista de nombres o bien, puede crearse un objeto anónimo usando la palabra `nil`. En el caso de un objeto anónimo, el sistema generará un identificador único para el mismo, el cual será desplegado en pantalla y podrá utilizarse para referenciar al objeto con otros comandos.
- Padre. Clase a la cual pertenece el nuevo objeto. Esta clase debe existir en la base de conocimiento.

nuevaPropClase(Nombre, Propiedad, [Valor], [no])
Agrega una nueva propiedad a la clase especificada. El sistema verifica que la clase no contenga previamente dicha propiedad, aun en su forma negada.
Argumentos:
- Nombre. Nombre de la clase a modificar. Esta clase debe existir en la base.
- Propiedad. Nombre de la nueva propiedad. La clase en cuestión no debe tener previamente esta propiedad, en ninguna de sus formas.
- Valor (Opcional). El valor que tendrá la nueva propiedad. Si no se especifica, la nueva propiedad se agregará sin valor.
- Negación (Opcional). Si se desea que la nueva propiedad se introduzca en su forma negada, proporcionar la palabra `no` como último parámetro del comando.

nuevaPropObjeto(Nombre, Propiedad, [Valor], [no])
Agrega una nueva propiedad a todos los objetos con el nombre especificado. El sistema verifica que los objetos no contengan previamente dicha propiedad, aun en su forma negada.
Argumentos:
- Nombre. Nombre o lista de nombres del objeto a modificar. Debe haber por lo menos un objeto con dicho nombre.
- Propiedad. Nombre de la nueva propiedad.
- Valor (Opcional). El valor que tendrá la nueva propiedad. Si no se especifica, la nueva propiedad se agregará sin valor.
- Negación (Opcional). Si se desea que la nueva propiedad se introduzca en su forma negada, proporcionar la palabra `no` como último parámetro del comando.

nuevaRelClase(Nombre, Relacion, Objetivo, [no])
Agrega una nueva relación a la clase especificada. El sistema verifica que la clase no contenga previamente dicha relación con el mismo objetivo, aun en su forma negada.
Argumentos:
- Nombre. Nombre de la clase a modificar. Esta clase debe existir en la base de conocimiento.
- Relación. Nombre de la nueva relación.
- Objetivo. Clase u objeto con el cual se entabla la nueva relación.
- Negación (Opcional). Si se desea que la nueva relación se agregue en su forma negada, proporcional el valor `no` en este argumento.

nuevaRelObjeto(Nombre, Relacion, Objetivo, [no])
Agrega una nueva relación a todos los objetos con el nombre especificado. El sistema verifica que los objetos no contengan previamente dicha relación con el mismo objetivo, aun en su forma negada.
Argumentos:
- Nombre. Nombre o lista de nombres del objeto a modificar. Debe haber por lo menos un objeto con dicho nombre.
- Relación. Nombre de la nueva relación.
- Objetivo. Clase u objeto con el cual se entabla la nueva relación.
- Negación (Opcional). Si se desea que la nueva relación se agregue en su forma negada, proporcional el valor `no` en este argumento.

2.2. Para eliminar información

borrarClase(Nombre)
Elimina la clase con el nombre especificado de la base de conocimiento. Dicha clase debe estar en la base de conocimiento.
Argumentos:

- Nombre. Nombre de la clase a eliminar.

borrarObjeto(Nombre)
Elimina todos los objetos con el nombre especificado de la base de conocimiento.
Argumentos:

- Nombre. Nombre del objeto a eliminar. Puede ser un nombre único o una lista de nombres.

borrarPropClase(Nombre, Propiedad)
Elimina una propiedad de una clase. Dicha clase debe estar en la base de conocimiento, y tener la propiedad mencionada. Se eliminará cualquier forma de la propiedad: negada, no negada, con valor o sin valor.
Argumentos:

- Nombre. Nombre de la clase a modificar.
- Propiedad. Nombre de la propiedad a eliminar.

borrarPropObjeto(Nombre, Propiedad)
Elimina una propiedad de todos los objetos con el nombre especificado. Debe haber por lo menos un objeto con el nombre especificado, y éstos deben tener la propiedad mencionada. Se eliminará cualquier forma de la propiedad: negada, no negada, con valor o sin valor.
Argumentos:

- Nombre. Nombre del objeto a modificar. Puede ser un nombre único, o una lista de nombres.
- Propiedad. Nombre de la propiedad a eliminar.

borrarRelClase(Nombre, Relacion, Objetivo)
Elimina una relación de la clase especificada. La relación se eliminará solamente si tiene el objetivo dado. La clase en cuestión debe existir, y tener la relación mencionada con el objetivo dado.
Argumentos:

- Nombre. Nombre de la clase a modificar.
- Relación. Nombre de la relación a eliminar.
- Objetivo. Clase u objeto con el cual se tiene la relación a eliminar.

borrarRelObjeto(Nombre, Relacion, Objetivo)
Elimina una relación de todos los objetos con el nombre especificado. La relación se eliminará solamente si tiene el objetivo dado. Debe haber por lo menos un objeto con el nombre dado, y éstos deben tener la relación mencionada con el objetivo dado.
Argumentos:

- Nombre. Nombre del objeto a modificar. Puede ser un nombre único, o una lista de nombres.
- Relación. Nombre de la relación a eliminar.
- Objetivo. Clase u objeto con el cual se tiene la relación a eliminar.

2.3. Utilitarios

cargar(Nombre)
Lee una base desde el archivo dado y la carga como la base de conocimiento actual.
Argumentos:

- Nombre. Nombre del archivo a leer. Este archivo debe existir dentro del directorio `bases`.

guardar(Nombre)
Guarda todos los contenidos de la base actual en un archivo, dentro del directorio *bases*. Si el archivo no existe, se creará; si ya existe, todos sus contenidos previos se perderán.
Argumentos:

- Nombre. Nombre del archivo a escribir.

ver
Imprime todos los contenidos de la base actual en pantalla.