Resumen CLIPS

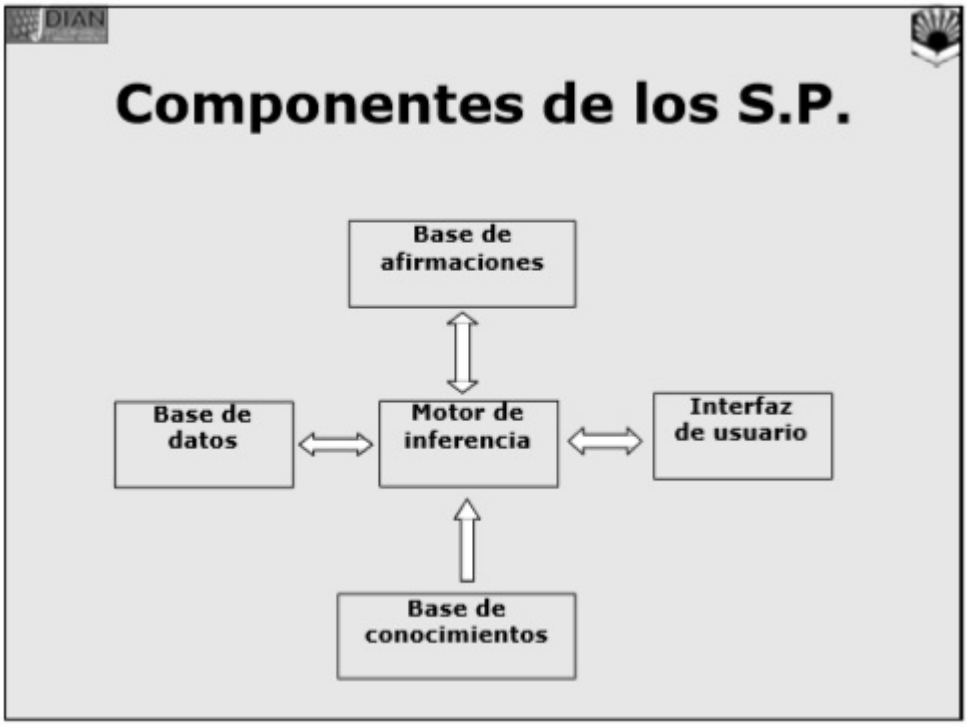
Tema 1. Introducción a los sistemas de producción

Sistemas de Producción

Las Reglas de Producción son reglas del tipo ***Si-Entonces***.

Las características de los Sistemas de producción:

- Se utilizan las reglas para examinar un conjunto de datos y solicitar nueva información hasta llegar a un diagnostico.
- También se denominan Sistemas basados en reglas



Componentes

- **Base de conocimientos**
  Todos los hombres son animales
  Todos los animales respiran
- **Base de afirmaciones**
  Juan es un hombre
- **Inferencia**
  Juan respira

Reglas de producción

La estructura general de las reglas son: ***Antecedente*** → ***Consecuente***. Donde:

- **Antecedente**: Contiene las claúsulas que deben de cumplirse para que la regla pueda ejecutarse
- **Consecuente**: Indica las conclusiones que se deducen de las premisas o las acciones que el sistema debe realizar cuando ejecuta la regla.

Inferencia

- Una regla se ejecuta (dispara) cuando se cumple su antecedente
- Las reglas se ejecutan hacia adelante: **Si se satisface el antecedente se efectúan las acciones del consecuente**
- Tipos de encadenamiento de reglas:
  - Encadenamiento hacia adelante o **basado en datos**.
  - Encadenamiento hacia detrás o basado en objetivos.

Control de Razonamiento
Se encarga de seleccionar una regla cuando hay varias disponibles. Métodos de resolución de conflictos:

- Ordenación de reglas
- Añadir nuevas cláusulas relacionadas con las inferencias
- Control mediante *Agenda*
- Conjunto de Reglas […]

Tema 2. Visión general de CLIPS

Componentes de CLIPS

- Interprete
- Interfaz interactivo
- Facilidades para la depuración
- Elementos de la Shell
  - Lista de hechos
  - Base de conocimientos
  - Motor de inferencia
- Dirigido por datos: Las reglas pueden emparejar con objeto y hechos.

Tipos de datos primitivos

- Entero (*integer*)
- Reales (*float*)
- Simbolo (*symbol*)
- Cadena (*string*)
- Direccion externa (*external-address*)
- Direccion de hecho (*fact-address*)
- Nombre de instancia
- Direccion de instancia (*instance-address*)

Otros conceptos

- **Campo**: Valor que puede tomar una variable
- **Tipos de campos dependiendo del valor que puedan tomar**:
  - Monocampo: Tipos de datos primitvos
  - Multicampo: Varios valores uni-campo
- **Constante**
- **Variable**

Ejemplos

- `?<nombre>` → monocampo (ámbito local)
- `$?<nombre>` → multicampo (ámbito local)
- `?*<nombre>*` → ambos (ámbito global)

Funciones

- **Lenguaje externo**

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

- **Funciones de CLIPS**

    - Funciones
    - Funciones genericas

- **Tipos**

    - Órdenes: Ejecutan una acción
    - Funciones: Devuelven un valor

- **Notación prefija para llamada**

    - ((nombrefuncion argumentos) → `(+(* 3 4) 8 )`)

Constructores

- **defmodule** → Define un módulo
- **defrule** → Define una regla
- **deffacts** → Define un hecho o conjunto de estos
- **deftemplate** → Define una plantilla
- **defglobal** → Define una variable global
- **deffunction** → Define una función
- **defclass** → Define una clase
- **definstances** → Define una instancia
- […]

Tema 3. Representación de Hechos en CLIPS

Representación de la información

Mediante:

- **Hechos**: Ordenados y no ordenados. Indice y dirección.
- **Objetos**: POO. Instancias de objetos.
- **Variables globales**: Constructor defglobal.

Hechos: Ordenes de uso

Órdenes de utilización de hechos:

- **assert** → Introduce datos en la base de hechos.
- **facts** → sirve para ver la base de hechos con formato `f-índice (hecho)`
- **retract** → Elimina hechos de la base de hechos, se debe especificar el nombre o el indice del hecho se pueden eliminar varios hechos de golpe (retrat hecho1 hecho 2 ..) o eliminar todos con (retract *)
- **modify** → Modifica un hecho de la base de hechos
- **duplicate** → Duplica un hecho de la base de hechos
- **deftemplate** → Define una plantilla
- **deffacts** → Define un conjunto de hechos
- **reset** → Añade cada hecho especificado con *deffacts* en la lista de hechos o factlist. Tambien añade el hecho *initial-fact*. Tambien dice que borra hechos e inserta hecho especial (?).
- **clear** → Limpia la base de hechos. Reinicializa el indice de hechos a 0 y elimina la base de conocimiento

Hechos: comandos

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

Hechos: Tipos y Ejemplos

- **Ordenados** → `(casa calle-nueva 32)`
- **No ordenados (Realizados mediante plantillas)**: `(coche (marca ford)(modelo fiesta))`
  - El orden de los campos no es importante
  - Se pueden modificar utilizando las órdenes (modify) y (duplicate)

Hechos: Dirección

```
(defrule comenzar
    ?h <- (iniciar-programa)
  =>
    retract(?h)
    (printout t "Iniciando..." crlf)
)
```

Hechos: Ejemplos de plantillas

```
  (deftemplate nombre-plantilla
      (slot nombre)
      (multislot apellidos)
      (slot DNI)
  )
```