Guión de prácticas 2.1 CLIPS

En esta práctica conoceremos las funcionalidades básicas de CLIPS y plantearemos su utilización para la construcción de un Sistema Basado en Conocimiento.

Para instalar CLIPS en cualquier equipo basta descargar el fichero de instalación correspondiente desde la página oficial de CLIPS: http://clipsrules.sourceforge.net/

Los ficheros CLIPS utilizan la extensión **`.clp`** y se pueden crear con cualquier editor de texto.

1. Fundamentos de CLIPS

En esta parte del guion trabajaremos algunos fundamentos básicos de CLIPS, con el objetivo de familiarizarnos con los elementos más importantes de dicho entorno. Al final de cada ejercicio se formulan un conjunto de cuestiones que deberéis responder razonadamente en la entrega de la práctica.

Ejercicio 1.1. Hechos

Introduce los siguientes hechos (todos ellos en formato dupla, o vector de dos elementos) en la ventana **Dialog** y visualiza el resultado en la ventana **Facts** (recuerda utilizar la sintaxis correcta de CLIPS en todos los casos):

```
(nombre Juanjo) (nombre Juan) (apellido-1 Cruz) (apellido-1 Perez) (apellido-2 Lopez) (nombre Federico) (apellido-1 Perez) (apellido-2 Jimenez)
```

Cuestiones 1.1 (ver anexo, al final del documento)

Ejercicio 1.2. Reglas

Programa las dos reglas que se indican a continuación y visualiza su activación en la ventana **Agenda**:

- REGLA 1: debe tener como antecedente el hecho **`(nombre TU_NOMBRE)`** y como consecuente una instrucción que imprima en pantalla "Tu nombre de pila es TU_NOMBRE"

Guión de prácticas 2.1 CLIPS

- REGLA 2: debe tener como antecedente los hechos **(nombre TU_NOMBRE)**, **(apellido-1 TU_PRIMER_APELLIDO)** y **(apellido-2 TU_SEGUNDO_APELLIDO),**

  y el consecuente imprimirá en pantalla "Te llamas **TU_NOMBRE TU_PRIMER_APELLIDO TU_SEGUNDO_APELLIDO** ".

Cuestiones 1.2 (ver anexo)

Ejercicio 1.3. Ejecución de Reglas

Ejecuta paso a paso las reglas del ejercicio anterior, utilizando **(run 1)**

Cuestiones 1.3 (ver anexo)

Ejercicio 1.4. Definición de hechos con deffacts

Define los hechos de los ejercicios anteriores con **deffacts** y guárdalos junto con las reglas en un fichero con extensión **.clp**. Usa para ello cualquier editor de texto (e.g., Notepad++). Carga el fichero con la opción **load** del menú principal.

Verifica que el programa produce el mismo resultado que en el ejercicio anterior y finalmente limpia la base de conocimiento.

Cuestiones 1.4 (ver anexo)

Ejercicio 1.5. Uso de variables en las reglas. Diseño paso a paso de un primer SBC elemental

Guión de prácticas 2.1 CLIPS

Se pide implementar un programa que controle el comportamiento básico de un *agente inteligente* (que puede ser vehículo o peatón) para pasar un cruce con semáforo. El programa tendrá un mínimo nivel de interactividad, porque preguntará:

- si el agente inteligente es **vehículo** o **peatón**
- el estado del semáforo (**rojo**, **verde** o **ámbar/intermitente**), para así decidir la acción del agente inteligente (seguir, parar o pasar con cuidado).

La primera regla se utiliza, como se ve a continuación, para que la persona usuaria defina por teclado el tipo de agente inteligente. La regla hace uso de **printout** para salida por pantalla y (**read**), para solicitar el dato por teclado.

```
;     regla para definir interactivamente el tipo de agente
;     en la v6.4 AL NO ESPECIFICAR ANTECEDENTE SE INSTANCIA CON f-0
;     en la v6.3 y anteriores se debe especificar el ANTECEDENTE
;
(defrule regla-tipo-agente
      =>
      (printout t "Que tipo de agente es (vehículo/peaton)? ") (assert (tipo-agente
(read)))
)
```

La segunda regla preguntará, de forma similar a la anterior, cómo está el semáforo para los coches, y se activará sólo en el caso de que el agente sea un vehículo. Almacenará el color del semáforo en un hecho (**semaforo <color>**), donde <**color**> puede ser verde, rojo, ámbar o intermitente.

```
;
;     regla regla-semaforo-vehículo
;
(defrule regla-semaforo-vehículo ...
```

La tercera regla preguntará cómo está el semáforo para los peatones, y se activará sólo en el caso de que el agente sea un peatón. Almacenará el color del semáforo en un hecho (**semaforo <color>**), donde <**color**> puede ser verde, rojo, ámbar o intermitente.

Guión de prácticas 2.1 CLIPS

```
;
;     regla regla-semaforo-peaton
;

(defrule regla-semaforo-peaton ...
```

La cuarta regla permitirá pasar al agente si el semáforo está en verde. Usará una variable que permite equiparar el hecho (**tipo-agente ?x**) con cualquier tipo de agente (peatón o vehículo). Dicha variable se queda ligada al valor con el que se equipara durante la ejecución de la regla, pudiéndose usar su valor en el consecuente de la regla (p.ej., en el **printout**).

```
;
; regla agente-puede-pasar
;
(defrule agente-puede-pasar (tipo-agente ?tipo) (semaforo verde)
=>
(printout t "El agente " ?tipo " puede pasar" crlf)
)
```

La quinta regla se activa si el estado del semáforo está rojo y, al igual que la anterior, muestra el mensaje correspondiente.

```
;
; regla debe-esperar
;
```

La sexta regla se activa cuando el semáforo está en ámbar o intermitente, y muestra el mensaje oportuno. Utilice el símbolo de la disyuntiva para considerar las dos opciones en el mismo antecedente de la regla: (**semaforo ambar | intermitente**)

```
;
```

Guión de prácticas 2.1 CLIPS

```
; regla con-cuidado
;
```

Cuestiones 1.5 (ver anexo)

Provoca diferentes errores de compilación para familiarizarse con los mensajes de error. Al menos probad a quitar algún paréntesis, escribir comandos erróneos como ***assertgdf***, ***defruledg***, suprimir el nombre de la regla, cambiar alguna mayúscula por minúscula, suprimir el punto y coma que precede a un comentario, usar una variable en la parte derecha de una regla sin definirla en la parte izquierda, cambiar el símbolo de implicación **=>** por otro como **->**…).

2. Ejercicios básicos con CLIPS

En este apartado realizaremos los ejercicios 1-5 correspondientes al tutorial de la UGR. Previamente debéis haber leído con detenimiento el tutorial correspondiente.
En estos ejercicios cargaremos siempre los conjuntos de reglas o hechos desde un fichero.

Para automatizar el proceso de ejecución, crearemos una macro para cada ejercicio. Las macros en CLIPS son ficheros con extensión **.bat** que contienen la secuencia de órdenes necesarias para cargar los ficheros de hechos y reglas. Como convención denotaremos cada fichero con el nombre **X_inicio.bat**, donde X representa el número del ejercicio.

Esta macro deberá contener las siguientes órdenes:

```
(clear)
(load <nombre de fichero>.clp) (load <otro fichero>.clp) …
(reset)
```

y la podemos ejecutar con la opción **Load batch** del menú.

Ejercicio 2.1

Cree un fichero llamado **1_personas.clp** con la siguiente regla:

```
(defrule Regla1
```

Guión de prácticas 2.1 CLIPS

```
(EsPadre Pedro)
=>
(assert (QuiereASusHijos Pedro)))
```

Crear el fichero **1_inicio.bat** correspondiente (para cargar el fichero **1_personas.clp**).

Añade desde la línea de comandos el hecho **(EsPadre Pedro)** y ejecuta el programa con **(run 1).**

Reinicia (**clear**) CLIPS y vuelve a cargar el fichero con la regla anterior. Ahora, repetiremos el ejercicio, pero añadiendo el hecho desde un fichero en lugar de hacerlo desde la línea de comandos. Para ello haremos lo siguiente:

1. Crearemos un fichero nuevo llamado **1_personas.facts.clp** que contenga:

```
(deffacts VariosHechos (EsPadre Pedro) (EsPadre Juan))
```

2. Cargaremos ambos ficheros desde **1_inicio.bat**,
3. Ejecutaremos el programa.

Por último, reinicia nuevamente CLIPS y modifica la regla como sigue:

```
(defrule LosPadresQuierenALosHijos (EsPadre ?variable)
=>
(assert (QuiereASusHijos ?variable)))
```

Si cargamos esta regla y el fichero de datos anterior, ¿cuántas activaciones se producen de la regla? Prueba con:

- **(run 1)** y (**reset**)
- **(run 2)** y **(reset)**
- **(run)**

Guión de prácticas 2.1 CLIPS

Ejercicio 2.2 Alarma

Implementa y prueba el ejemplo de la alarma descrito en la p. 17 del tutorial de CLIPS de la UGR (también en la diapositiva 27 de la presentación, donde se ha corregido una errata), creando para ello los ficheros

`2_alarma.clp` y **`2_alarma.facts.clp`**

Ejercicio 2.3 Relaciones de parentesco (I)

Definir un sistema basado en conocimiento que permita determinar las relaciones de consanguinidad entre varias personas de una misma familia. El sistema utilizará como hechos los siguientes:

- Una persona es Padre/madre de otra persona
- Una persona es hombre/mujer

Considerar el caso siguiente: Isabel y Felipe son pareja y padres de Carlos, Ana, Andrés y Eduardo. Carlos ha tenido dos hijos: Guillermo (padre de Jorge, Carlota y Luis) y Enrique, padre de Archie y Lillibet.

Se definirán **mediante reglas** las siguientes relaciones de consanguinidad:

- hermano/a
- tío/tía
- sobrino/a
- primo/a
- abuelo/a

Cuestiones 2.3 (ver anexo)

Ejercicio 2.4 Relaciones de parentesco (II)

En el ejercicio anterior se quiere incluir a las siguientes personas dentro del árbol familiar:

- La actual pareja de Carlos (Camilla) y su anterior pareja (Diana, madre de sus hijos)
- La actual pareja de Ana (Timothy) y su anterior pareja (Mark, padre de sus hijos)
- La anterior pareja de Andrés (Sarah, madre de sus hijos)
- La actual pareja de Eduardo (Sofía, madre de sus hijos)

Guión de prácticas 2.1 CLIPS

Cuestiones 2.4 (ver anexo)

3. Un sistema de diagnóstico en CLIPS

El ejercicio consiste en utilizar y comprender el sistema de diagnóstico de vehículos **auto.clp**, que es uno de los ejemplos existentes en CLIPS, y que implementa una versión muy simplificada del problema, con un número limitado de respuestas. El ejemplo, sin embargo, permite apreciar cómo se puede controlar el flujo del razonamiento y de la generación de hipótesis de diagnóstico.

En primer lugar, debéis descargar el ejemplo y ejecutarlo para el caso inicial siguiente:

```
Does the engine start (yes/no)? no
Does the engine rotate (yes/no)? yes
Does the tank have any gas in it (yes/no)? no

Suggested Repair:

 Add gas.
```

Puedes probar a realizar otras dos interacciones para familiarizarte con el flujo del sistema (entrada, razonamiento y salida).

No nos detendremos mucho en la definición de funciones auxiliares (**DEFFUNCTIONS**), cuya utilidad es simplemente facilitar la entrada de datos. La función (ask-question) recibe el texto de una pregunta como primer parámetro y un número ilimitado de parámetros que corresponden a las respuestas aceptables. La función yes-or-no-p sólo admite cuatro respuestas válidas (yes no y n) y devuelve verdadero o falso según la respuesta sea afirmativa o negativa.

El programa se divide en tres grupos de reglas:

- Reglas que hacen preguntas al usuario (**QUERY RULES**) y determinan el diagnóstico
- Reglas que a partir del diagnóstico establecido establecen la reparación a realizar (**REPAIR RULES**)
- Reglas que presentan el sistema y escriben el resultado final (**STARTUP AND CONCLUSION RULES).** El flujo de estas reglas se controla utilizando la saliencia y la existencia en la memoria de trabajo del hecho que indica que se ha obtenido una reparación.

Guión de prácticas 2.1 CLIPS

Para establecer el diagnóstico, el sistema utiliza los siguientes hechos y posibles valores (QUERY RULES):

- `engine-starts` `yes |no`
- `runs-normally` `yes |no`
- `engine-rotates` `yes |no`
- `engine-sluggish` `yes |no`
- `engine-misfires` `yes |no`
- `engine-knocks` `yes |no`
- `tank-has-gas` `yes |no`
- `battery-has-charge` `yes|no`
- `point-surface-state` `normal| contaminated | burned`
- `conductivity-test-positive` `yes |no`

Cuestiones 3 (ver anexo)

Entrega de la práctica

La fecha límite de entrega será la indicada en el aula virtual. Este ejercicio se evaluará sobre un máximo de diez puntos mediante un test de autoevaluación.

La entrega consistirá en un único fichero comprimido que contenga todos los ficheros con la resolución de los ejercicios indicados en el guion:

- Un documento PDF con las respuestas a las cuestiones formuladas en los apartados. El documento contendrá únicamente el Anexo, con las preguntas y las respuestas (no el guion completo de la práctica).
- Todos los ficheros **`.clp`** y **`.bat`** correspondientes al ejercicio 2.4

Guión de prácticas 2.1 CLIPS

Anexo (cuestiones a responder)

Cuestiones 1.1:

- *¿Qué devuelve Clips al añadir un hecho a la Base de Hechos (BH)?*
- *¿Qué ocurre cuando se introduce un hecho "repetido" en la base de hechos (***apellido-1 Perez***)?*

Cuestiones 1.2:

¿Se ha activado alguna regla? ¿Qué hechos activan cada regla?

Cuestiones 1.3:

- *¿Qué regla se ejecuta en primer lugar? ¿Por qué?*
- *¿Qué pasa si reiniciamos con* `(clear)`?

Cuestiones 1.4:

- *Si se introducen los hechos con (*`deffacts`*), y se carga el fichero ¿qué ocurre en la BH y en la Agenda?*
- *¿Qué ocurre en la BH y en la Agenda al ejecutar* `(reset)`*?*
- *¿Cuál es el primer hecho que se ha almacenado en la BH?*

Cuestiones 1.5

¿Qué devuelve el programa cuando el agente es un coche y el semáforo para los coches está en rojo? Copia y pega el resultado mostrado en la Dialog Window tras ejecutar el programa.

Guión de prácticas 2.1 CLIPS

Cuestiones 2.3

Utilizando el sistema, indicar la respuesta a las cuestiones siguientes:

1. ¿Quiénes son los hermanos, tíos, sobrinos, primos y abuelos de Archie?
2. ¿Quiénes son abuelos y de quién?
3. ¿Quiénes son tíos y de quién?

Cuestiones 2.4

Incluir los hechos y, de ser necesario, las nuevas reglas que tengan en cuenta la nueva situación e indicar cuáles son ahora las respuestas a las cuestiones anteriores.

Cuestiones 3.

1. Analiza el orden de ejecución de las reglas del bloque QUERY RULES y represéntalo gráficamente en forma de árbol, donde los nodos son las diferentes reglas (determine-engine-state, determine-runs-normally, …)

2. Justifique por qué todas las reglas del bloque QUERY RULES incluyen el antecedente (not (repair ?)) y si este es necesario o podría prescindirse de él.

3. Identifica qué reglas del sistema utilizan la variable saliencia. Revisando la documentación de CLIPS y otras fuentes investiga acerca del concepto de saliencia y justifica el papel que juega dicha variable y los valores elegidos en las reglas del sistema.