INTELIGENCIA ARTIFICIAL

Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería de Sistemas e Informática

Rolando A. Maguiña Pérez

Agenda

CLIPS

- Fundamentos (características, representación de conocimiento, etc.)
- Comandos básicos
- Hechos
- Reglas

Introducción a CLIPS – El sistema CLIPS

❖ CLIPS es una herramienta para SEs desarrollada por el Área de Tecnología de Software del Centro Espacial Lyndon B. Johnson/NASA

CLIPS: **C** **L**anguage **I**ntegrated **P**roduction **S**ystem

❖ Define un lenguaje que permite la representación de conocimiento declarativo y procedimental

❖ Su lenguaje permite representar reglas de producción y frames

❖ Tiene implementado un mecanismo de inferencia con razonamiento hacia adelante

Introducción a CLIPS – El sistema CLIPS

- Núcleo del CLIPS proporciona los elementos básicos de un SE:

  Memoria de trabajo (MT) (memoria global de datos)

  Base de Conocimiento

  Mecanismo de inferencia (encadenamiento progresivo)

Introducción a CLIPS – El lenguaje de CLIPS

- El lenguaje CLIPS deriva su sintaxis del lenguaje LISP
- Se trata de un lenguaje parentizado con notación prefija y case-sensitive
- Los tipos de datos predefinidos a usar son: reales, enteros, strings, símbolos, apuntador a hechos, nombre de instancia y apuntador a instancia
- El lenguaje de CLIPS permite tres paradigmas de programación: lenguaje de reglas, lenguaje funcional, lenguaje orientado a objetos

Representación del conocimiento en CLIPS

- Reglas (defrules)
- Funciones (deffunctions y funciones genéricas)
- Programación orientada a objetos (POO)

Introducción a CLIPS – Características

- Un entorno de desarrollo completo:
  - Tres formalismos para la representación del conocimiento
  - Motor de inferencia con encadenamiento progresivo
  - Implementación en C del algoritmo de RETE (eficiente)
  - Lenguaje de control y de programación al estilo LISP

Elementos básicos para programas

- Tipos de datos
- Constructores
- Funciones

Elementos básicos para programas

- Tipos de datos

  Símbolos (SYMBOL)
  Cadenas de caracteres (STRING)
  Enteros (INTEGER)
  Coma flotante (FLOAT)
  Direcciones de la MT (externas, de hechos, de instancias)

Elementos básicos para programas

- Constructores

  Plantillas (deftemplate)
  Reglas (defrule)
  Especificar un conjunto de hechos como conocimiento inicial (deffacts)

Elementos básicos para programas

- Funciones
  - Funciones (internas) del sistema
  - Funciones definidas por el usuario
    - fcs externas
    - fcs definidas directamente en CLIPS (deffunctions)
    - fcs genéricas

Elementos básicos para programas

- Funciones

EJM.
(* 2 (+ 4 3))

```
(deffunction pide-numero (?pregunta)
  (printout t ?pregunta)
  (bind ?respuesta (read))
  (while (not (numberp ?respuesta)) do
     (printout t ?pregunta)
     (bind ?respuesta (read)))
  ?respuesta); devuelve ?respuesta
```

El lenguaje de reglas de CLIPS – Hechos

- Los dos elementos que permiten representar problemas utilizando reglas de producción son los hechos y las reglas
- Un hecho es una forma básica de alto nivel para representar información
- Un hecho se compone de una serie de campos (un campo es un lugar que lleva asociado un valor)

EJMS.

```
(color bloque1 rojo)
(altitud es 2400 metros)
(alumnos Melendez Carpio Salazar Reyes)
(nombre "Alberto")
(edad 21)
```

El lenguaje de reglas de CLIPS – Hechos

- Los hechos en CLIPS pueden ser de dos tipos:

  ordered facts y deftemplate facts

- Los ordered facts tienen formato libre, no tienen una estructura predefinida, siguen el esquema:

```
(relacion p1 p2 ... pn)
```

- relacion debe ser un símbolo, el resto de parámetros puede ser de cualquier tipo

  EJM.

```
(padre martin carlos)
```

```
(num-cursos roberto 7)
```

El lenguaje de reglas de CLIPS – Hechos

- Los deftemplate facts tienen una estructura predefinida, análogos a representaciones mediante frames

- En estos hechos se definen una serie de campos (slots) que definen su estructura. Cada campo puede tener una serie de restricciones como tipo, cardinalidad y un valor por defecto (cte o una fc para calcularlo)

Estructura:

```
(deftemplate nombre-template "comentario"
    (slot nombre-slot)
    (multislot nombre-slot))
```

El lenguaje de reglas de CLIPS – Hechos

Estructura:

```
(deftemplate nombre-template "comentario"
    (slot nombre-slot)
    (multislot nombre-slot))
```

EJM.

```
(deftemplate estudiante
    (slot nombre (type STRING))
    (slot edad (type INTEGER) (default 16)))
```

```
(deftemplate persona “Una plantilla ejm”
  (slot nombre)
  (slot edad)
  (slot ocupacion))
```

El lenguaje de reglas de CLIPS – Hechos

- Son los datos iniciales del problema y los generados durante el funcionamiento del sistema

- Comando para ingresar hechos: assert

EJM. (assert (altitud es 2400 metros))

- Sintaxis:

(assert <hecho1> <hecho2> …)

EJM.

(assert (bloque2 es rojo) (alumnos Melendez Carpio Salazar Reyes))

El lenguaje de reglas de CLIPS – Hechos

❖ La MT consta de una lista de hechos (fact-list) y de una lista de instancias (instance-list)

❖ Comando para obtener un listado con todos los hechos de la BH: facts

EJM.

```
CLIPS> (assert (bloque2 es rojo)
                    (alumnos Melendez Carpio Salazar Reyes))
CLIPS> (facts)
f-0    (bloque2 es rojo)
f-1    (alumnos Melendez Carpio Salazar Reyes)
```

Hechos

- Posibles valores para los campos de los hechos

Símbolos (SYMBOL)
Cadenas de caracteres (STRING)
Enteros (INTEGER)
Coma flotante (FLOAT)
Direcciones de la MT (de hechos, de instancias, externa)

EJMS.

```
(assert (2400))
(assert (altitud 2400))
(assert (nombre "Alberto"))
(assert (edad 21))
```

Hechos

- Comando para eliminar hechos: retract
  - Sintaxis:

```
(retract <índice-del-hecho>)
```

EJM.

```
CLIPS> (retract 0)
CLIPS> (facts)
f-1    (alumnos Melendez Carpio Salazar Reyes)
```

Comandos básicos para manejo de la MT

(facts) Lista los hechos de la MT	**(instances)** Lista las instancias de la MT
(assert <hecho>) Añade un hecho a la MT.	**(make-instance <instancia>)** Añade una instancia a la MT
(retract <índice-hecho>) Elimina un hecho de la MT	**(unmake-instance <instancia>)** Elimina una instancia a la MT.

(clear) Elimina todos los hechos y construcciones de la MT
(reset) Elimina todos los hechos de la MT, elimina las activaciones de la agenda y restaura las condiciones iniciales: • Añade *initial-fact* e *initial-object* • Añade el conocimiento inicial definido con *deffacts* y *definstances* • Añade las variables globales con su valor inicial. • Fija como módulo principal el módulo *main*

El lenguaje de reglas de CLIPS – Hechos

En CLIPS los hechos se almacenan en la fact-list

Los hechos pueden ser ordenados y no ordenados

Hechos ordenados (Ordered facts)

- El primer campo suele representar una relación entre los restantes campos
- Patrones con uno o varios campos

```
(altitud es 2400 metros)
(alumnos Melendez Carpio Salazar Reyes)
(nombre "Alberto")
(edad 21)
(assert (ejm-hecho1) (ejm-hecho2))
```

Hechos ordenados (Ordered facts)-continuación

- La información se codifica por su posición

```
(altitud es 2400 metros)
(alumnos Melendez Carpio Salazar Reyes)
(nombre "Alberto")
(edad 21)
```

No necesitan plantilla

Hechos no ordenados (Non-Ordered facts)

- Permiten abstraer la estructura de un hecho asignando un nombre a cada campo
- Permiten definir la información en forma de clases de elementos y atributos de esas clases
- El primer nombre del deftemplate se corresponde con el nombre de la clase

Necesitan plantilla

Hechos no ordenados (Non-Ordered facts)

Creación de templates

```
(deftemplate persona
"Una plantilla ejm"
  (slot nombre)
  (slot edad)
  (slot ocupacion))
```

```
(deftemplate <nombre-plantilla>
[<comentario-opcional>]
  <definición-de-slot1>
  <definición-de-slot2>
  …
  <definición-de-slotN>)
```

```
(persona (nombre "Paolo Guerrero") (edad 27)
         (ocupacion futbolista))
```

```
(persona (nombre "Claudia Cisneros") (edad 38)
         (ocupacion periodista))
```

Hechos no ordenados (Non-Ordered facts)

Creación de templates

```
(deftemplate prospecto
    "informacion vital"
  (slot nombre
     (type STRING)
     (default ?DERIVE))
  (slot cualidades
     (type SYMBOL)
     (default rico))
  (slot edad
     (type NUMBER)
     (default 80)))
```

```
(deftemplate <nombre-plantilla>
[<comentario-opcional>]
<definición-de-slot1>
<definición-de-slot2>
...
<definición-de-slotN>)
```

Hechos no ordenados (Non-Ordered facts)

- El orden de los slots no tiene importancia

EJM.

```
(cliente (nombre Juan Perez)
        (id 33435))
(cliente (id 33435)
        (nombre Juan Perez))
(clase (profesor Marta Lopez)
     (estudiantes 30)
     (aula "204"))
```

Hechos no ordenados (Non-Ordered facts)

Creación de templates

EJM.

```
(deftemplate robot
    (slot bandeja
        (type SYMBOL)
        (allowed-values LLENA VACIA)
        (default VACIA))
```

Atributos de los slots

- De restricción: type, allowed-values, range.
- De valor: default (?NONE, ?DERIVE)

EJM.

El problema de la asignación de habitaciones

Hay 4 clases de habitaciones: simples, dobles, triples, y cuádruples.
Es más económico llenar las habitaciones más grandes.
Los estudiantes de una habitación deben ser del mismo sexo.
Los ocupantes de una habitación deben ser todos fumadores o todos no fumadores.

Nombre	Atributos
Estudiante	Nombre
	Sexo
	Fuma?
	Alojado

Nombre	Atributos
Habitación	Número
	Capacidad
	Sexos
	Fuman?
	Plazas-libres
	Ocupantes

El problema de la asignación de habitaciones

EJM.

```
(deftemplate habitación
      (slot numero)
      (slot capacidad)
      (slot sexos)
      (slot fuman?)
      (slot plazas-libres)
      (multislot ocupantes))
```

```
(deftemplate estudiante
      (slot nombre)
      (slot sexo)
      (slot fuma?)
      (slot alojado))
```

Hechos no ordenados (Non-Ordered facts)

Para no tener que teclear los hechos iniciales, podemos introducirlos con el constructor deffacts

Sintaxis:

```
(deffacts <nombre-colección-hechos>
[<comentario-opcional>]
    <patrónRHS1>
    <patrónRHSt2>
    …
    <patrónRHSN>)
```

Hechos no ordenados (Non-Ordered facts)

EJM.

```
(deffacts estudiante "Todos los estudiantes iniciales"
  (estudiante (nombre Juan) (sexo varon) (fuma? no) (alojado no))
  (estudiante (nombre Pepe) (sexo varon) (fuma? si) (alojado no))
  (estudiante (nombre Luisa) (sexo mujer) (fuma? no) (alojado no))
  (estudiante (nombre Pedro) (sexo varon) (fuma? no) (alojado no)))
```

El lenguaje de reglas de CLIPS – Hechos

EJM.

```
(deftemplate persona
  (slot nombre)
  (slot ojos))

(persona (nombre Blanca) (ojos azules)))
```

```
(deffacts personas
  (persona (nombre Ana) (ojos verdes))
  (persona (nombre Jaime) (ojos azules))
  (persona (nombre Juan) (ojos negros))
  (persona (nombre Luis) (ojos negros))
  (persona (nombre Blanca) (ojos azules)))
```

Universidad Politécnica Salesiana
Ing. Fernando Montaluisa

El lenguaje de reglas de CLIPS - reglas

- Las reglas en CLIPS están formadas por:
  - Una parte izquierda (LHS) que define las condiciones a cumplir
  - Una parte derecha (RHS) que define las acciones a realizar

Sintaxis:

```
(defrule nombre_regla "comentario-opcional"
        (patrón-1)
         ...
        (patrón-N)
=>
        (acción-1)
        ...
        (acción M))
```