Preguntas CLIPSPy

IC

Miguel Lentisco Ballesteros

1. ¿Cómo se instala CLIPSPy?

   Ejecutando en terminal `pip install clipspy`.

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

   Sí, con `eval(expresion)`, por ejemplo evaluemos una suma:

   ```
   from clips import Environment
   env = Environment()
   expr = "(+ 3 2)"
   env.eval(expr)
   ```

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

   Usando `define_function(funcion)` incluir código de Python en el entorno de CLIPS; por ejemplo, una función para multiplicar por 3:

   ```
   from clips import Environment

   def mult_tres(num):
       return num * 3

   env = Environment()
   env.define_function(mult_tres)

   env.eval("(mult_tres 5)")
   ```

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

   Con los métodos `assertit(hecho)` (o `assert_string(string_hecho)` y `retract(hecho)`, por ejemplo:

   ```
   import clips import Environment
   env = Environment()
   # Hecho como string
   env.assert_string("(hecho)")
   ```

```
temp = env.find_template("hecho")
# Nuevo hecho
fact = temp.new_fact()
fact.append(32)
fact.extend(("foo", "bar"))
# Asertamos el hecho
fact.assertit()
# Vemos que está en los hechos
for f in env.facts().
  print(f)
# Retractamos el hecho
f.retract()
# Ya no está
for f in env.facts().
  print(f)
```

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

   Cargamos el fichero con `load(ruta_fichero)`, reseteamos con `reset()` y ejecutamos con `run()`, por ejemplo:

   ```
   from clips import Environment
   env = Environment()
   # Cargamos
   env.load("reglas.clp")
   # Reseteamos
   env.reset()
   # Ejecutamos
   env.run()
   ```

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

   Primero iniciamos CLIPS y ejecutamos `(constructs-to-c reglas 1)` para convertir todas las estructuras de "reglas.clp" en distintos archivos ".c" con las estructuras de nuestro sistema basado en reglas.

   Después ponemos `RUN_TIME = 1` en el archivo "setup.h" y compilamos todos los archivos ".c" que hemos generado. Finalmente podemos usar las estructuras en nuestro "main.c" usando un entorno que tiene todo guardado (`InitCImage_1`, siendo 1 la id).

   Un ejemplo de "main.c" sería:

   ```
   #include "clips.h"
   int main() {
     void* env;
     extern void* InitCImage_1();
     env = InitCImage_1();
   ```

```
  EnvReset(env);
  EnvRun(env, -1);
  DestroyEnvironment(env);
}
```

Recompilamos todo el código fuente de CLIPS, enlazamos todo, obteniendo finalmente un ejecutable que no depende del archivo ".clp", ya integra todo en un solo ejecutable.

7. Describe brevemente como incluirías en CLISP una función definida en C

   Añadimos en nuestro archivo "main.c" nuestra función que queremos incluir, y modificando la función `EnvUserFunctions` en el archivo "userfunctions.c" para incluir nuestra función (declaramos y llamamos a `DefineFunction`).

   Por ejemplo nuestro archivo "main.c", creamos una función `TripleDouble`:

```
#include "clips.h"

int main() {
  void *env;

  env = CreateEnvironment();
  EnvLoad(env, "reglas.clp");
  EnvReset(env):
  EnvRun(env, -1);
  DestroyEnvironment(env);
}

void TripleDouble(void* env, DATA_OBJECT_PTR returnValuePtr) {
  double doubleValue;
  void *value;

  value = GetpValue(returnValuePtr);
  doubleValue = 3.0 * ValueToDouble(value);
  SetpValue(returnValuePtr, EnvAddDouble(env, doubleValue));

  return;
}
```

   Y modificamos en "userfunctions.c":

```
void EnvUserFunctions(void *environment) {
  extern double TripleDouble(void *, DATA_OBJECT_PTR);
  EnvDefineFunction(environment, "triple", "d", PTIEF TripleDouble, "TripleDouble");
}
```

   De esta manera compilamos todo, enlazamos y ya tenemos nuestro ejecutable.

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

   Cogemos todos los archivos fuentes de CLIPS, creamos un nuevo archivo donde en el main hacemos `EnvLoad` para cargar nuestro fichero .clp, y añadimos las funciones `EnvReset` y `EnvRun`; compilamos todo excepto "main.c" y enlazamos todo obteniendo nuestro ejecutable.

   Por ejemplo creamos el archivo "main.c":

```
# include "clips.h"

int main() {
  void *env;

  env = CreateEnvironment();

  EnvLoad(env, "reglas.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?

   Para asertar hechos tenemos `EnvAssert(env, hecho)` o `EnvAssertString(env, string)` y para retractar `EnvRetract(env, hecho)`.

   Por ejemplo si tenemos en "reglas.clp":

```
(deftemplate ejemplo
  (slot x)
)
```

   Creamos nuestro archivo "reglas.c":

```
#include "clips.h"

int main() {
  void *env;

  env = CreateEnvironment();
  EnvLoad(env, "reglas.clp");
  templatePtr = EnvFindDeftemplate(env, "ejemplo");
  nuevoHecho = EnvCreateFact(env, templatePtr);

  DATA_OBJECT valor;
  valor.type = INTEGER;
```

```
  valor.value = EnvAddLong(env, 3);
  EnvPutFactSlot(env, nuevoHecho, "x", &valor);
  EnvAssert(env, nuevoHecho);

  EnvReset(env);
  EnvRun(env, 1);

  EnvRetract(env, nuevoHecho);
  EnvRUn(env, -1);

  DestroyEnvironment(env);
}
```

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

    Sí, utilizando varios `Environment` ya que cada uno mantiene su propio conjunto de estructuras de datos y pueden ser ejecutandos independientemente de otros entornos.

    Por ejemplo podemos cargar "reglas1.clp" y "reglas2.clp" en "reglas.C":

```
#include "clips.h"
int main() {
  void *env1, *env2;

  env1 = CreateEnvironment();
  env2 = CreateEnvironment();

  EnvLoad(env1, "reglas1.clp");
  EnvLoad(env2, "reglas2.clp");

  EnvReset(env1);
  EnvReset(env2);

  EnvRun(env1, -1);
  EnvRun(env2, -1);

  DestroyEnvironment(env1);
  DestroyEnvironment(env2);
}
```