DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Sistemas Inteligentes de Gestión

Relación de ejercicios

CLIPS

Sistemas expertos basados en reglas con encadenamiento hacia adelante

© *Juan Carlos Cubero & Fernando Berzal*

ENTREGA DE LA PRÁCTICA

```
clips.doc
1_personas.clp
1_personas.datos.clp
2_alarma.clp
2_alarma.datos.clp
3_eccema.clp
3_eccema.datos.clp
4_IRPF.clp
5_eccema.clp
5_eccema.datos.clp
6_minimo.clp
7_suma.clp
8_sustituciones.clp
9_union.clp
10_dependencia.clp
10_dependencia.datos.clp
11_empaquetado.clp
11_empaquetado.datos.clp
12_bandera.clp
12_bandera.datos.clp
```

PREPARATIVOS PARA LA REALIZACIÓN DE LA PRÁCTICA:

Para instalar CLIPS en casa, basta con que descarguemos su programa de instalación desde la página web oficial de CLIPS: http://clipsrules.sourceforge.net/

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

Para cargar un conjunto de reglas o hechos desde un fichero, seleccionaremos la opción *File* > *Load*, escogeremos dicho fichero y luego ejecutaremos `(reset)` para cargar los deffacts definidos en el fichero.

Cada vez que se haga este proceso, previamente, conviene limpiar el entorno de CLIPS con `(clear)`, pues en la base de conocimiento pueden quedar hechos y reglas de la sesión de trabajo anterior.

Para automatizar el proceso, podemos crear una macro en CLIPS. Para ello, creamos un fichero con extensión `.bat` para cada ejercicio, al que, por ejemplo, le podemos dar 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 *File* > *Load batch*.

Para ver en cada momento la agenda de CLIPS (las activaciones de las reglas aplicables en cada momento), seleccionamos *Window* >*Agenda*.

Para ver los hechos, seleccionamos *Window* >*Facts*.

Ambas ventanas se crean dentro del espacio de trabajo de CLIPS, por lo que, para poder verlas cómodamente, habrá que agrandar la ventana principal lo suficiente.

Ejercicios tipo C

Ejercicio 1

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

```
(defrule Regla1
      (EsPadre Pedro)
      =>
      (assert (QuiereASusHijos Pedro)))
```

Cree el fichero `1_inicio.bat` correspondiente
(para cargar el fichero `1_personas.clp`) .

Añada desde la línea de comandos el hecho `(EsPadre Pedro)`.

Ejecute con `(run 1).`

Fuerze distintos errores de compilación (como, por ejemplo, quitar algún paréntesis, escribir 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 `->`…).

Resetee CLIPS y vuelva a cargar el fichero con la regla anterior. Ahora, haremos lo mismo que antes, pero añadiendo el hecho desde un fichero en vez de hacerlo desde la línea de comandos. Cree un fichero nuevo llamado `1_personas.datos.clp` que contenga:

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

Cargue ambos ficheros desde `1_inicio.bat`, y ejecute.

Por último, resetee nuevamente CLIPS y modifique la regla como sigue:

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

Si cargamos esta regla y el fichero de datos anterior, ¿cuantas activaciones hay de la regla? Pruebe `(run 1)`, o bien `CTRL+T`, y `(reset)`; `(run 2)` y `(reset)` y, finalmente `(run)` o bien `CTRL+R`.

Ejercicio 2

Implemente el ejemplo de la alarma del tutorial de CLIPS, creando para ello los ficheros `2_alarma.clp` y `2_alarma.datos.clp`

Ejercicio 3

Implemente el ejemplo de los eccemas del tutorial de CLIPS en los ficheros `3_eccema.clp` y `3_eccema.datos.clp`.

Además de lo que aparece en el tutorial:

1. Añada reglas para que, una vez obtenido un diagnóstico, lo muestre en pantalla.

2. La correcta aplicación de la regla `DiagnosticoEccema` requiere que los síntomas `picor` y `vesículas` se presenten en ese orden. Introduczca las modificaciones necesarias para que no importe el orden en el que aparecen los síntomas en el vector `DatosExploracion`.

Ejercicio 4

Defina en un fichero `4_IRPF.clp` un template llamado `Persona` con campos `Nombre`, `Edad`, `NombreConyuge`, `PosicionEconomica` y `Salario`.

Declare varios datos con un `deffacts` (al menos, tres personas de 40 años y otras tres de 60 años) y defina reglas que permitan hacer lo siguiente:

a. Mostrar en pantalla los nombres de todas las personas de 60 años.

b. Mostrar en pantalla el nombre y salario de las personas de 40 años.

c. Mostrar en pantalla los datos de todas las personas.

d. Mostrar en pantalla el nombre de aquellas personas cuyo cónyuge tenga una posición económica desahogada.

e. Por cada persona cuyo cónyuge tenga una posición económica desahogada, añadir a la memoria de trabajo un vector ordenado de características de la forma `(DatosFiscales <Nombre> ConyugeDesahogado)` .

f. Borrar de la memoria de trabajo aquellas personas que tengan un cónyuge con una posición económica desahogada. Recomendación: Use los hechos `(DatosFiscales)` del apartado anterior.

g. Borrar de la memoria de trabajo aquellas personas que tengan una posición económica desahogada.

Ejercicio 5

Implementad el ejemplo del diagnóstico de eccemas sobre varios pacientes, con un fichero `5_eccema.clp` para las reglas y la definición de los registros (templates) y otro fichero `5_eccema.datos.clp` para los hechos (añadidos con un `(deffacts)`).

Utilice los siguiente esquemas:

```
(deftemplate FichaPaciente
      (field Nombre)
      (field Edad)
      (field Casado)
      (field Sexo)
      (field Peso))
(deftemplate DatosExploracion
      (field Nombre)
      (multifield Sintomas)
      (field GravedadAfeccion))
```

a. Añada una regla para el siguiente diagnóstico: Si un paciente tiene los síntomas `picor` y `vesículas`, entonces mostrar un mensaje diciendo que tiene un eccema.

b. Modifique el anterior programa para que, en vez de mostrar el resultado del diagnóstico, añada un hecho con el diagnóstico correspondiente. Añada, además, otra regla que se activará cuando exista un diagnóstico y muestre el correspondiente mensaje en pantalla. Utilize el siguiente esquema para el diagnóstico:

```
(deftemplate Diagnostico
      (field Nombre)
      (field Resultado)
      (field ProximaRevision))
```

c. Añada una regla que añada un vector ordenado del tipo `(Paciente Juan Es_un_bebe)` cuando un paciente tenga una edad menor a dos años.

d. Añada reglas que nos sugieran qué terapia sería más recomendable. Se administrará un corticoides de uso tópico si el enfermo no es un bebé y una crema hidratante en caso contrario, a no ser que la afección sea muy grave, en cuyo caso también aplicaremos corticoides a los bebés. Utilize el siguiente esquema:

```
(deftemplate Terapia
      (field Nombre)
      (field PrincipioActivo)
      (field Posologia))
```

e. Añada reglas que nos muestren en pantalla el resultado de la terapia que sería recomendable administrar.

Ejercicio 6

Escriba un programa para hallar el mínimo elemento de un vector cualquiera de enteros.

Hay que tener en cuenta que, tal y como ocurre con la programación imperativa, no debemos modificar un dato (en nuestro caso el vector) si no se especifica explícitamente en el enunciado del problema.

Incluya las reglas necesarias y los datos usados como batería de pruebas en el mismo fichero: `6_minimo.clp`.

Ejercicio 7

Escriba un programa para sumar los elementos de un vector cualquiera de enteros.

Siga las mismas recomendaciones las indicadas en el ejercicio anterior y guarde su solución (reglas y batería de pruebas) en el fichero `7_suma.clp`.

Ejercicio 8

Supongamos un vector de características de la forma:

```
(palabra <caracteres separados por espacios>)
```

Implemente una solución para el problema de las sustituciones simbólicas, en el que cada carácter puede ser sustituido por una lista de caracteres.

Las reglas de sustitución que deberá utilizar son las siguientes:

```
C → D L
C → B M
B → M M
Z → B B M
```

Por ejemplo, el vector `(palabra B C D)` sería sustituido, aplicando la primera regla, por `(palabra B D L D)`.

> NOTA: Se pretende conseguir que el vector tenga sólo `M`s (aunque esta comprobación se hará, simplemente, viendo la memoria de trabajo en cada momento).

Para realizar la sustitución, se borrará el antiguo hecho y se añadirá uno nuevo.

Compruebe que es un problema irreversible, en el sentido de que la aplicación de una regla puede hacer que no lleguemos a la solución del problema. Esto se debe a que CLIPS emplea una estrategia irrevocable.

Guarde las reglas y los datos en el fichero `8_sustituciones.clp`.

Ejercicios tipo B

Ejercicio 9

Supongamos que tenemos dos vectores ordenados de características de la forma

```
(cadena B C A D E E B C E)
(cadena E E B F D E)
```

Construya un vector que contenga la unión de las letras que aparecen en todos ellos (sin elementos repetidos). Guarde las reglas y los datos en el fichero `9_union.clp`.

Con los vectores anteriores, por ejemplo, el resultado podría ser el siguiente:

```
(union B C A D E F H)
```

NOTA 1: El orden de las letras del vector `union` podría ser distinto.

NOTA 2: No pueden modificarse los vectores originales (cadena …).

Ejercicio 10

Supongamos un esquema relacional dado por el template `Producto`, con tres atributos (fields): `CodigoVendedor`, `CodigoProducto` y `PVPProducto`.

Defina varios hechos válidos para este esquema.

Escriba un programa en CLIPS que indique por pantalla si existe, o no, la dependencia funcional `CodigoProducto` → `PVPProducto`; es decir, que compruebe que si dos códigos de producto coinciden, entonces también coinciden los correspondientes precios de venta al público.

Guarde las reglas en el fichero `10_dependencia.clp`
y los datos en `10_dependencia.datos.clp`

NOTA: Pueden definirse cuantos predicados auxiliares se consideren necesarios.

Ejercicios tipo A

Ejercicio 11

Se desea construir un sistema para empaquetar artículos en cajas de forma automática. Tendremos en cuenta las siguientes suposiciones:

a) Cada artículo debe primero forrarse y, a continuación, empaquetarse en alguna caja (son todas de cartón, aunque de distintas dimensiones) de forma que guardaremos todos los artículos de un mismo tipo juntos; es decir, un artículo frágil no se empaquetará en una caja que ya se ha abierto para meter artículos pesados.

b) Si tenemos una caja ya empezada con algún artículo, seguiremos llenándola con artículos del mismo tipo antes de abrir una caja nueva (insistimos, en una misma caja no meteremos artículos frágiles junto con artículos pesados, por ejemplo).

c) Para simplificar, supondremos que cada artículo y caja tienen unos volúmenes asociados, de forma que un artículo podrá empaquetarse en una caja si el volumen disponible en ésta es mayor que el volumen del artículo.

Utilice los siguientes templates:

```
(deftemplate Articulo
      (field Nombre)
      (field Tipo)           ; Valores permitidos: fragil, pesado
      (field Forrado)        ; Valores permitidos Si - No
      (field Empaquetado)    ; Valores permitidos Si - No
      (field Dimension)      ; Valor numérico de 0 a 200
)

(deftemplate Caja
      (field IdCaja)
      (field Abierta)        ; Valores permitidos Si - No
      (field Empezada)       ; Valores permitidos Si - No
      (field TipoContenido)  ; Valores permitidos: fragil, pesado
      (field EspacioLibre)   ; Valor numérico que indica
                             ; el espacio que todavía queda libre.
                             ; Al principio, contiene la dimensión
                             ; de la caja. Viene en las mismas
                             ; unidades que el field dimensión
                             ; del template Articulo.
)
```

Guarde las reglas en el fichero `11_empaquetado.clp`
y los datos en `11_empaquetado.datos.clp`

Ejercicio 12

Supongamos un template llamado `Pais` con un campo `Nombre` y otro campo `Bandera` de tipo `multifield`, que contendrá una lista con los colores que aparecen en la bandera del país (no necesariamente ordenada).

El usuario asertará en línea de comandos un vector del `tipo (ColoresABuscar Blanco Amarillo)` y el programa debe mostrar en pantalla el nombre de todos aquellos países que contienen una bandera en la que aparezcan todos los colores especificados en el anterior vector (la bandera del país podría contener otros colores más, además de los pedidos).

Cree un fichero de datos con varios ejemplos reales.
http://es.wikipedia.org/wiki/Wikipedia:Banderas_nacionales_y_de_territorios_dependientes

EVALUACIÓN DE LAS PRÁCTICAS

Para cada problema, se crearán los correspondientes ficheros `.clp` con la definición de las reglas y de los hechos.

Al final, también se creará un fichero único llamado `clips.doc` que contenga la solución de todos los ejercicios realizados.