Técnicas de representación empleadas en Sistemas Expertos de CLIPS

Universidad Veracruzana

May 9, 2019

Outline

Sistemas expertos

- Programa que se comporta como un experto humano de algún dominio específico, usando conocimiento referente al dominio mismo.
- Es conveniente dividir a un sistema experto en tres módulos:
  - Base de conocimiento, hechos y reglas del dominio.
  - Máquina de inferencia, procedimientos para emplear el conocimiento.
  - Interfaz de usuario, puente de comunicación entre el usuario y el sistema.

CLIPS

- Desarrollado por el Centro Espacial Johnson de la NASA desde 1985.
- Lenguaje de programación basado en reglas, útil para crear sistemas expertos.
- Modela el conocimiento y la experiencia humana.
- Acrónimo de C Language Integrated Production System.

Aplicaciones sobresalientes

- Sistema de entrenamiento inteligente para controladores de vuelo de transbordadores espaciales (IAAI-89)
- HUB SIAASHING: un sistema basado en el conocimiento para la reducción temporal del horario de las aerolíneas (IAAI-92)
- Sistema experto de procesamiento de seguridad del personal de la NASA (IAAI-96)
- Sistema experto para el reconocimiento de acciones faciales y su intensidad (IAAI-00)
- Sistema híbrido basado en el conocimiento para la optimización multiobjetivo de las operaciones de sistemas de distribución de energía (IAAI-05).

RC en CLIPS

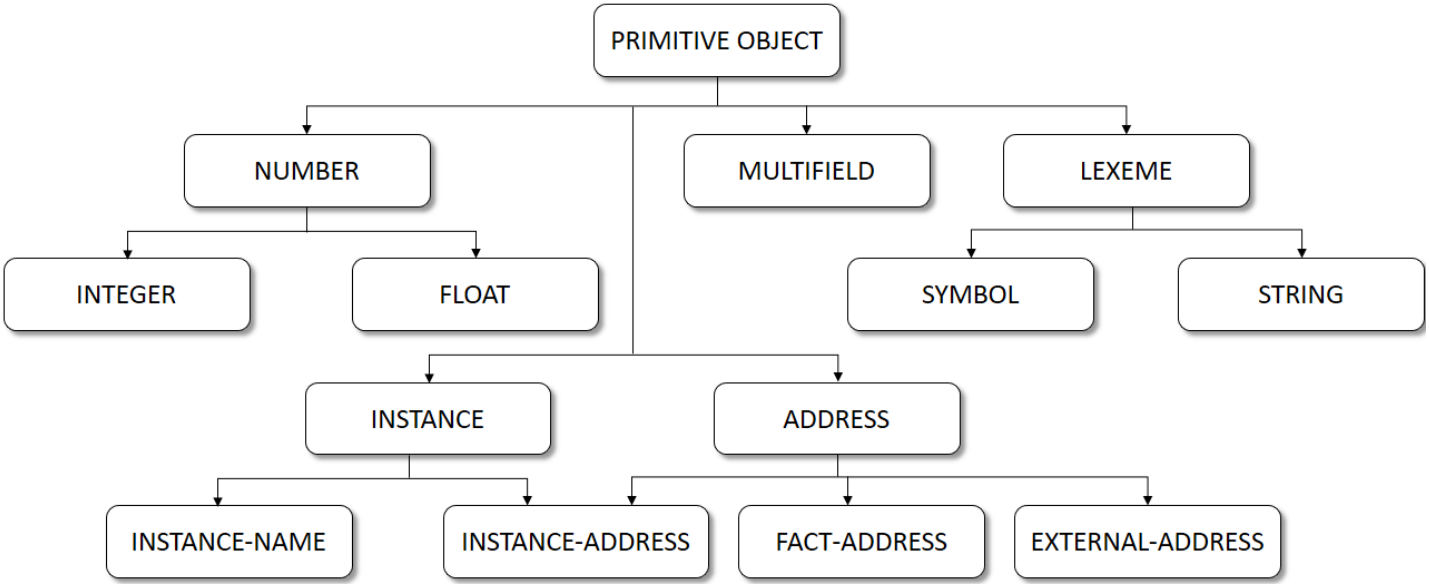
- CLIPS ofrece tres formas de representación:
  - Hechos y reglas, conocimiento heurístico basado en la experiencia.
  - Funciones, conocimiento procedural.
  - Programación orientada a objetos, conocimiento procedural que acepte las funciones de clases, manejadores de mensajes, abstracción, encapsulación, herencia y polimorfismo.

Ejecutando CLIPS

- Todos los comandos en CLIPS comienzan y terminan en paréntesis, y cada uno cuenta con su sintaxis de definida:
  - Los corchetes, '[ ]' indicarán que lo que se encuentra en su interior es opcional.
  - Lo que se encuentre entre los símbolos <y >, indica que debe de sustituirse necesariamente.
  - El símbolo * se asocia a un tipo, e indica que la descripción puede reemplazarse por cero o más ocurrencias del tipo especificado.
  - El símbolo + se asocia a un tipo, e indica que la descripción puede reemplazarse por uno o más ocurrencias del tipo especificado.
  - La barra vertical indica que debe hacerse una elección entre los elementos que se encuentran separador por la barra.
  - El símbolo ::= se utiliza para definir los términos que aparecen en una expresión.

Tipos de Datos

- CLIPS provee ochos tipos de datos primitivos para la representación de la información.



Funciones

- Pieza de código ejecutable identificada por un nombre específico que regresa un valor o realiza una acción.
- El constructor **deffunction** define nuevas funciones en el ambiente CLIPS.
- Los constructores **defgeneric** y **defmethod** permiten al usuario definir funciones genéricas.
- La llamada de funciones en CLIPS usa una notación prefija.

Constructores

- Especifican un tipo de extructura de datos.
- Creación de constructores expliciamentes altera el ambiente de CLIPS.
- Todos los constructores son encerrados en parentesis.
- Varios constructores son definidos en CLIPS: defmodule, defrule, defacts, deftemplate, defglobal, deffclass, definstances, defmessage-handler, defgeneric y defmethod.

Hechos

Comando	Descripción
assert	Agrega un hecho a fact-list.
retract	Remueve un hecho de fact-list.
modify	Modifica un hecho de fact-list.
duplicate	Duplica un hecho de fact-list.
reset	Reinicia los índices.
clear	Reinicia los índices y elimina el conocimiento declarado.

Table 1: Comandos de CLIPS para modificar fact-list.

Hechos

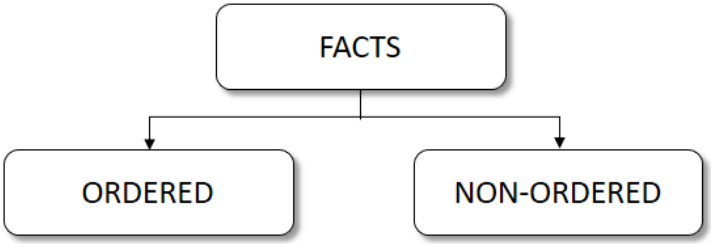


Figure 1: Tipo de hechos.

Hechos

- Datos ordenados
- Datos no ordenados

Hechos

```
(deftemplate <deftemplate-name> [<comment>]
   <slot-definition>*)

<slot-definition> ::= <single-slot-definition> |
                      <multislot-definition>

<single-slot-definition>
               ::= (slot <slot-name>
                         <template-attribute>*)

<multislot-definition>
               ::= (multislot <slot-name>
                              <template-attribute>*)

<template-attribute> ::= <default-attribute> |
                         <constraint-attribute>

<default-attribute>
               ::= (default ?DERIVE | ?NONE | <expression>*) |
                   (default-dynamic <expression>*)
```

Figure 2: Sintaxis de los templares.

Hechos

```
(perro    (nombre "Maya")
          (raza "Labrador")
          (color-ojos marrones)
          (peso 27) )
```

Figure 3: Ejemplo de deftemplate.

Reglas

```
(defrule <rule-name> [<comment>]
   [<declaration>]                ; Rule Properties
   <conditional-element>*         ; Left-Hand Side (LHS)
   =>
   <action>*)                     ; Right-Hand Side (RHS)
```

Figure 4: Sintaxis de las reglas.

Ejecución de reglas

- CLIPS era un lenguaje de reglas basado en el Algoritmo Rete.
- Reconocimiento de patrones de manera eficiente para implementar unsistema de producción
- Quinta edición introdujo la programación imperativa y la programación orientada aobjetos.

Ejecución de reglas

(run [<máximo de reglas a ejecutar>])

1. Mientras el límite de ejecución de reglas no se alcance.
2. La agenda se actualiza atendiendo a la lista de hechos de la memoria de trabajo.
3. Se selecciona la mejor regla atendiendo a la estrategia de resolución de conflictos y las prioridades de las reglas.
4. La instancia seleccionada de una regla se dispara, y es eliminada de la agenda.
5. Regresar al paso 1.

Objetos

- COOL - Lenguaje orientado a objetos de CLIPS por sus siglas en inglés.
- Características de muchos sistemas OOP diferentes y nuevas ideas.
- Empleo de los constructores defclass y defmessage-handler.

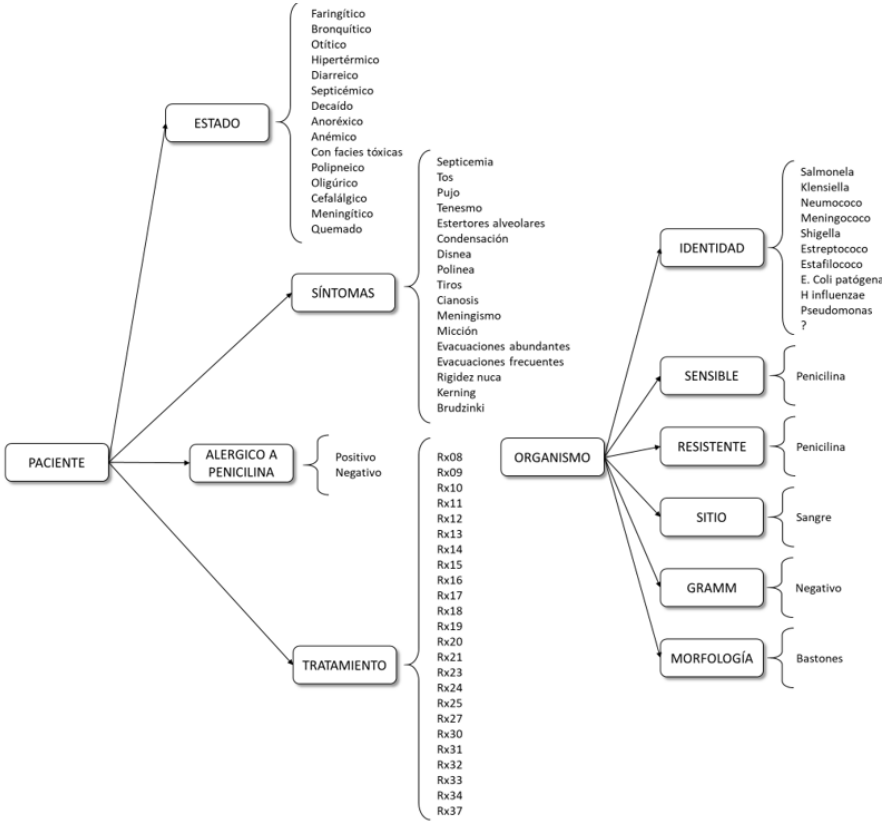
Defclass

```
(defclass <name> [<comment>]
  (is-a <superclass-name>+)
  [<role>]
  [<pattern-match-role>]
  <slot>*
  <handler-documentation>*)

<role>  ::= (role concrete | abstract)

<pattern-match-role>
        ::= (pattern-match reactive | non-reactive)

<slot>  ::= (slot <name> <facet>*) |
            (single-slot <name> <facet>*) |
            (multislot <name> <facet>*)

<facet> ::=  <default-facet> | <storage-facet> |
             <access-facet> | <propagation-facet> |
             <source-facet> | <pattern-match-facet> |
             <visibility-facet> | <create-accessor-facet>
             <override-message-facet> | <constraint-attributes>
```

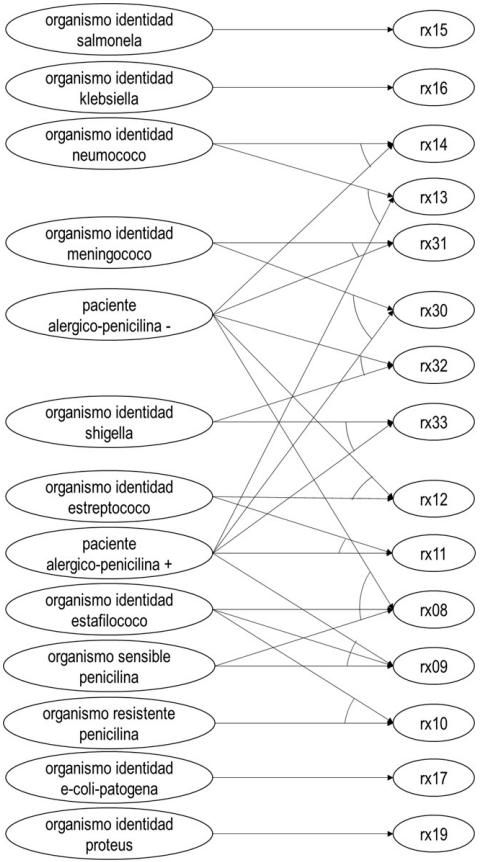
Caso Práctico

- Exponer las diversas técnicas de representación del conocimiento que pueden ser empleadas en los sistemas expertos.
- Caso práctico derivado de una base de conocimiento médica.
- Se busca medicar a un paciente dado el cuadro clínico que este presenta, tomando encuenta tanto aspectos personales como su estado de salud y sintomas, como aspectos relacionado con organismos bacterianos como su morfología y dónde se localiza.

Relaciones


PACIENTE
ESTADO
Faringítico
Bronquítico
Otítico
Hipertérmico
Diarreico
Septicémico
Decaído
Anoréxico
Anémico
Con facies tóxicas
Polipneico
Oligúrico
Cefalálgico
Meningítico
Quemado
SÍNTOMAS
Septicemia
Tos
Pujo
Tenesmo
Estertores alveolares
Condensación
Disnea
Polinea
Tiros
Cianosis
Meningismo
Micción
Evacuaciones abundantes
Evacuaciones frecuentes
Rigidez nuca
Kerning
Brudzinki
ALERGICO A PENICILINA
Positivo
Negativo
TRATAMIENTO
Rx08
Rx09
Rx10
Rx11
Rx12
Rx13
Rx14
Rx15
Rx16
Rx17
Rx18
Rx19
Rx20
Rx21
Rx23
Rx24
Rx25
Rx27
Rx30
Rx31
Rx32
Rx33
Rx34
Rx37
ORGANISMO
IDENTIDAD
Salmonela
Klensiella
Neumococo
Meningococo
Shigella
Estreptococo
Estafilococo
E. Coli patógena
H influenzae
Pseudomonas
?
SENSIBLE
Penicilina
RESISTENTE
Penicilina
SITIO
Sangre
GRAMM
Negativo
MORFOLOGÍA
Bastones

AND/OR


organismo identidad salmonela
organismo identidad klebsiella
organismo identidad neumococo
organismo identidad meningococo
paciente alergico-penicilina -
organismo identidad shigella
organismo identidad estreptococo
paciente alergico-penicilina +
organismo identidad estafilococo
organismo sensible penicilina
organismo resistente penicilina
organismo identidad e-coli-patogena
organismo identidad proteus
rx15
rx16
rx14
rx13
rx31
rx30
rx32
rx33
rx12
rx11
rx08
rx09
rx10
rx17
rx19

AND/OR


paciente decaido +
paciente anoréxico +
paciente anemico +
paciente con-facies-toxicas +
paciente hipertérmico (inicial excepto rn y lactantes)
paciente polipneico (sin relacion con la hipertermia)
paciente oligurico +
paciente rigidez-nuca +
paciente kerning +
paciente brudzinski +
paciente disnea +
paciente polipnea +
paciente tiros +
paciente cianosis +
organismo sitio sangre
organismo gramm -
organismo morfologia bastones
paciente quemado +
paciente septicémico +
paciente meningismo +
paciente cefalalgico +
paciente hipertermico +
paciente hipertermia prolongada
paciente micción alterada
paciente diarreico +
paciente tos +
paciente insuf-resp +
paciente estertores-alvolares +
organismo identidad h-influenza
organismo identidad pseudomonas
organismo identidad ?
paciente meningitico +
Paciente infec-urinaria +
paciente faringitico +
paciente bronquitico +
paciente otitico +
paciente diarr-bact +
paciente neumonico +
paciente evac_abundantes +
paciente evac_frecuentes +
paciente pujo -
paciente tenesmo -
rx20
rx18
rx25
rx34
rx37
rx21
rx23
rx24
rx27

Representación Simple

Reglas

```
(defrule regla012
        (organismo identidad estreptococo)
        (paciente alergico-penicilina -)
=>
        (assert (paciente rx12 penicilina-g 5 eritromicina 4 lincomicina 4))
        (printout t "paciente rx12 penicilina-g 5 eritromicina 4 lincomicina 4" crlf))
```

Figure 5: Ejemplo de Regla

Ejecución

```
1 CLIPS> (load /home/arturo/GitHub/rc-mia/tarea2/rules.clp)
2 CLIPS> (assert (organismo identidad estreptococo))
3 <Fact-0>
4 CLIPS> (assert (paciente alergico-penicilina -))
5 <Fact-1>
6 CLIPS> (run)
7 paciente rx12 penicilina-g 5 eritromicina 4 lincomicina 4
```

Grados de Certeza

Motivación

- Muy alto
- Frío
- Caliente
- Muy oscuro
- Poco probable

Reglas

```
1
2  ;Plantilla para almacenar la informacion de los organismos.
3
4  (deftemplate organismo
5
6          (slot identidad
7          (default x))
8
9          (slot sensible
10         (default x))
11
12         (slot resistente
13         (default x))
14
15         (slot sitio
16         (default x))
17
18         (slot gramm
19         (default x))
20
21         (slot morfologia
22         (default x)))
```

Reglas

```
1
2  ;Plantilla para almacenar la informacion de los pacientes.
3
4  (deftemplate paciente
5
6          (slot alergico-penicilina
7          (default x))
8
9          (slot hipertermia
10         (default x))
11
12         (slot miccion
13         (default x))
14
15         (slot faringitico
16         (default x))
17
18         (slot bronquitico
19         (default x))
20
21         (slot otitico
22         (default x))
```

Reglas

```
1
2  ;Si se pregunta por una o mas instancias de la regla, esta asignara
3  un valor de certeza (porcentaje) y las sumara para conocer cual es
4  la certeza de que dichas instancias provoquen un determinado efecto.
5
6  (defrule regla008
7     (organismo (identidad ?estafilococo))
8     (organismo (sensible ?penicilina))
9     (paciente (alergico-penicilina ?negativo))
0
1     =>
2
3     (progn
4     (bind ?certeza 0)
5     (if (eq ?estafilococo estafilococo) then
6          (bind ?certeza (+ ?certeza 23)))
7     (if (eq ?penicilina penicilina) then
8          (bind ?certeza (+ ?certeza 12)))
9     (if (eq ?negativo negativo) then
0          (bind ?certeza (+ ?certeza 36)))
1     (if (> ?certeza 0) then
2          (printout t "El␣tratammiento␣a␣seguir␣es:
3  ␣␣␣␣penicilina-g␣5.␣Con␣una␣certeza␣de:␣" ?certeza "%" crlf))
4     ))
```

Reglas

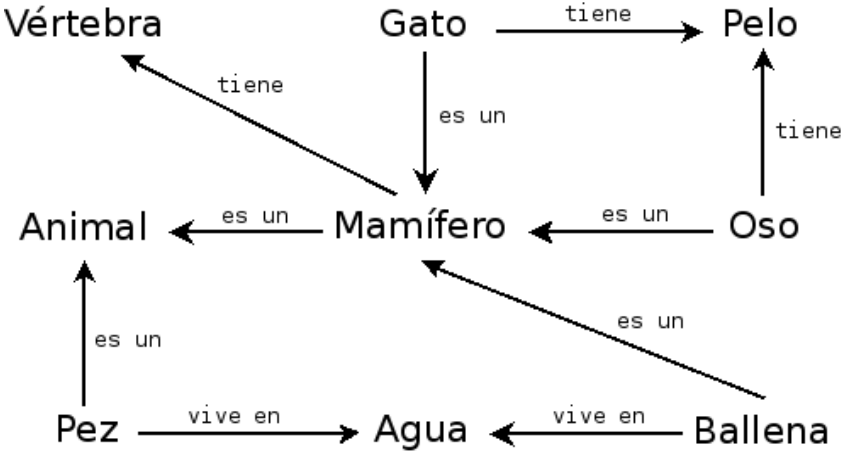
```
1
2  ;Algunas enfermedades o tratamientos son causados por un solo hecho,
3  por lo  que se podria pensar que dado ese hecho,
4  la certeza de que se de el tratamiento o enfermedad es alta.
5
6  (defrule regla015
7     (organismo (identidad ?salmonela))
8  =>
9     (progn
0     (bind ?certeza 0)
1     (if (eq ?salmonela salmonela) then
2          (bind ?certeza (+ ?certeza 90)))
3     (if (> ?certeza 0) then
4          (printout t "El␣tratammiento␣a␣seguir␣es:
5  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣cloromicetina␣5␣ampicilina␣5.
6  ␣␣␣␣␣␣␣␣␣␣␣␣␣␣Con␣una␣certeza␣de:␣" ?certeza "%" crlf))
7     ))
```

Ejecución

```
1
2  ;Algunos hechos tienen mas de un efecto.
3
4  CLIPS> (assert (organismo (identidad estafilococo)
5  (sensible penicilina)) (paciente))
6  <Fact-1>
7  CLIPS> (run)
8  El tratammiento a seguir es: penicilina-g 5.
9  Con una certeza de: 35%
10 El tratammiento a seguir es: eritromicina.
11 Con una certeza de: 67%
12 El tratammiento a seguir es: cefalotina cefaloridina 5
13 cloxacilina dicloxacilina meticilina naficilina
14 oxacilina 5 eritromicina.
15 Con una
16 certeza de: 10%
17
18
19 CLIPS> (assert (organismo (identidad salmonela)) (paciente))
20 <Fact-2>
21 CLIPS> (run)
22 El tratammiento a seguir es: cloromicetina 5 ampicilina 5. Con
23 certeza de: 90%
```

Redes semánticas

Una red semántica es una forma de representación de conocimiento, en la cual los conceptos y sus interrelaciones se representan mediante un grafo dirigido o no dirigido. Los conceptos son representados mediante nodos, mientras que los arcos representan las relaciones semánticas entre los conceptos.



Redes semánticas

Las redes semánticas fueron desarrolladas por Robert F. Simmons, Sheldon Klein, Karen McConologue, M. Ross Quillian.

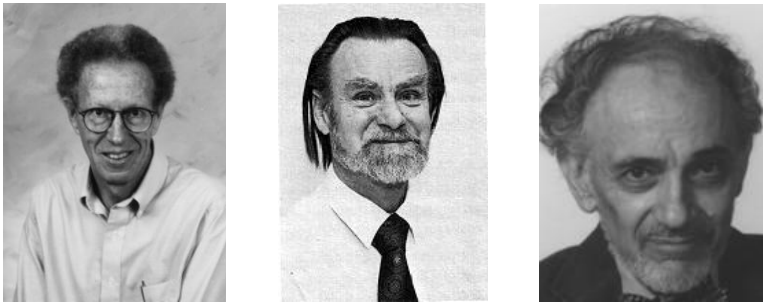
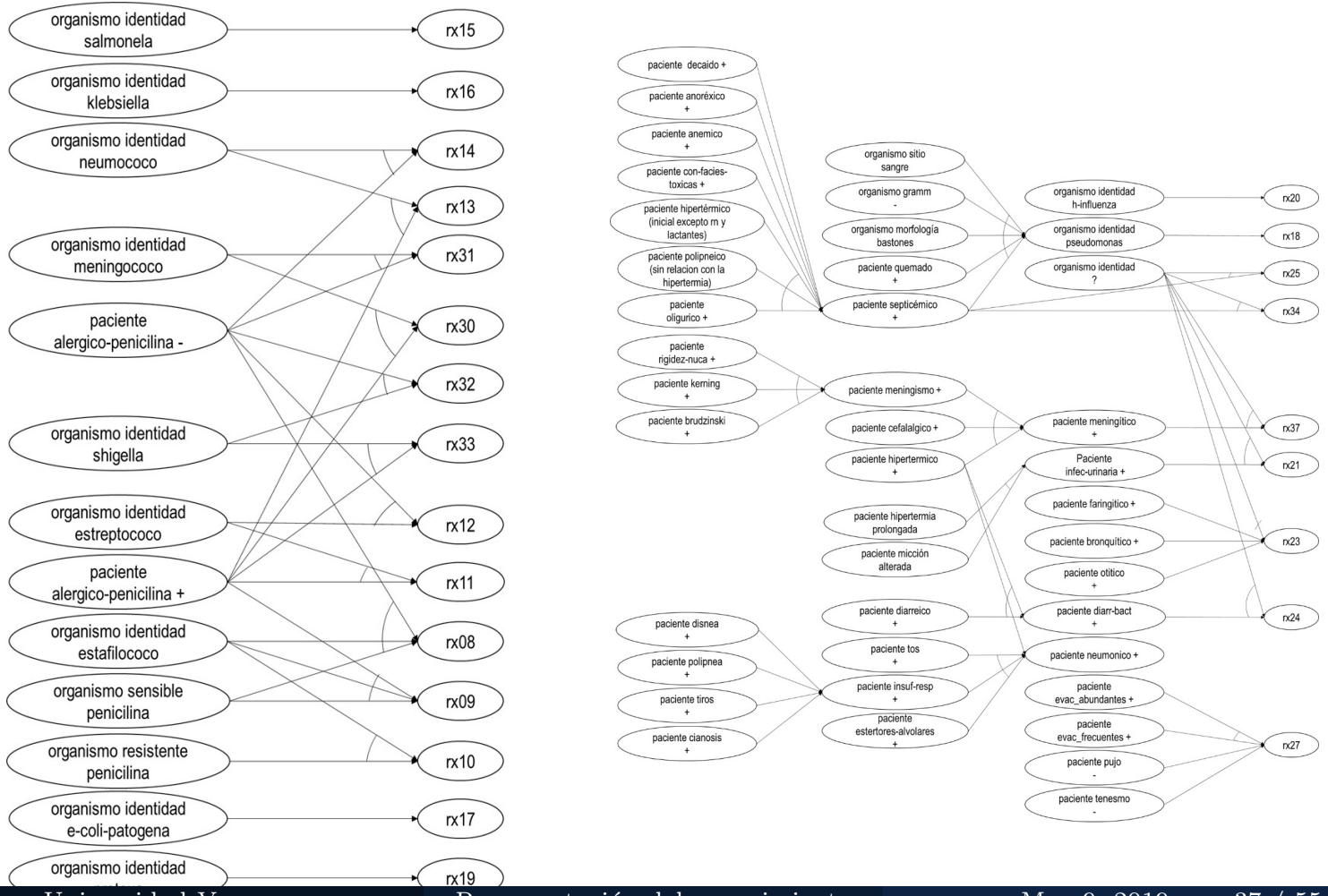


Figure 6: M. Ross Quillian, Robert F. Simmons y Sheldon Klein.

Las redes semánticas son utilizadas en el área del procesamiento del lenguaje natural, en aplicaciones como analizadores gramaticales semánticos, para recuperar información de gigantescas bases de datos en forma rápida y eficiente, así como en aplicaciones enfocadas al entendimiento textos.

Implementación en CLIPS



Implementación en CLIPS

Se realizó la implentación de una red semántica para la base de conocimiento *rc-kb.clp*. Para lo cual se utilizó el software *CLIPS*. Primordialmente se utilizó el constructor *deftemplate*.

```
1  (deftemplate idObjeto
2
3          (slot aributo1
4          (default valorDefault1))
5
6          (slot atributo2
7          (default valorDefault2))
8
9          (slot atributo3
10         (default valorDefault3)))
```

Plantilla *organismo*

```
1  (deftemplate organismo
2
3          (slot identidad
4          (default x))
5
6          (slot sensible
7          (default x))
8
9          (slot resistente
10         (default x))
11
12         (slot sitio
13         (default x))
14
15         (slot gramm
16         (default x))
17
18         (slot morfologia
19         (default x)))
```

Plantilla *paciente*

```
1  (deftemplate paciente
2
3          (slot alergico-penicilina
4          (default x))
5
6          (slot hipertermia
7          (default x))
8
9          (slot miccion
10         (default x))
11
12         (slot faringitico
13         (default x))
14
15         (slot bronquitico
16         (default x))
17
18         (slot otitico
19         (default x))
20                 .
21                 .
22                 .
```

Reglas

```
1
2  (defrule regla008
3
4     (or (organismo (identidad estafilococo))
5     (organismo (sensible penicilina))
6     (paciente (alergico-penicilina -)))
7
8     =>
9
10    (printout t "La␣superclase␣es␣el␣tratamiento:
11 ␣␣␣penicilina-g␣5" crlf))
```

Reglas

```
1  (defrule regla022
2
3     (or (paciente (hipertermia prolongada))
4     (paciente (miccion alterada)))
5
6     =>
7
8     (progn
9     (printout t "La␣superclase␣es
10 ␣␣␣(paciente␣(infec-urinaria␣+))" crlf)
11    (assert (paciente (infec-urinaria +)))))
```

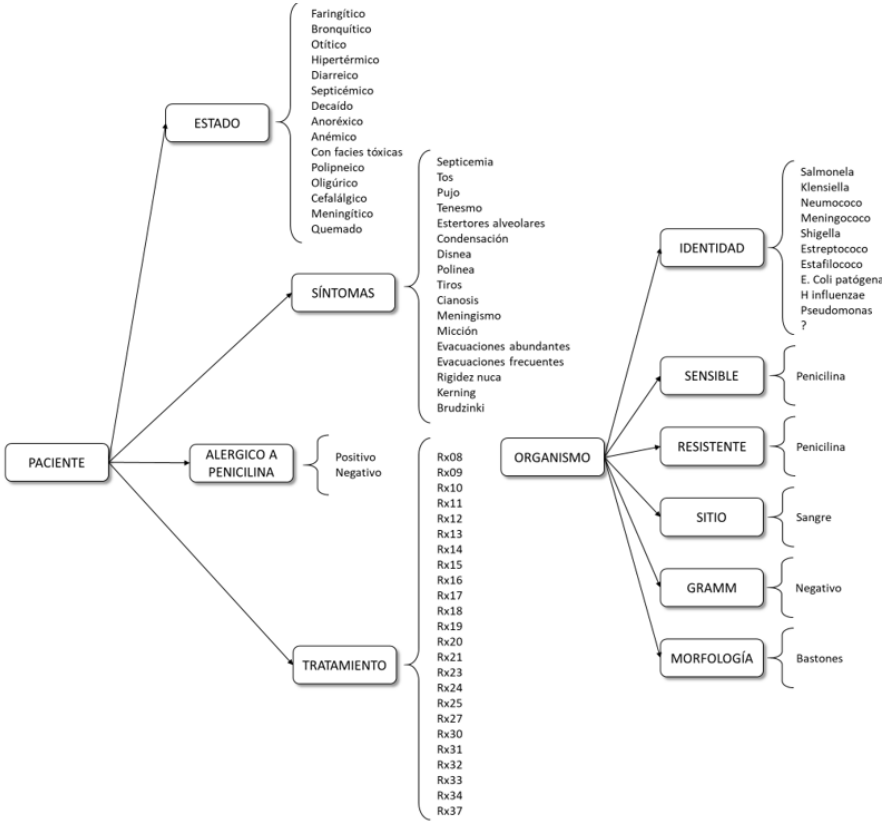
Frames-Macros

- Ténica de representación propuesta por Marvin Minsky.
- A Framework for Representing Knowledge.
- Antecedente de la programación orientada a objetos.
- Estructura de datos llamada marcos conformada por ranuras.
- Almacenan valores, referencias a otros marcos y procedimientos que regresan valores.

Propuesta

- Clases abstractas organismos y pacientes.
- Reglas que realizan pattern matching con ranuras de objetos.
- Regla que dispara inferencias.
- Definición de instancias.
- Message-handler.

Diagrama


PACIENTE
ESTADO
Faringítico
Bronquítico
Otítico
Hipertérmico
Diarreico
Septicémico
Decaído
Anoréxico
Anémico
Con facies tóxicas
Polipneico
Oligúrico
Cefalálgico
Meningítico
Quemado
SÍNTOMAS
Septicemia
Tos
Pujo
Tenesmo
Estertores alveolares
Condensación
Disnea
Polinea
Tiros
Cianosis
Meningismo
Micción
Evacuaciones abundantes
Evacuaciones frecuentes
Rigidez nuca
Kerning
Brudzinki
ALERGICO A PENICILINA
Positivo
Negativo
TRATAMIENTO
Rx08
Rx09
Rx10
Rx11
Rx12
Rx13
Rx14
Rx15
Rx16
Rx17
Rx18
Rx19
Rx20
Rx21
Rx23
Rx24
Rx25
Rx27
Rx30
Rx31
Rx32
Rx33
Rx34
Rx37
ORGANISMO
IDENTIDAD
Salmonela
Klensiella
Neumococo
Meningococo
Shigella
Estreptococo
Estafilococo
E. Coli patógena
H influenzae
Pseudomonas
?
SENSIBLE
Penicilina
RESISTENTE
Penicilina
SITIO
Sangre
GRAMM
Negativo
MORFOLOGÍA
Bastones

Clase Organismos

```
1  (defclass organismos (is-a USER)
2          (slot identidad (allowed-symbols desconocido
3          salmonela klebsiella meningococo shigella
4          estreptococo estafilococo e-coli-patogena
5          proteus h-influenzae pseudomonas neumococo))
6          (slot sensible
7          (allowed-symbols nil penicilina))
8          (slot resistente
9          (allowed-symbols nil penicilina))
10         (slot sitio
11         (allowed-symbols nil sangre))
12         (slot gramm
13         (allowed-symbols nil + -))
14         (slot morfologia
15         (allowed-symbols nil bastones)))
```

Clase Pacientes

```
1  (defclass pacientes (is-a USER)
2          (slot tos (allowed-symbols nil + -))
3          (slot pujo (allowed-symbols nil + -))
4          (slot tenesmo (allowed-symbols nil + -))
5          (slot condensacion (allowed-symbols nil + -))
6          (slot disnea (allowed-symbols nil + -))
7          .
8          .
9          .
10         (slot neumonico(allowed-symbols nil + -))
11         (slot insuf-resp (allowed-symbols nil + -))
12         (slot diarreico (allowed-symbols nil + -))
13         (slot diarr-bact (allowed-symbols nil + -))
14         (slot rigidez-nuca (allowed-symbols nil + -)))
```

Instancias Organismos

```
1  (definstances baseMedica
2          (org1 of organismos (identidad salmonela))
3          (org2 of organismos (identidad estafilococo)
4          (resistente penicilina))
5          (org3 of organismos (identidad estreptococo))
6          (org4 of organismos (identidad desconocido))
7          (org5 of organismos (identidad pseudomonas)
8          (sitio sangre) (gramm +) (morfologia bastones)))
```

Instancias Organismos

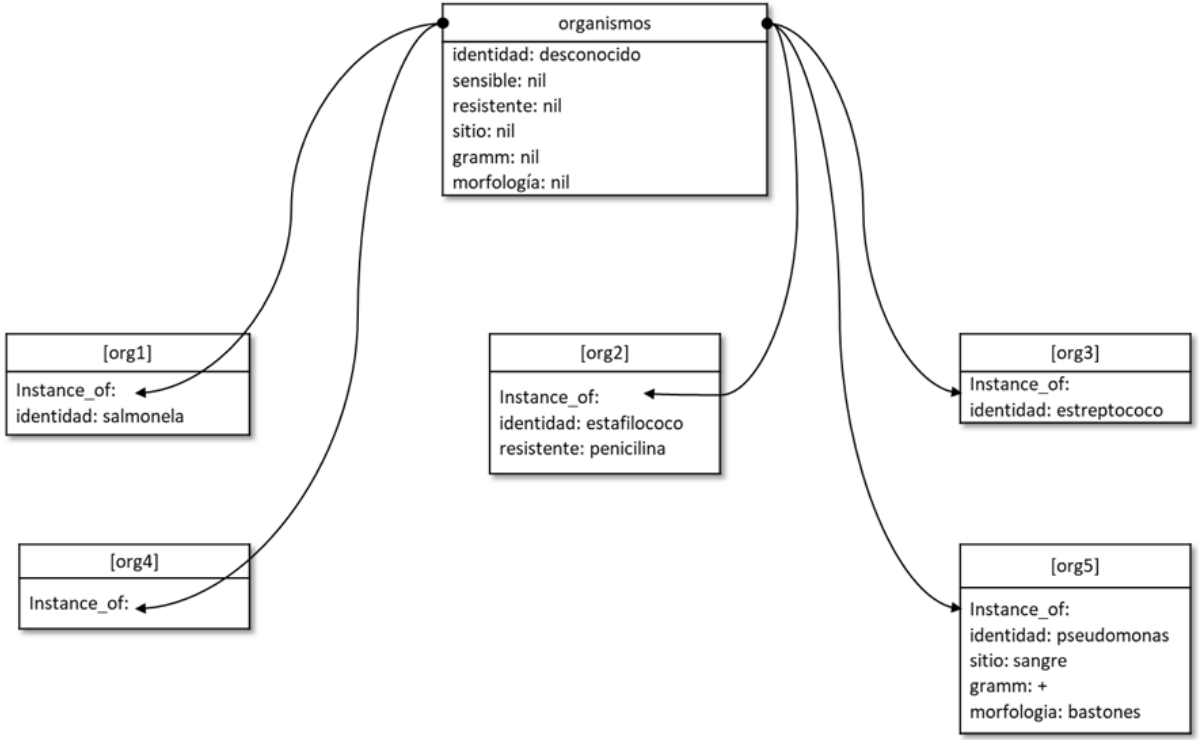


Figure 7: Clase organismos.

Instancias Pacientes

```
1 (definstances baseMedica
2         (pac1 of pacientes (alergico-penicilina +))
3         (pac2 of pacientes (hipertermia prolongada)
4         (miccion alterada))
5         (pac3 of pacientes (diarreico +) (hipertermico +))
6         (pac4 of pacientes (faringitico +) (septicemico +)
7         (quemado +))
8         (pac5 of pacientes (rigidez-nuca +) (kernig +)
9         (brudzinski +) (hipertermico +) (cefalalgico +)))
```

Instancias Pacientes

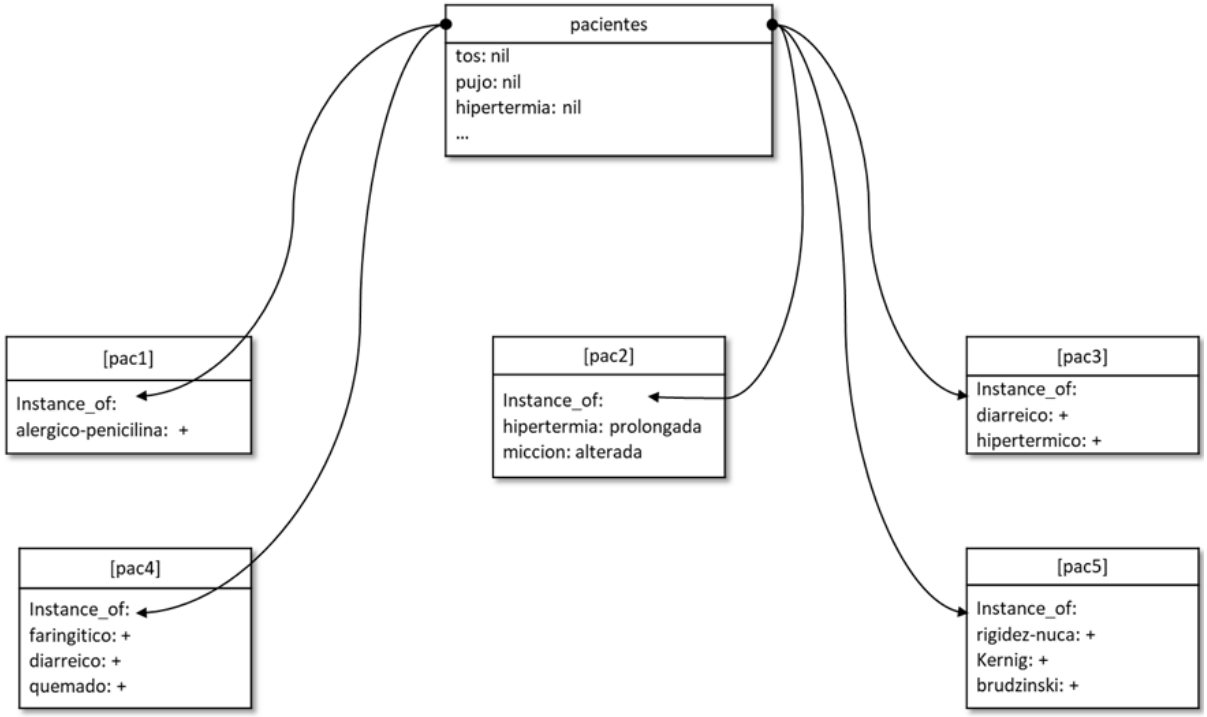


Figure 8: Clase pacientes.

Reglas de matching

```
1  (defrule regla008
2          (caso-organismo ?org)
3          (caso-paciente ?pac)
4          ?o1 <- (object (is-a organismos) (name ?org)
5          (identidad estafilococo) (sensible penicilina))
6          ?o2 <- (object (is-a pacientes) (name ?pac)
7          (alergico-penicilina -))
8  =>
9          (assert (paciente rx08 penicilina-g 5))
10         (printout t ''paciente rx08 penicilina-g 5'' crlf))
```

Reglas de matching

```
1  (defrule regla009
2          (caso-organismo ?org)
3          (caso-paciente ?pac)
4          ?o1 <- (object (is-a organismos) (name ?org)
5          (identidad estafilococo) (sensible penicilina))
6          ?o2 <- (object (is-a pacientes) (name ?pac)
7          (alergico-penicilina +))
8   =>
9          (assert (paciente rx09 eritromicina))
10         (printout t ''paciente alergico-penicilina +'' crlf))
```

Reglas de inicio

```
1  (defrule iniciar
2          (not (iniciado))
3  =>
4          (printout t ''Introduzca la instancia del
5          organismo a consulta'' crlf)
6          (bind ?org (read))
7          (printout t ''Introduzca la instancia del
8          paciente a consulta'' crlf)
9          (bind ?pac (read))
10         (assert (iniciado))
11         (assert (caso-organismo ?org))
12         (assert (caso-paciente ?pac)))
```

Message-handler

```
1 (defmessage-handler pacientes tratamiento ()
2 (reset)
3 (run))
```