Seminario 2: Implementación de SBR en C y Python

1) ¿Cómo se instala CLIPSPy?

Primeramente, necesitamos tener instalado Python y CLIPS (ya sea en Windows o en Linux). Tras asegurarnos de ello, la instalación es directa usando:

- **Linux → *[sudo] pip install clipspy***
- **Windows → *pip install clipspy***

Una vez instalado, para usarlo en Python lo importamos con el comando:

import clips

2) ¿Se pueden evaluar en Python (mediante CLIPSPy) expresiones de CLIPS? ¿Cómo?

Sí se puede. Para ello, seguimos los siguientes pasos:

1. Importar CLIPSPy en Python:

   import clips.

2. Crear una instancia del entorno CLIPS:

   env = clips.Environment()

3. Cargar y ejecutar las reglas de CLIPS:

   env.load("reglas.clp")

   env.reset()

   env.run()

4. Evaluar la expresión de CLIPS en Python utilizando el método *eval()*:

   resultado = env.eval("(+ 2 3")

   print(resultado)

3) Mediante CLIPSPy se pueden utilizar dentro de CLIPS funciones de Python. ¿Cómo se hace?

Para ello, seguimos los siguientes pasos:

1. Importar CLIPSPy en Python:

   import clips.

2. Crear una instancia del entorno CLIPS:

   env = clips.Environment()

3. Definir la función en Python:

   def suma(a, b):

   return a+b

4. Registrar la función de Python en el entorno CLIPS:

   env.add_python_function(suma, "suma")

5. Cargar y ejecutar las reglas de CLIPS:

   env.load("reglas.clp")

   env.reset()

   env.run()

   En dichas reglas de CLIPS, podremos llamar a la función de Python utilizando la sintaxis (Python-function <nombre> <argumentos>. Por ejemplo:

```
(defrule ejemplo
    ?resultado <- (Python-function suma 2 3)
=>
    (printout t "El resultado es " ?result crlf)
)
```

4) **¿Qué métodos de CLIPSPy asertan y retractan un hecho en CLIPS?**

- **Asertar** un hecho:

  hecho = env.build_fact("(hecho)")
  fact.set_slot("atributo1", "valor1")
  env.assert_fact(hecho)

- **Retractar** un hecho:

  hecho = env.find_fact("(hecho)")
  env.retract(hecho)

5) **¿Cómo se ejecutaría en Python (mediante CLIPSPy) un sistema basado en reglas definido mediante un fichero .clp?**

1. Importar CLIPSPy en Python:

   import clips.

2. Crear una instancia del entorno CLIPS:

   env = clips.Environment()

3. Cargar y ejecutar las el sistema basado en reglas de CLIPS:

   env.load("sistema.clp")

   env.reset()

   env.run()

6) **Describe brevemente cómo convertirías un sistema basado en reglas definido mediante un fichero .clp en un fichero ejecutable.**

En Python, ya se ha respondido a esta pregunta en la pregunta 5 (ejecutamos directamente dicho código).

En C, hay que compilarlo y posteriormente ejecutarlo. Hay que tener en cuenta que el archivo .clp tiene que estar donde el ejecutable. Otra cosa a destacar es que los fichero .clp no se pueden convertir a ejecutables, sino que se acceden a ellos a través de los ejecutables. Un ejemplo del código en C, similar al que se dio para Python, sería el siguiente:

```
#include “clips.h”
Int main() {
        void *env;
        env = CreateEnvironment();
        EnvLoad(env, “sistema.clp”);
        EnvReset(env);
        EnvRun(env, -1);
        DestroyEnvironment(env);
}
```

7) Describe brevemente cómo incluirías en CLIPS una función definida en C.

1. Crear un archivo en C con la implementación de la función que queremos incluir. Llamemos a dicho archivo, por ejemplo, *función.c*.

2. Compilar el archivo de código fuente en C:
   gcc -c funcion.c -o función.o

3. Vincular el archivo objeto con la biblioteca CLIPS, creando así una biblioteca compartida:
   gcc -shared funcion.o -o función.so -L/path_to_clips_library -lclips

4. Cargar dicha biblioteca compartida en CLIPS:
   (ffi:load-foreign-librart “/path_to_funcion.so”)

5. Ya podemos llamar a la función definida en C desde CLIPS. Por ejemplo, si en *funcion.c* teníamos definida la función *sumar* para sumar dos números enteros:
   (ffi:foreign-funcall “suma” :int 2 :int 3)

8) **Describe brevemente cómo incluirías un sistema basado en reglas definido mediante un fichero .clp dentro de tu programa escrito en C.**

Esto ya se respondió en la pregunta 6. El código en C sería básicamente el siguiente:

```
#include “clips.h”
Int main() {
        void *env;
        env = CreateEnvironment();
        EnvLoad(env, “sistema.clp”);
        EnvReset(env);
        EnvRun(env, -1);
        DestroyEnvironment(env);
}
```

9) **¿Qué funciones se utilizan para asertar o retractar un hecho en un sistema basado en reglas embebido en un programa de C?**

- **Asertar** un hecho:

  EnvAssertString(environment, “hecho”)

- **Retractar** un hecho:

  EnvRetract(environment, hechoPtr)

  (*hechoPtr* es un puntero al hecho que queremos retractar)

10) **¿Se pueden ejecutar varios sistemas basados en reglas distintos dentro de un mismo programa de C?**

Sí se puede. Para ello, inicializamos varios entornos CLIPS en C, uno para cada sistema basado en reglas. Por ejemplo, para ejecutar dos sistemas basados en reglas distintos en un mismo programa de C:

```
#include “clips.h”
Int main() {
        void *env1;
        void *env2;
        env1 = CreateEnvironment();
        env2 = CreateEnvironment();
        EnvLoad(env1, “sistema1.clp”);
        EnvLoad(env2, “sistema2.clp”);
        EnvReset(env1);
        EnvReset(env2);
        EnvRun(env1, -1);
        EnvRun(env2, -1);
        DestroyEnvironment(env1);
        DestroyEnvironment(env2);
}
```