Respuestas Seminario 2

1. ¿Cómo se instala CLIPSPy?

El proceso de instalación depende del S.O.:

- Linux:

En Linux CLIPSPy es un paquete para la arquitectura x86_64. Para instalar el paquete sólo hace falta ejecutar el siguiente comando en el terminal, con los privilegios de root:

```
[sudo] pip install clipspy
```

- Windows:

En Windows se debe ejecutar el mismo comando pero sin incluir sudo, que es propio de Linux:

```
pip install clipspy
```

- A partir de los ficheros fuente:

En el caso de que se quiere instalar el paquete a partir de sus ficheros fuente, se deben descargar junto con el archivo Makefile; y en la ruta donde se hayan descargado los ficheros fuente y el archivo Makefile, ejecutar el siguiente comando:

```
make
sudo make install
```

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

Si, una vez instalado el paquete CLIPSPy, se importa el paquete en el fichero .py donde se vaya a crear el programa. Una vez importado lo primero que se debe hacer es crear un entorno de CLIPS con la siguiente función:

```
environment = clips.Environment()
```

Y por último se llama a la función eval del paquete CIPSPy para evaluar un valor multicampo, escrito con un string. Ejemplo:

```
environment.eval(“(create$ hammer drill saw screw)”)
```

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

Al igual que en la anterior pregunta hace falta importar el paquete y crear un entorno de CLIPS. Una vez hecha esta parte se ejecuta el siguiente método de CLIPSPY:

```
environment.define_function(funcion_python)
```

Cuyo único parámetro es el nombre de la función de python que ha sido creada antes de llamar al método.

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

Para asertar un hecho se usa el siguiente método de CLIPSPy:

- assert <hecho>

El hecho a asertar puede ser ordenado o no ordenado.

Para retractar un hecho se usa el siguiente método de CLIPSPy:

- retract() → Con este método se retracta el hecho del entorno de CLIPS

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

Primero se debe importar el paquete y crear el entorno de CLIPS. Luego se debe cargar el archivo .clp en el entorno con el siguiente método:

```
environment.load('nombre_fichero.clp')
```

Después se deben asertar al entorno los hechos iniciales. Estos hechos iniciales se pueden añadir en forma de hechos ordenados o de hechos no ordenados mediante templates.

Para crear un hecho ordenado se utiliza el siguiente método:

- assert_string('formato de hecho en CLIPS') → Este método devuelve un hecho ordenado

Para crear un hecho no ordenado se utilizan los siguiente métodos:

- build(string: str) → Este método crea la plantilla del template
- find_template(string:str) → Este método devuelve la plantilla del template cuyo nombre es igual al del string que se para como parámetro
- assert_fact(**slots) → Este método devuelve un hecho no ordenado con los valores de los campos que se indican en los parámetros del método

Para asertar ambos tipos de hechos se utiliza el método visto en la pregunta 4.

Una vez asertados todos los hechos iniciales, se procede a ejecutar el sistema con el siguiente método:

- run()

6. Describe brevemente como convertirías un sistema basado en reglas definido mediante un fichero .clp en un fichero ejecutable

Los pasos a realizar son los siguientes:

- Convertir los constructores en archivo de C, para ello es necesario abrir CLIPS, cargar todos los constructores y llamar al comando constructs-to-c. Este comando lo que hace es convertir los distintos constructores en ficheros de salida en C. En este comando se debe indicar un id para identificar al sistema que se está convirtiendo en un fichero en C.
- Fijar en el fichero de cabecera el flag RUN_TIME a 1, y posteriormente compilar todos los ficheros en C que se generaron en el anterior paso.
- Debemos modificar el fichero main.c para que inicie el entorno, lo reinicie para cargar los hechos y la agenda, y para que ejecute el sistema. Y por último para que destruya el entorno que se creo al principio del programa. Para ello se utilizan las siguientes funciones:
  - InitCImage_<id>
  - EnvReset(<entorno>)
  - EnvRun(<entorno>)
  - DestroyEnvironment(<entorno>)
- Recompilamos todos los ficheros en C.
- Enlazamos todos los ficheros que hemos ido creando con el fichero principal. Y con esto obtenemos el ejecutable del sistema definido mediante un fichero .clp

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

Para incluir una función definida en C en un programa en CLIPS debemos modificar el archivo userfunctions.c, que reside en CLIPS.

Dentro de la función EnvUserFunctions debemos hacer los siguiente:

- Primero debemos declarar la función en C a añadir
- Segundo debemos definir nuestra función para registrarla en el sistema, esto se hace con la función EnvDefineFunction, cuyos parámetros son los siguiente:
  - 1º: Entorno en el que se va a definir la función
  - 2º: Nombre de la función, en forma de string
  - 3º: Tipo del valor el cual va a ser devuelto a CLIPS
  - 4º: Puntero a la función actual en C, el nombre de la función compilada
  - 5º: String que representa al cuarto parámetro, debería ser idéntico al tercer argumento

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

CLIPS fue diseñado para ser incluido en otros programa, por ello es muy fácil de hacer. El proceso es el siguiente:

- Incluimos en el fichero en C el fichero de cabera “clips.h”, de la siguiente forma: #include “clips.h”
- Para poder usar la mayoría de funciones de CLIPS en el fichero en C debemos crear un entorno, para ello usamos la función CreateEnvironment, con la que crearemos un puntero a un entorno que nos servirá para llamar a las funciones de CLIPS.
- Para crear el programa en C que incluye el sistema de CLIPS debemos compilar y enlazar todos los ficheros en c con todos los ficheros escritos con CLIPS, excepto main.c. Si además de usar la librería de CLIPS se utiliza otra se deberá usar una opción de enlazado distintas o usar wrapper.

Al incluir el archivo “clips.h” disponemos de muchas funciones que realizan las mismas capacidades que CLIPS. Disponemos de funciones de entorno, de funciones de depuración, de funciones para definir templates, etc; es decir, podemos hacer la mayoría de cosas que se pueden hacer en CLIPS pero a través de las funciones que se nos proporcionan y del puntero al entorno creado.

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

Para asertar hechos se utilizan las siguientes funciones:

- EnvAssert(environment, factPtr) → Función para asertar un hecho en el entorno indicado por el primer parámetro de la función, y el hecho está indicado por el puntero al hecho indicado en el segundo parámetro.
- EnvAssertString(environment, string) → Función para asertar un hecho en el entorno indicado por el primer parámetro de la función, y el hecho está indicado por el string indicado en el segundo parámetro. El string puede representar tanto un hecho ordenado como un hecho no ordenado.

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

Si se pueden ejecutar varios sistemas basados en reglas en el mismo programa en C, para realizarlo lo único que hay que hacer es realizar los mismos pasos que se realizan en la pregunta 8 con la única diferencia de que se realizan varias ejecuciones del comando CreateEnvironment, tantas como sistemas basado en reglas se quieran añadir, para que cada sistema tenga su propio entorno. Y no habrá problemas ya que cada entorno se ejecutará de forma independiente.

Cuando se haya terminado con un entorno se puede borrar con el comando DestroyEnvironment.