Tema 2: Visión general de CLIPS

Nociones básicas y elementos fundamentales

¿Qué es CLIPS?

- CLIPS (C Language Integrated Production System) es:
  - Un lenguaje de programación
  - Una herramienta

  para el desarrollo de sistemas expertos.

Componentes de CLIPS

- Intérprete
- Interface interactivo
- Facilidades de depuración
- Shell
  - Base de hechos
  - Base de conocimiento
  - Motor de inferencia

Uso de CLIPS

- Procesamiento interactivo
- Procesamiento por lotes
- SE empotrados

Elementos básicos de programación

- Tipos de datos
- Funciones
- Constructores

Tipos de datos

- Entero
- Número en coma flotante

} Información numérica

- Símbolo
- Cadena

} Información simbólica

- Dirección externa
- Dirección de hecho
- Nombre de instancia
- Dirección de instancia

Tipos de datos: otros conceptos

- Campo
- Valor
  - Monocampo
  - Multicampo
- Constante
- Variable

Ejemplos de valores

- Valores unicampo (constantes):

```
Perro
“Juan Manuel”
5
23.8
```

- Valores multicampo:

```
(Perro “Juan Manuel” 5 23.8)
(Perro)
()
```

Variables

Sintaxis	Ámbito	Valor mono/multicampo
?<nombre>	local	monocampo
$?<nombre>	local	multicampo
?*<nombre>*	global	ambos

Funciones

- Funciones definidas por el usuario
- Funciones definidas por el sistema

} Lenguaje externo

- Funciones convencionales
- Funciones genéricas

} CLIPS

Constructores

- deftemplate
- deffacts
- defrule

  -
  -
  -

Representación de la información

- Hechos
- Variables globales
- Objetos

Hechos

assert, facts, retract, deffacts

- Hechos ordenados
- Hechos no ordenados

deftemplate, modify, duplicate

- Dirección de hecho `fact-0`
- Índice de hecho `0`
- Identificador de hecho `f-0`

Hechos: ejemplos

- Ordenados

  `(leche arroz huevos)`

  `(perro gato vaca)`

  `(IA 2 Pedro)`

- No ordenados

  `(coche (marca Ford) (modelo focus) (color gris))`

  `(cliente (nombre “Juan Pérez”) (tlf 957123456))`

Hechos: comandos

```
(assert <hecho>+)
(facts [<inicio> [<final> [máximo]]])
(retract <índice>+ | *)
(modify <índice> <nueva-casilla>+)
(duplicate <índice> <nueva-casilla>+)
<nueva-casilla>::=
  (<nombre-casilla> <valor-casilla>)
```

Hechos: comandos

```
(deftemplate    <nombre> [<comentario>]
                <definición-casilla>*)
<definición-casilla> ::= <def-casilla-simple> |
  <def-casilla-múltiple>
<def-casilla-simple> ::= (slot <nombre-casilla>)
<def-casilla-múltiple> ::= (multislot <nombre-
  casilla>)
```

Hechos: ejemplos

```
CLIPS> (clear)
CLIPS> (assert (color rojo))
<Fact-0>
CLIPS> (assert (color azul) (valor (+ 3 4)))
<Fact-2>
CLIPS> (assert (color rojo))
FALSE
CLIPS> (deftemplate estado (slot temperatura) (slot presion))
CLIPS> (assert (estado (temperatura alta) (presion baja)))
<Fact-3>
CLIPS> (facts)
f-0     (color rojo)
f-1     (color azul)
f-2     (valor 7)
f-3     (estado (temperatura alta) (presion baja))
For a total of 4 facts.
```

Hechos: ejercicio 1

- Muestra los hechos con índice >=1
- Muestra los hechos 1 a 2
- Crea un nuevo hecho que sea como el 3 pero con *(temperatura baja)*
- Elimina el hecho 1
- Añade un hecho *(color verde)*
- Elimina todos los hechos

Representación del conocimiento

- Representación heurística
  - Reglas `defrule`
- Representación procedimental
  - Funciones `deffunction, defgeneric, defmethod`
  - Objetos

Reglas: comandos

```
(defrule <nombre-regla>
  [<comentario>]
  [<propiedades>]
  <elemento-condicional>*
  =>
  <acción>*)
```

Reglas: ejemplo

```
CLIPS> (clear)
CLIPS> (defrule PuedoCruzar
         (Semaforo verde)
         =>
         (printout t "Puedo cruzar" crlf))
CLIPS> (defrule NoPuedoCruzar
         (Semaforo rojo)
         =>
         (printout t "No puedo cruzar" crlf))
CLIPS> (assert (Semaforo rojo))
<Fact-0>
CLIPS> (agenda)
0      NoPuedoCruzar: f-0
For a total of 1 activation.
CLIPS> (run)
No puedo cruzar
```

Reglas: ejercicio 2

- Afirma el hecho de que el semáforo está verde
- Ejecuta las reglas activadas

Reglas: ejemplo

```
CLIPS> (clear)
CLIPS> (defrule ElectricidadOff (Electricidad apagada) =>
   (assert (Comida estropeada)))
CLIPS> (defrule FrigoAbierto (Frigorifico puerta abierta) =>
   (assert (Comida estropeada)))
CLIPS> (defrule AlSuper (Comida estropeada) =>
   (assert (Necesito-ir-a supermercado)))
CLIPS> (assert (Frigorifico puerta abierta))
<Fact-0>
CLIPS> (facts)
f-0     (Frigorifico puerta abierta)
For a total of 1 fact.
CLIPS> (run)
CLIPS> (facts)
f-0     (Frigorifico puerta abierta)
f-1     (Comida estropeada)
f-2     (Necesito-ir-a supermercado)
For a total of 3 facts.
```