Tema 4: Hechos definidos a partir de plantillas

El constructor deftemplate

Introducción

- Hechos ordenados y no ordenados.
- Plantillas de conceptos. Similaridad con las estructuras.
- Representan conceptos con sus atributos o relaciones entre conceptos.
- Nombre de los campos: slots.

Sintaxis del constructor deftemplate

(**deftemplate** <nombre> [<comentario>]

<definición-casilla>*)

<definición-casilla> ::= <def-simple> | <def-múltiple>

<def-simple> ::= (**slot** <nombre-casilla> <atributo>*)

<def-múltiple> ::= (**multislot** <nombre-casilla> <atributo>*)

Sintaxis del constructor deftemplate

<atributo> ::= <atributo-default> |

<atributo- constraint>

<atributo- default>::= (default ?DERIVE | ?NONE |

<expression>*) | (default-dynamic) <expression>*)

Constructor deftemplate

- Si al definir una plantilla se identifica con el mismo nombre de una ya existente, la plantilla previamente definida desaparecerá.
- Una plantilla no se puede volver a definir mientras se esté usando por un hecho o por un patrón en el antecedente de una regla.

Número y Tipos de campos en una plantilla

- Una plantilla puede tener cualquier número de campos simples o múltiples.
- Al definir los campos de una plantilla, CLIPS siempre obliga que se indique si va a tratar de un campo de tipo *simple* o univaluado, o un campo de tipo *compuesto* o multivaluado.

Ejemplo deftemplate

- ***persona.clp***

(**deftemplate** persona “datos de una persona”
  (**slot** nombre)
  (**slot** edad)
  (**multislot** direccion))

- Cargar *persona.clp* y afirmar los hechos:

  (assert (persona (edad 34)))
  (assert (persona (nombre juan)))
  (assert (persona (nombre juan) (edad 34)
  (direccion Avda de Cervantes)))

Propiedades de los slots

Valor por defecto.

- Propiedad **<default-attribute>** puede ser de dos tipos:
  - Propiedad **default**. *?DERIVE*. *?NONE*.
  - Propiedad **default-dynamic**.
- Restricciones de los campos por defecto para la comparación de patrones.

Ejemplo propiedades por defecto

- ***dato.clp***

```
(deftemplate dato
  (slot w (default ?NONE))
  (slot x (default ?DERIVE))
  (slot y (default (gensym*) ))
  (slot z (default-dynamic (gensym*) )) )
```

- Cargar *dato.clp* y afirmar los hechos:

```
(assert (dato))
(assert (dato (w 3)))
(assert (dato (w 4)))
```

Propiedades de los slots

- Propiedad **type** de los slots:
  - SYMBOL
  - STRING
  - NUMBER
  - INTEGER
  - FLOAT

Ejemplo

Plantilla denominada *persona* con las siguientes casillas simples: *nombre*, *apellidos*, *color-ojos, altura* y *edad*, indicando los tipos de cada campo (*nombre* y *apellidos* de tipo cadena, *color-ojos* de tipo símbolo, *altura* de tipo float y *edad* de tipo entero) y un valor por defecto para todos los campos excepto al nombre y apellidos.

```
(deftemplate persona "Detalles de una persona"
  (slot nombre (type STRING) (default ?NONE))
  (slot apellidos (type STRING) (default ?NONE))
  (slot edad (type INTEGER) (default 20))
  (slot color-ojos (type SYMBOL) (default negro))
  (slot altura (type FLOAT) (default 1.65))
)
(assert (persona (nombre "Isabel")
   (apellidos "Perez Ramirez")) )
```

Propiedades de los slots

- Propiedad para especificar valores permitidos de un slot.

allowed-symbols	rico pobre
allowed-strings	"Ricardo" "Juan" "Pedro"
allowed-numbers	1 2 3 4.5 -2.01 1.3e-4
allowed-integers	-100 53
allowed-floats	-2.3 1.0 300.00056
allowed-values	"Ricardo" rico 99 1.e9

Ejemplo Propiedades type y allowed

```
(deftemplate objeto
  (slot nombre
     (type SYMBOL)
     (default ?DERIVE))
  (slot peso
     (allowed-values ligero pesado)
     (default ligero))
  (slot contenidos
     (type SYMBOL)
     (default ?DERIVE)) )
```

Ejemplo

Modifique la plantilla persona para que admita sólo los siguientes colores de ojos: negro, gris, marron, verde, azul

```
(deftemplate persona "Detalles de una persona"
  (slot nombre (type STRING) (default ?NONE))
  (slot apellidos (type STRING) (default ?NONE))
  (slot edad (type INTEGER) (default 20))
  (slot color-ojos (default negro)
   (allowed-symbols negro gris marron verde
azul))
  (slot altura (type FLOAT) (default 1.65))
)
(assert (persona (nombre "Isabel")
   (apellidos "Perez Ramirez")
   (color-ojos verde) ) )
```

Propiedades de los slots

- Propiedad de slot para especificar un rango de valores permitidos.

```
(deftemplate persona
  (slot nombre
        (type STRING)
        (default ?DERIVE))
  (slot edad
        (type FLOAT)
        (default (* 2.0 3.4))
        (range 0.0 100.0)) )
```

Propiedades de los slots

- Propiedad de slot para restringuir los valores de los slots de una plantilla.
- Verificación de las restriciones estáticas (por defecto) o dinámicas.
- Función *set-static-constraint-checking* y *set-dynamic-constraint-checking* para cambiar el chequeo.

Ejemplo

Modifique la plantilla persona para que admita edades sólo en el intervalo [1, 130]. Fije también el rango de la altura permitida.

```
(deftemplate persona "Detalles de una persona"
  (slot nombre (type STRING) (default ?NONE))
  (slot apellidos (type STRING) (default ?NONE))
  (slot edad (type INTEGER) (default 20)
    (range 1 130) )
  (slot color-ojos (type SYMBOL) (default negro)
   (allowed-values negro gris marron verde azul))
  (slot altura (type FLOAT) (default 1.65)
    (range 0.1 3.0) ) )
```

```
(assert (persona (nombre "Isabel")
   (apellidos "Perez Ramirez")
   (color-ojos verde) (edad 21) (altura 1.7) ) )
```

Orden (list-deftemplates)

- Lista las plantillas que están siendo usadas por el sistema.

CLIPS> *(list-deftemplates)*

initial-fact

persona

For a total of 2 deftemplates.

Orden (ppdeftemplate)

- La orden (***ppdeftemplate*** nombre) muestra una plantilla ya definida.

Orden (undeftemplate)

- La orden (***undeftemplate*** nombre) elimina una plantilla existente.

CLISP> (undeftemplate persona)

CLIPS> (list-deftemplates)

initial-fact

For a total of 1 deftemplates.

CLIPS>