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

Índice

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 consultar información

extClase(Clase)
Muestra en una lista el conjunto de todos los objetos que pertenecen a una clase (cada objeto con su lista de nombres) ya sea que se hayan declarado directamente o están en la cerradura de la relación de herencia.
Argumentos:

- Clase. Nombre de la clase a extender.

extProp(Propiedad)
Muestra en una lista el conjunto de todos los objetos que tienen una propiedad específica ya sea por declaración directa o por herencia. Además se indica su valor.
Argumentos:

- Propiedad. Nombre de la propiedad a extender.

Ejemplo:
En una base de conocimiento de animales parecida a los ejemplos de la clase, si tenemos una propiedad *color=>negro* que tienen los ornitorrincos pero no los lemures, y además *orni* es un ornitorrinco y *juan* es un lémur:

```
|: extProp(color).
[orni:negro,no(juan:negro)]
```

extRel(Relación)
Muestra en una lista el conjunto de todos los objetos que tienen una relación específica ya sea por declaración directa o por herencia. Además indica con quién están relacionados.
Argumentos:

- relación. Nombre de la relación a extender.

Ejemplo:
Si tenemos la relación *amigo=>juan* en el objeto *orni* del ejemplo pasado:

```
|: extRel(amigo).
[orni:juan]
```

Cabe mencionar que tanto **extProp** como **extRel** dan respuestas adecuadas en caso de que haya defaults y excepciones. Por ejemplo si en la clase aves hay una propiedad *tamaño=>nil* y un objeto "chucho" de su subclase águilas tiene la propiedad *tamaño=>largo* entonces la salida de la extensión de la propiedad tamaño será:

```
[chucho:largo,otro1:nil,otro2:nil]
```

donde *otro1* y *otro2* son objetos de alguna subclase de aves diferente de águilas.

clasesObj(Objeto)
Muestra todas las clases a las que pertenece un objeto incluidas los ancestros, es decir, las clases en el camino hacia la raíz desde la clase a la que pertenece el objeto.
Argumentos:

- Objeto. Nombre del objeto a consultar.

propsObjeto(Objeto)
Muestra todas las propiedades de un objeto declaradas directamente o por herencia.
Argumentos:

- Objeto. Nombre del objeto a consultar.

propsClase(Clase)
Muestra todas las propiedades de una clase declaradas directamente o por herencia.
Argumentos:
- Clase. Nombre de la clase consultar.

relsObjeto(Objeto)
Muestra todas las relaciones de un objeto declaradas directamente o por herencia.
Argumentos:
- Objeto. Nombre del objeto a consultar.

relsClase(Clase)
Muestra todas las relaciones de una clase declaradas directamente o por herencia.
Argumentos:
- Clase. Nombre de la clase consultar.

2.2. 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.

2.3. 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.

2.4. Modificar información

modificarNombreClase(AntiguoNombre, NuevoNombre)
Modifica el nombre de una Clase. Si el nuevo nombre existe en la base de datos el nombre no se modifica para evitar clases con el mismo nombre. El cambio de nombre se actualiza en las relaciones que puedan existir con otras clases y objetos.
Argumentos:

- AntiguoNombre. El nombre de la clase a se va a cambiar.
- NuevoNomnbre. El nuevo nombre de la clase.

modificarNombreObjeto(AntiguoNombre, NuevoNombre)
Modifica el nombre de un Objeto. El cambio de nombre se actualiza en las relaciones que puedan existir con otras clases y objetos.
Argumentos:

- AntiguoNombre. El nombre del objeto que se va a cambiar.
- NuevoNombre. El nuevo nombre del objeto.

modificarPropiedad(Nombre, Propiedad, NuevoValor)
Modifica el valor de una propiedad de una clase u objeto, no es necesario especificar el valor actual de la propiedad en caso de tenerlo.
Argumentos:

- Nombre. El nombre de la clase u objeto cuya propiedad será modificada.
- Propiedad. El nombre de la propiedad que será modificada.
- NuevoValor. El nuevo valor de la propiedad.

modificarPropiedad(Nombre, Propiedad, Valor, NuevoValor)
Modifica el valor de una propiedad de una clase u objeto, se puede especificar el valor actual de la porpiedad en caso de que existan más propiedades con el mismo nombre.
Argumentos:

- Nombre. El nombre de la clase u objeto cuya propiedad será modificada.
- Propiedad. El nombre de la propiedad que será modificada.
- Valor. El valor actual en la base de datos de la propiedad.
- NuevoValor. El nuevo valor de la propiedad.

modificarRelacion(Nombre, Relacion, NuevoValor)
Modifica el valor de una relacion de una clase u objeto, no es necesario especificar el valor actual de la relacion.
Argumentos:

- Nombre. El nombre de la clase u objeto cuya relación será modificada.
- Propiedad. El nombre de la relación que será modificada.
- NuevoValor. El nuevo valor de la relación.

modificarRelacion(Nombre, Relacion, Valor, NuevoValor)
Modifica el valor de la relación de una clase u objeto, se puede especificar el valor actual de la relacion en caso de que existan más relaciones con el mismo nombre.
Argumentos:

- Nombre. El nombre de la clase u objeto cuya relación será modificada.
- Propiedad. El nombre de la relación que será modificada.
- Valor. El valor actual de la relación.
- NuevoValor. El nuevo valor de la relación.

negarPropiedad(Nombre, Propiedad)
Niega la propiedad de la calse u objeto especificado.
Argumentos:

- Nombre. El nombre de la clase u objeto que será modificado.
- Propiedad. La propiedad que será negada.

negarPropiedad(Nombre, Propiedad, Valor)
Niega la propiedad de la clase u objeto especifucado, se puede especificar el valor de la propiedad para evitar modificar distintas propiedades con el mismo nombre.
Argumentos:

- Nombre. El nombre de la clase u objeto que será modificado.
- Propiedad. La propiedad que será negada.
- Valor. El valor actual de la propiedad.

negarRelacion(Nombre, Relacion)
Niega la relación de la clase u objeto especificado.
Argumentos:

- Nombre. El nombre de la clase u objeto que será modificado.
- Relación. La relación que será negada.

negarRelacion(Nombre, Relacion, Valor)
Niega la relación de la clase u objeto especificado, se puede especificar el valor de la relación para evitar modificar distintas relaciones con el mismo nombre.
Argumentos:

- Nombre. El nombre de la clase u objeto que será modificado.

- Relación. La relación que será negada.
- Valor. El valor actual de la relación.

2.5. 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.