

Técnicas de representación y razonamiento

❑ Tema 3: Representación del conocimiento e inferencia

❑ 3.4: Sistemas de producción – Índice de contenidos

- ❑ Arquitectura y funcionamiento de los sistemas de producción
 - ❑ Memoria de trabajo. Representación de hechos
 - ❑ Base de reglas. Representación de reglas
 - ❑ Motor de inferencia. Ciclo de funcionamiento
- ❑ Ventajas e inconvenientes de los sistemas de producción
- ❑ El proceso de razonamiento
 - ❑ Encadenamiento progresivo y encadenamiento regresivo
- ❑ Implementación
 - ❑ Fase de reconocimiento. Algoritmo RETE
 - ❑ Fase de selección. Estrategias de resolución de conflictos
 - ❑ Sistemas de producción en Prolog

Técnicas de representación y razonamiento

❑ Técnicas de representación del conocimiento

❑ Representaciones básicas

- ❑ Lógica de predicados. Representación en Prolog

❑ Sistemas de producción

- ❑ Redes semánticas

❑ Representaciones estructuradas

- ❑ Marcos (frames) y guiones (scripts)

❑ Estudio comparativo de las técnicas de representación

❑ Lenguajes de representación del conocimiento

Técnicas de representación y razonamiento

- ❑ Diversos formalismos para construir bases de conocimiento
 - ❑ Representaciones basadas en relaciones
 - ❑ Lógica
 - ❑ Redes semánticas
 - ❑ Representaciones basadas en objetos
 - ❑ Marcos
 - ❑ Objeto-Atributo-Valor
 - ❑ Representaciones basadas en acciones
 - ❑ Sistemas de producción
 - ❑ Guiones
 - ❑ Combinaciones y modificaciones de los anteriores

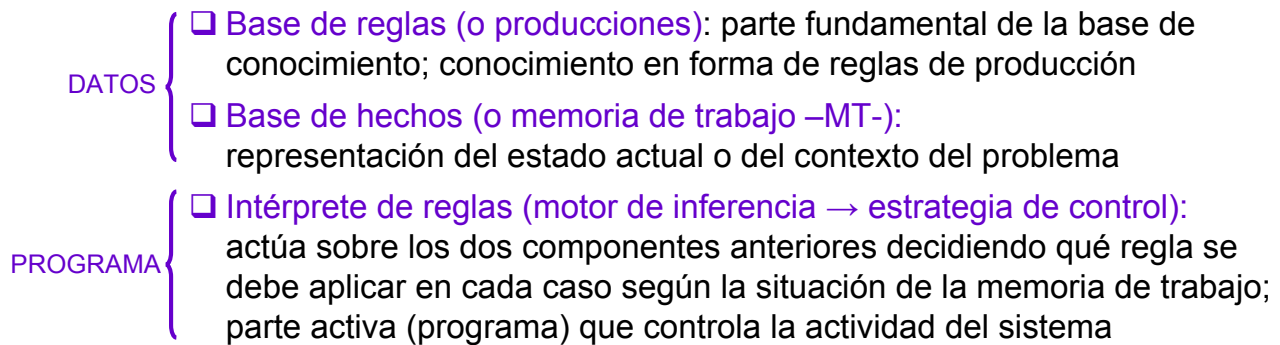
Producciones

- ❑ Técnica de representación “declarativa” utilizada para representar **conocimiento procedimental** (qué hacer en una situación determinada)
 - SI condiciones ENTONCES acciones**
 - ❑ No son implicaciones lógicas sino recomendaciones imperativas
 - ❑ Establecen patrones (situación-acción o premisas-conclusión) que indican cómo reaccionar ante una situación
 - ❑ Inspirada en teorías de comportamiento (estímulo → respuesta)
 - ❑ Las reglas representan el conjunto de habilidades de resolución de un problema de las que dispone un ser humano: modelo de su comportamiento
 - ❑ Muy adecuadas para representar conocimiento heurístico
 - ❑ Sistemas de producción o sistemas basados en reglas: esquema de representación de conocimiento procedimental más popular
 - ❑ El conocimiento sobre las acciones es más importante que el conocimiento estático del dominio

Sistemas de producción

❑ ARQUITECTURA

❑ Los sistemas de producción constan de tres partes:



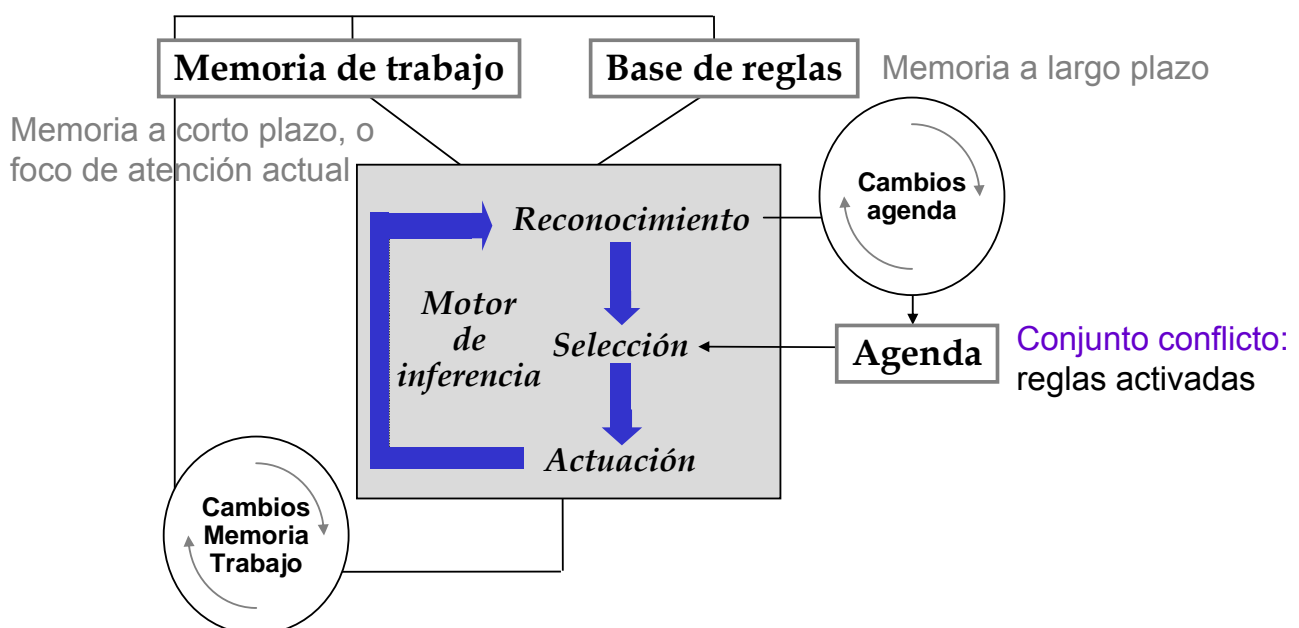
❑ FUNCIONAMIENTO EN CICLOS DE OPERACIÓN

❑ En cada ciclo hay tres fases:

- ❑ **Reconocimiento o equiparación (*matching*):** qué reglas son aplicables (**activación**; **conjunto conflicto**; **agenda**)
- ❑ **Selección o resolución de conflictos:** qué regla se “dispara”
- ❑ **Actuación:** se realiza la acción correspondiente (cambio en la MT)

Arquitectura y funcionamiento

❑ Funcionamiento de un sistema de producción



La **estrategia de control del motor de inferencia** especifica la forma de resolver conflictos si hay varias reglas aplicables

Memoria de trabajo o base de hechos

- ❑ MEMORIA DE TRABAJO (base de hechos)
 - ❑ Contiene conocimiento declarativo
 - ❑ Representa el estado actual o contexto de la tarea o problema
 - ❑ Estado inicial: situación inicial u origen del problema
 - ❑ Estados finales: situación objetivo (puede haber más de una)
 - ❑ Estados intermedios: situación actual o en curso de resolución
 - ❑ Parte de la MT es permanente, otra es temporal y se modificará conforme se vayan aplicando los procesos de inferencia
 - ❑ La ejecución de reglas modifica la MT añadiendo, modificando o eliminando hechos en ella
 - ❑ Representación de hechos: *Pepe nació en Zaragoza en 1966*
 - ❑ CLIPS:
(Persona (Nombre Pepe) (FechaNacimiento 1966)
(LugarNacimiento Zaragoza))
 - ❑ Prolog: `persona(pepe, 1966, zaragoza).`

Base de reglas

- ❑ BASE DE REGLAS
 - ❑ Representa conocimiento procedimental sobre cómo solucionar problemas en un dominio
 - ❑ Representación uniforme del tipo:
SI Condiciones ENTONCES Acciones
 - ❑ Condiciones: lista de cosas a verificar en la MT
 - ❑ Acciones: conjunto de acciones a realizar sobre la MT
 - ❑ No son reglas las estructuras del tipo *if-then-else*
 - ❑ Las reglas son independientes unas de otras
 - ❑ Una regla no puede hacer referencia a otra
 - ❑ La única comunicación entre reglas es a través de la MT del sistema, que es visible para todas ellas
 - ❑ Así se facilita añadir, modificar o eliminar reglas de la BR
 - ❑ Flexibilidad/modularidad versus ineficiencia
 - ❑ Cada regla modifica poco la MT
 - ❑ Existen técnicas de *chunking* para agrupar reglas

Representación de reglas

❑ Ejemplo de regla en “pseudo-código”:

- ❑ Si la edad del paciente es inferior a 10 años, tiene manchas rojas y fiebre entonces tiene varicela

**SI (Paciente \$p \$edad) and (\$edad < 10) and
(Síntomas \$p fiebre) and (Síntomas \$p manchas-rojas)**

ENTONCES (Enfermedad \$p varicela)

❑ Constantes:

- ❑ Paciente,
- ❑ Síntomas,
- ❑ Enfermedad,
- ❑ 10,
- ❑ fiebre,
- ❑ manchas-rojas,
- ❑ varicela

❑ Variables:

- ❑ \$p,
- ❑ \$edad

Motor de inferencia

❑ MOTOR DE INFERENCIA: colección integrada de algoritmos de resolución de problemas (algoritmos de búsqueda)

- ❑ Se ocupa de determinar qué reglas son aplicables, seleccionar una y aplicarla, hasta alcanzar el objetivo
- ❑ Está ya codificado y probado, puede utilizarse con diferentes BRs
- ❑ Permite cambiar la estrategia de control (selección de reglas)

❑ El conocimiento (reglas y hechos) es independiente del programa que lo usa (motor de inferencia)

- ❑ Representación declarativa de conocimiento procedimental
- ❑ Flexibilidad, modularidad
- ❑ Una representación procedimental sería más eficiente, pero perdería la independencia, la modularidad (control/conocimiento)
 - ❑ Por ejemplo, una secuencia de *ifs* que dejen claro el orden de aplicación de las reglas
 - ❑ Flujo de control determinado: rigidez, incapacidad de adaptación

Ejemplo sencillo: ordenación de cadenas

- SP que ordena las letras de una cadena de $\{a, b, c\}^*$

- Conjunto de producciones

Ejemplo de traza

1) $ba \rightarrow ab$

2) $ca \rightarrow ac$

3) $cb \rightarrow bc$

iteración	MT	C.conflicto	R.disparada
0	cbaca	1, 2, 3	1
1	cabca	2	2
2	acbca	2, 3	2
3	acbac	1, 3	1
4	acabc	2	2
5	aacbc	3	3
6	aabcc	\emptyset	Halt

- Los sistemas de producción son un modelo de cómputo general (misma potencia de cómputo que las máquinas de Turing)

Ventajas

- Ventajas de los sistemas de producción
 - Separación de conocimiento y control
 - Simplifica el desarrollo de sistemas expertos
 - Carácter totalmente modular de la base de conocimiento
 - Cada regla es una unidad de conocimiento que puede ser añadida, modificada o eliminada independientemente del resto de las reglas
 - Independencia de las reglas: no hay interacciones entre ellas
 - La comunicación se establece a través de la memoria de trabajo
 - Uniformidad
 - Conocimiento representado con una sintaxis muy simple y homogénea
 - Naturalidad para representar cierto tipo de conocimiento
 - Control dirigido por patrones
 - Los programas en IA necesitan flexibilidad
 - Las reglas pueden “dispararse” en cualquier secuencia
 - Estado \rightarrow reglas aplicables \rightarrow camino de la solución

Ventajas de los sistemas de producción

- ❑ Independencia del lenguaje
 - ❑ El modelo es independiente de la representación elegida para reglas y hechos, siempre que el lenguaje soporte encaje de patrones
- ❑ Correspondencia natural con búsqueda en espacios de estados
 - ❑ Nodos: estados sucesivos de la MT, operadores: reglas
- ❑ Modelo plausible del mecanismo humano de resolución de problemas
 - ❑ Razonamiento basado en reglas: decidir qué hacer o qué se puede deducir cuando se dan una serie de condiciones
 - ❑ Uno de los primeros usos de los sistemas de producción (*Newell & Simon*). Siguen usándose para estudiar el comportamiento humano
- ❑ Trazabilidad del proceso de razonamiento
 - ❑ Se pueden comunicar qué condiciones se conocían en un momento determinado, por qué se ha escogido una regla y qué se ha deducido tras aplicarla
 - ❑ Ayuda a la depuración

Inconvenientes

- ❑ Inconvenientes de los sistemas de producción
 - ❑ Ineficiencia
 - ❑ El proceso de reconocimiento de patrones es muy ineficiente
 - ❑ El precio a pagar por la modularidad, flexibilidad y uniformidad
 - ❑ Dificultad de verificación de la consistencia de la BC
 - ❑ ¿Conocimiento contradictorio al modificar la BC?
 - ❑ Opacidad y dificultad de depuración
 - ❑ Es difícil examinar una BC y determinar qué acciones van a ocurrir (depende del motor de inferencia)
 - ❑ No hay un flujo de control claro
 - ❑ La división del conocimiento en reglas hace que cada regla individual sea fácilmente tratable, pero se pierde la visión global
 - ❑ Las reglas representan pasos muy pequeños en la resolución del problema sin que haya una jerarquía de reglas
 - ❑ Dificultad en cubrir todo el conocimiento
 - ❑ Cuello de botella de la adquisición de conocimiento

Dominios apropiados

- ☐ Dominios apropiados para los sistemas de producción
 - ☐ Aquellos en los que la tarea que se intenta resolver puede verse como un conjunto de transiciones de un estado a otro
 - ☐ Dominios con conocimiento difuso, con muchos hechos y con conocimiento incompleto o no muy bien definido
 - ☐ Como en medicina clínica
 - ☐ Aquellos en los que los procesos pueden representarse como un conjunto de acciones independientes
 - ☐ Dominios que separan el conocimiento de la forma de uso
 - ☐ Razonamiento basado en reglas: especialmente indicado para representar conocimiento experto en dominios que necesitan heurísticas para lidiar con información compleja y/o incompleta
 - ☐ Eficiencia en dominios no demasiado amplios

El proceso de razonamiento

- ☐ Es una progresión desde un conjunto de datos hacia una solución, respuesta o conclusión
- ☐ Hay dos alternativas:
 - ☐ Pocos datos iniciales y/o muchas posibles conclusiones → lo razonable: progresar desde los datos iniciales hasta una solución
 - ☐ Razonamiento dirigido por los datos (antecedentes)
 - ☐ Encadenamiento de reglas hacia adelante: **encadenamiento progresivo**
 - ☐ Muchos datos iniciales, pero sólo unos pocos son relevantes
 - ☐ Razonamiento dirigido por los objetivos (consecuentes)
 - ☐ Encadenamiento de reglas hacia atrás: **encadenamiento regresivo**
- ☐ Sistemas híbridos: encadenamiento hacia delante y hacia atrás

Encadenamiento progresivo y regresivo

- ❑ Encadenamiento hacia delante o progresivo
 - ❑ Es un tipo de razonamiento dirigido por los datos (antecedentes)
 - ❑ Comienza con todos los datos conocidos y progresa hasta la conclusión
 - ❑ Si los datos conocidos verifican las condiciones de una regla, entonces la regla se puede aplicar
- ❑ Encadenamiento hacia atrás o regresivo
 - ❑ Es un tipo de razonamiento dirigido por los objetivos (consecuentes)
 - ❑ Selecciona una conclusión posible e intenta probar su validez buscando evidencias que la soporten
 - a) Un antecedente es cierto si está en la MT del sistema
 - b) Si el antecedente no está en la MT, se busca si es consecuente de alguna regla y se prueban recursivamente los antecedentes de dicha regla
 - c) Si no son aplicables a) y b), puede asumirse la **hipótesis del mundo cerrado**

Encadenamiento progresivo y regresivo

- ❑ Las producciones pueden utilizarse de dos formas:
 - $(P1) \$ \rightarrow a\a
 - $(P2) \$ \rightarrow b\b
 - $(P3) \$ \rightarrow c\c

\$ encaja con cualquier cadena
- ❑ **Encadenamiento hacia adelante:** utilizar las reglas para generar palíndromos
 - ❑ Dado un símbolo inicial como c , la secuencia de reglas $(P1)$, $(P2)$, $(P3)$, $(P2)$, $(P3)$ generará la siguiente secuencia de cadenas:
 $aca \quad bacab \quad cbacabc \quad bcbacabcb \quad cbcbacabcb$
- ❑ **Encadenamiento hacia atrás:** utilizar las reglas para reconocer palíndromos
 - ❑ Dado un palíndromo como $bacab$, podemos trazar la secuencia de reglas que llevan a su construcción
 - ❑ $bacab$ satisface la parte derecha de $(P2)$, cuya parte izquierda da aca
 - ❑ aca satisface la parte derecha de $(P1)$, cuya parte izquierda da c

Ejercicio de encadenamiento de reglas

- ❑ Dado un sistema basado en reglas con la siguiente base de conocimiento

- ❑ R1: **Si** h_2 y h_5 **entonces** h_1

- ❑ R2: **Si** h_4 y h_3 **entonces** h_2

- ❑ R3: **Si** h_6 **entonces** h_3

donde cada h_i representa una situación o concepto y los números al lado de las reglas marcan la prioridad de ejecución de las mismas en caso de conflicto

- ❑ La base de hechos inicial contiene los siguientes datos:

h_6, h_7, h_9, h_8, h_4 y h_5

- a) Aplica encadenamiento hacia delante, mostrando cómo evoluciona el sistema en cada ciclo del proceso
 - b) Aplicando encadenamiento hacia atrás, determina si es posible establecer la existencia de la situación h_1 a partir de la base de hechos inicial

a) Solución ejercicio: encadenamiento hacia adelante

- ❑ Seleccionar aquellas reglas cuyo antecedente se cumple a partir de la base de hechos actual. Si hay más de una regla seleccionada, emplear una estrategia de resolución de conflictos que elimine todas las reglas anteriores menos una.
- ❑ Ejecutar la regla resultante del paso anterior

Explicación

Inicialmente se tendría BH₀, de las tres reglas existentes en la base de conocimientos sólo la 3 se selecciona.

Tras su ejecución, se obtiene BH₁. Ahora existirá un conflicto entre las reglas 2 y 3. 2 es más prioritaria y 3 ya se ha aplicado. Se ejecuta 2 obteniendo BH₂.

En esta situación todas las reglas forman parte de la agenda. Finalmente, es la regla 1 la que se ejecuta, quedando BH₃.

BH ₀	BH ₁	BH ₂	BH ₃
h_6	h_6	h_6	h_6
h_7	h_7	h_7	h_7
h_9	h_9	h_9	h_9
h_8	h_8	h_8	h_8
h_4	h_4	h_4	h_4
h_5	h_5	h_5	h_5
	h_3	h_3	h_3
		h_2	h_2
			h_1

b) Solución ejercicio: encadenamiento hacia atrás

- ☐ Se parte de un concepto objetivo a verificar a partir de la BC y la BH.
- ☐ Comprobar si dicho concepto pertenecía ya o no a la BH. En caso negativo recurrir a las reglas en la BC en cuyo consecuente figure el concepto objetivo. Así los conceptos del antecedente de dichas reglas pasan a ser subobjetivos. Si se demuestra la validez de estos subobjetivos, podrá inferirse el concepto objetivo global.
- ☐ En el ejemplo el objetivo global es *h1*, que no figura en la BH inicial;
- ☐ Ir a la BC. La regla 1 tiene *h1* en su consecuente. Ello nos permite fijar *h2* y *h5* como subobjetivos. Al encontrarse *h5* en la BH inicial, sólo quedaría por demostrar *h2*.
- ☐ La regla 2, con *h2* en su consecuente, establece *h4* y *h3* como nuevos subobjetivos. El hecho *h4* está en la BH inicial. Habrá que verificar, la existencia de *h3*.
- ☐ La regla 3 fija como subobjetivo *h6*, que está en la BH inicial, por tanto **final**.
- ☐ Entonces *h1* es válido aplicando las reglas 3, 2 y 1 a la BH inicial.

Elección de la dirección de búsqueda

- ☐ Depende por completo del problema
 - ☐ Complejidad de las reglas
 - ☐ Aspecto del espacio de estados
 - ☐ Naturaleza y disponibilidad de los datos del problema
 - ☐ Factor de ramificación en ambas direcciones
- ☐ Empezar por el menor número de hechos (inicial u objetivo)
- ☐ Si el proceso de resolución es interactivo, el encadenamiento tiene que ser el mismo que utilice el usuario de forma natural para resolver el problema
- ☐ En los sistemas de contestación de preguntas lo lógico es trabajar hacia atrás (*p.ej. sistemas de diagnóstico*)
- ☐ Los sistemas que tienen que reaccionar ante datos necesitan trabajar hacia delante
 - ☐ Por ejemplo, extraer conclusiones acerca de muchos datos recolectados, que pueden irse incorporando en ejecución

Ejemplo: reglas de identificación de frutas

- R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Intérprete con encadenamiento progresivo

❑ Pasos del intérprete:

- 1) **Reconocimiento, equiparación o encaje:** encuentra reglas aplicables y las activa
- 2) **Resolución de conflictos:** desactiva reglas que no añadan hechos nuevos, y selecciona la **primera regla aplicable** (la de menor número en este ejemplo)
- 3) **Actuación:** ejecuta la acción de la regla (la dispara). Si no hay, se detiene
- 4) **Reset:** vacía la agenda (desactiva reglas aplicables) y vuelve al paso 1)

❑ Si la memoria de trabajo tiene los siguientes hechos iniciales:

Diámetro = 0.4 cm, Forma = redonda, Numsemillas = 1, Color = rojo

Ciclo de ejecución	Reglas aplicables	Regla seleccionada	Hecho derivado
--------------------	-------------------	--------------------	----------------

1

❑ Ésta es sólo una de las posibles estrategias de selección (control)

Ejemplo: reglas de identificación de frutas

- R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
- R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
- R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
- R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
- R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
- R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
- R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
- R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
- R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
- R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
- R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Diámetro = 0.4 cm,
Forma = redonda,
Numsemillas = 1,
Color = rojo

Intérprete con encadenamiento progresivo

❑ Pasos del intérprete:

- 1) Reconocimiento, equiparación o encaje:** encuentra reglas aplicables y las activa
- 2) Resolución de conflictos:** desactiva reglas que no añadan hechos nuevos, y selecciona la **primera regla aplicable** (la de menor número en este ej.)
- 3) Actuación:** ejecuta la acción de la regla (la dispara). Si no hay, se detiene
- 4) Reset:** vacía la agenda (desactiva reglas aplicables) y vuelve al paso **1)**

❑ Si la memoria de trabajo tiene los siguientes hechos iniciales:

Diámetro = 0.4 cm, Forma = redonda, Numsemillas = 1, Color = rojo

Ciclo de ejecución	Reglas aplicables	Regla seleccionada	Hecho derivado
1	3, 4	3	claseFruta = árbol
2			

Ejemplo: reglas de identificación de frutas

R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana

R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta

R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol

R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso

R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple

R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía

R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón

R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque

R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja

R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza

R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón

R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana

R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Diámetro = 0.4 cm,
Forma = redonda,
Numsemillas = 1,
Color = rojo,
claseFruta = árbol

Intérprete con encadenamiento progresivo

❑ Pasos del intérprete:

- 1) Reconocimiento, equiparación o encaje:** encuentra reglas aplicables y las activa
- 2) Resolución de conflictos:** desactiva reglas que no añadan hechos nuevos, y selecciona la **primera regla aplicable** (la de menor número en este ej.)
- 3) Actuación:** ejecuta la acción de la regla (la dispara). Si no hay, se detiene
- 4) Reset:** vacía la agenda (desactiva reglas aplicables) y vuelve al paso 1)

❑ Si la memoria de trabajo tiene los siguientes hechos iniciales:

Diámetro = 0.4 cm, Forma = redonda, Numsemillas = 1, Color = rojo

Ciclo de ejecución	Reglas aplicables	Regla seleccionada	Hecho derivado
1	3, 4	3	claseFruta = árbol
2	3, 4	4	claseSemilla = hueso
3			

Desactivada por el principio de refracción

Ejemplo: reglas de identificación de frutas

- R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
- R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
- R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
- R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
- R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
- R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
- R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
- R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
- R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
- R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
- R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

...
Color = rojo,
claseFruta = árbol,
claseSemilla = hueso

Intérprete con encadenamiento progresivo

❑ Pasos del intérprete:

- 1) **Reconocimiento, equiparación o encaje:** encuentra reglas aplicables y las activa
- 2) **Resolución de conflictos:** desactiva reglas que no añadan hechos nuevos, y selecciona la **primera regla aplicable** (la de menor número en este ej.)
- 3) **Actuación:** ejecuta la acción de la regla (la dispara). Si no hay, se detiene
- 4) **Reset:** vacía la agenda (desactiva reglas aplicables) y vuelve al paso 1)

❑ Si la memoria de trabajo tiene los siguientes hechos iniciales:

Diámetro = 0.4 cm, Forma = redonda, Numsemillas = 1, Color = rojo

Ciclo de ejecución	Reglas aplicables	Regla seleccionada	Hecho derivado
1	3, 4	3	claseFruta = árbol
2	3, 4	4	claseSemilla = hueso
3	3, 4, 10	10	Fruta = cereza
4	3, 4, 10	Ya no hay más reglas en la agenda – FIN	

Conclusión: la fruta es una cereza

Intérprete con encadenamiento regresivo

❑ Pasos de un intérprete con encadenamiento regresivo



- 1) Formar una pila con todos los objetivos iniciales
- 2) Reunir todas las reglas capaces de satisfacer el 1º objetivo
- 3) Para cada una de estas reglas, **examinar sus premisas**:

orden ↗

- a) Si las premisas son satisfechas, ejecutar la regla para derivar sus conclusiones. Eliminar el objetivo de la pila y volver al paso 2)
 - b) Si una de las premisas no se cumple, buscar las reglas que pueden derivar esta premisa. Si se encuentra alguna, se considera la premisa como subobjetivo, se coloca al principio de la pila, y se va al paso 2)
 - c) Si el paso b) no puede encontrar una regla, entonces preguntar al usuario por el valor del parámetro, y añadir éste a la memoria de trabajo. Si este valor satisface la premisa en curso, continuar con la siguiente premisa de esta regla. Si no, continuar con la siguiente regla
- 4) Si todas las reglas que pueden satisfacer el objetivo actual se han probado y han fallado, entonces el objetivo en curso permanece indeterminado. Sacar éste de la pila y volver al paso 2). Si la pila de objetivos está vacía, el intérprete se detiene

Ejemplo de encadenamiento regresivo

❑ Traza de ejecución para derivar cereza como valor de fruta

- 1) Objetivos: (Fruta) Base de hechos inicial: vacía
- 2) Reglas que pueden satisfacer el objetivo: 1, 6 - 13
- 3) Examinamos premisas:
 - ❑ Regla 1: La 1ª premisa (Forma= alargada) no se encuentra en la memoria de trabajo. No hay reglas que deriven este valor → c)
 - ❑ ¿Cuál es el valor de Forma? redonda
 - Memoria de trabajo: ((Forma redonda)) 
 - La regla 1 falla
 - ❑ Regla 6: La 1ª premisa (claseFruta = planta) no se encuentra en la MT. Las reglas 2 y 3 pueden derivar el valor claseFruta → b)
 - Objetivos: (claseFruta, Fruta)
- 2) Reglas que pueden satisfacer el objetivo: 2, 3 
- 3) Examinamos premisas

Objetivo = Fruta

✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana

R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta

R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol

R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso

Objetivos: (Fruta)

R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple

Hechos: Forma = redonda

R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía



R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón

R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque

R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja

R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza

R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón

R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana

R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Objetivo = claseFruta

✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana

R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta

R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol

R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso

Objetivos: (claseFruta, Fruta)

R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple

Hechos: Forma = redonda

R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía

R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón

R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque

R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja

R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza

R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón

R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana

R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Ejemplo de encadenamiento regresivo

- ❑ Regla 2: La 1ª premisa (Forma = redonda u ovalada) es satisfecha puesto que el valor de Forma es redonda. Se continúa con la siguiente premisa (Diámetro > 1.6 cm). Puesto que no existe un valor de Diámetro ni se puede derivar de otras reglas → c)

❑ ¿Cuál es el valor de Diámetro? 0.4

Memoria de trabajo: ((Forma redonda) (Diámetro 0.4))

La regla 2 falla

- ❑ Regla 3: Las premisas se cumplen: (Forma = redonda), (Diámetro < 1.6 cm) → a) Se deriva (claseFruta = árbol)

Memoria de trabajo: ((Forma redonda) (Diámetro 0.4) (claseFruta árbol))

Objetivos: (Fruta)

3) Volvemos a la regla 6

- ❑ Regla 6: La 1ª premisa (claseFruta = planta) no se encuentra en la MT ni puede derivarse pues ya sabemos que no se cumple. Falla
- ❑ Regla 7: Falla por el mismo motivo

Objetivo = Fruta

- ✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- ✗ R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
- R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
- R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
- ✗ R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
- ✗ R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
- R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
- R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
- R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
- R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
- R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
- R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Objetivos: (Fruta)
Hechos: Forma = redonda,
Diámetro = 0.4
claseFruta = árbol

Ejemplo de encadenamiento regresivo

- ❑ Regla 8: La 1ª premisa (*claseFruta = árbol*) es satisfecha. Se continúa con la siguiente premisa (*Color = naranja*). Puesto que no existe un valor de color ni se puede derivar de otras reglas → c)

❑ ¿Cuál es el valor de Color? rojo

Memoria de trabajo:

((Forma redonda) (Diámetro 0.4) (*claseFruta árbol*) (*Color rojo*))

Falla la regla 8

- ❑ Regla 9: Falla por el mismo motivo
 - ❑ Regla 10: Se cumplen las dos primeras premisas (*claseFruta = árbol*), (*Color = rojo*). Se continúa con la siguiente premisa (*claseSemilla = hueso*). Puesto que no existe un valor para *claseSemilla* pero se puede derivar de las reglas 4 y 5 → b)
- Objetivos: (*claseSemilla, Fruta*)

2) Reglas que pueden satisfacer el objetivo: 4, 5

3) Examinamos premisas

Objetivo = claseSemilla

- ✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- ✗ R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES *claseFruta = planta*
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES *claseFruta = árbol*
- R4: SI numSemillas = 1 ENTONCES *claseSemilla = hueso*
- R5: SI numSemillas > 1 ENTONCES *claseSemilla = múltiple*
- ✗ R6: SI *claseFruta = planta* Y Color = verde ENTONCES Fruta = sandía
- ✗ R7: SI *claseFruta = planta* Y Color = amarillo ENTONCES Fruta = melón
- ✗ R8: SI *claseFruta = árbol* Y Color = naranja Y *claseSemilla = hueso* ENTONCES Fruta = albaricoque
- ✗ R9: SI *claseFruta = árbol* Y Color = naranja Y *claseSemilla = múltiple* ENTONCES Fruta = naranja
- R10: SI *claseFruta = árbol* Y Color = rojo Y *claseSemilla = hueso* ENTONCES Fruta = cereza
- R11: SI *claseFruta = árbol* Y Color = naranja Y *claseSemilla = hueso* ENTONCES Fruta = melocotón
- R12: SI *claseFruta = árbol* Y Color = rojo o amarillo o verde Y *claseSemilla = múltiple* ENTONCES Fruta = manzana
- R13: SI *claseFruta = árbol* Y Color = morado Y *claseSemilla = hueso* ENTONCES Fruta = ciruela

Objetivos: (*claseSemilla, Fruta*)
Hechos: Forma = redonda,
Diámetro = 0.4
claseFruta = árbol
Color = rojo

Ejemplo de encadenamiento regresivo

- ❑ Regla 4: La premisa (numSemillas = 1) no se encuentra en la memoria de trabajo. No hay reglas que deriven este valor → c)

❑ ¿Cuál es el valor de NumSemilla? 1

Memoria de trabajo:

((Forma redonda) (Diámetro 0.4) (claseFruta árbol) (Color rojo) (numSemilla 1))

La premisa se cumple → a) Se deriva (claseSemilla = hueso)

Memoria de trabajo:

((Forma redonda) (Diámetro 0.4) (claseFruta árbol) (Color rojo) (numSemilla 1)
(claseSemilla = hueso))

Objetivos: (Fruta)



3) Volvemos a la regla 10

- ❑ Regla 10: La premisa se cumple → a) Se deriva (Fruta = cereza)

Memoria de trabajo:

((Forma redonda) (Diámetro 0.4) (claseFruta árbol) (Color rojo) (numSemilla 1)
(claseSemilla hueso) (Fruta cereza))

Objetivos: ()



4) El intérprete se detiene (pila de objetivos vacía)

Objetivo = Fruta

- ✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- ✗ R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
- R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
- R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
- ✗ R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
- ✗ R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
- ✗ R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
- ✗ R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
- R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
- R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
- R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
- R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela

Objetivos: (Fruta)

Hechos:

numSemilla = 1

claseSemilla = hueso



No quedan objetivos: fin

- ✗ R1: SI Forma = alargada Y Color = verde o amarillo ENTONCES Fruta = banana
- ✗ R2: SI Forma = redonda u ovalada Y Diámetro > 1.6 cm ENTONCES claseFruta = planta
- R3: SI Forma = redonda Y Diámetro < 1.6 cm ENTONCES claseFruta = árbol
- R4: SI numSemillas = 1 ENTONCES claseSemilla = hueso
- R5: SI numSemillas > 1 ENTONCES claseSemilla = múltiple
- ✗ R6: SI claseFruta = planta Y Color = verde ENTONCES Fruta = sandía
- ✗ R7: SI claseFruta = planta Y Color = amarillo ENTONCES Fruta = melón
- ✗ R8: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = albaricoque
- ✗ R9: SI claseFruta = árbol Y Color = naranja Y claseSemilla = múltiple ENTONCES Fruta = naranja
- R10: SI claseFruta = árbol Y Color = rojo Y claseSemilla = hueso ENTONCES Fruta = cereza
- R11: SI claseFruta = árbol Y Color = naranja Y claseSemilla = hueso ENTONCES Fruta = melocotón
- R12: SI claseFruta = árbol Y Color = rojo o amarillo o verde Y claseSemilla = múltiple ENTONCES Fruta = manzana
- R13: SI claseFruta = árbol Y Color = morado Y claseSemilla = hueso ENTONCES Fruta = ciruela
- Objetivos: ()
- Hechos:
- claseFruta = árbol
- Color = rojo
- claseSemilla = hueso
- Fruta = cereza

Razonamiento versus encadenamiento

- ❑ Distinción entre razonamiento y encadenamiento:
 - ❑ Razonamiento dirigido por los datos: de los hechos a los objetivos
 - ❑ El encadenamiento progresivo de reglas es una implementación natural del razonamiento dirigido por los datos
 - ❑ Razonamiento dirigido por los objetivos: de los objetivos a los hechos
 - ❑ El encadenamiento regresivo de reglas es una implementación natural del razonamiento dirigido por los objetivos
 - ❑ Modo de razonamiento adecuado: depende del tipo de problema
 - ❑ Modo de encadenamiento: depende del entorno de desarrollo
 - ❑ Aunque es posible prescindir del encadenamiento proporcionado por el entorno y programar directamente un mecanismo alternativo
 - ❑ CLIPS: mecanismo de encadenamiento progresivo (hacia adelante)
 - ❑ Prolog: mecanismo de encadenamiento regresivo (hacia atrás)
 - ❑ En CLIPS se puede hacer razonamiento regresivo y en Prolog se puede hacer razonamiento progresivo si se hace un control explícito del encadenamiento

Implementación: fase de reconocimiento

- ❑ Encaje (*matching*) por ajuste o encaje de patrones
 - ❑ Qué reglas son aplicables en un cierto estado
- ❑ La situación es distinta dependiendo de si el enfoque es con encadenamiento hacia delante o hacia atrás
 - ❑ Encadenamiento hacia atrás → parte derecha de las reglas
 - ❑ Encadenamiento hacia adelante → parte izquierda de las reglas
- ❑ Existencia de variables → encaje con ligadura de variables
- ❑ Los mecanismos de encadenamiento utilizan diversas estrategias para agilizar el reconocimiento de reglas
 - ❑ Se pueden indexar las reglas para mejorar la eficiencia
 - ❑ Con encadenamiento hacia delante se indexan por el antecedente
 - ❑ Con encadenamiento hacia atrás se indexan por el consecuente
 - ❑ *Tablas hash*
- ❑ Algoritmo de encaje más eficiente que el encaje uno a uno

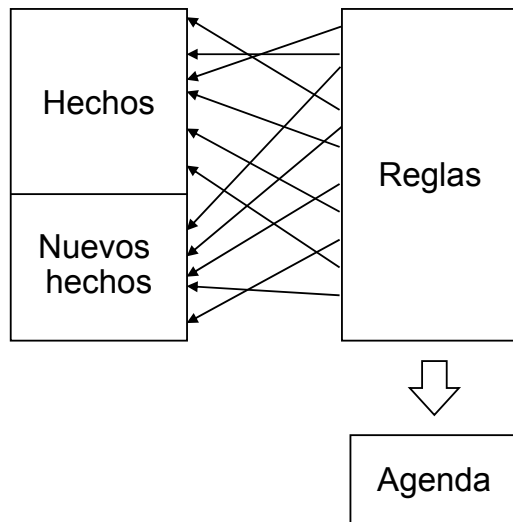
Implementación: fase de reconocimiento

- ❑ El encaje uno a uno es muy poco eficiente:
 - ❑ Si en cada ciclo del proceso de inferencia llevado a cabo en el encadenamiento hacia delante hubiera que comprobar todas las reglas aplicables en función de la memoria de trabajo, el sistema sería muy poco eficiente
 - ❑ Teniendo en cuenta que en cada ciclo los cambios producidos en la base de hechos son los únicos que influyen en la modificación del conjunto conflicto, se toman dichas actualizaciones como base del cálculo de las reglas que pasarán a formar parte de este conjunto (*redundancia temporal*)
 - ❑ De esta forma, los nuevos hechos que son introducidos en la memoria de trabajo son los que determinan las nuevas reglas aplicables
 - ❑ Idea del algoritmo RETE: en lugar de buscar qué reglas satisfacen los hechos existentes en cada momento, son los nuevos hechos generados en cada ciclo los que buscan o determinan qué nuevas reglas se seleccionan

Implementación: fase de reconocimiento

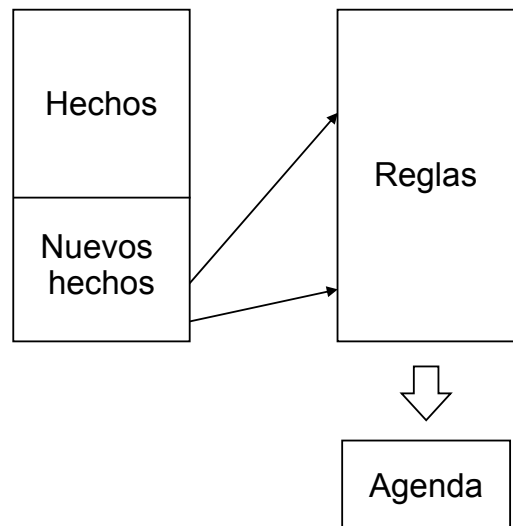
❑ Método tradicional

- ❑ Cada regla verifica si se cumplen sus condiciones a partir de la base de hechos total



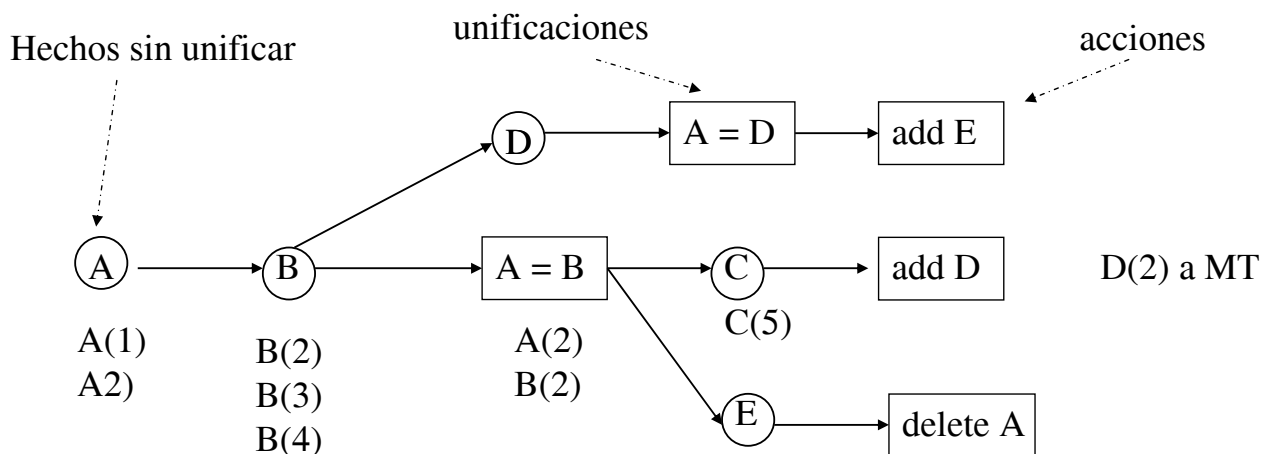
❑ Algoritmo RETE

- ❑ Los nuevos hechos buscan reglas



Ejemplo de RETE

- ❑ R1 - $A(x), B(x), C(x) \Rightarrow \text{add } D(x)$
- ❑ R2 - $A(x), B(x), D(x) \Rightarrow \text{add } E(x)$
- ❑ R3 - $A(x), B(x), E(x) \Rightarrow \text{delete } A(x)$
- ❑ MT - $\{A(1), A(2), B(2), B(3), B(4), C(5)\}$



Algoritmo RETE (Forgy, 1982)

- ❑ Desarrollado por Forgy para el shell de sistemas expertos OPS en Carnegie-Mellon durante su tesis doctoral
 - ❑ Cap. 11 de “Expert Systems”. Giarratano & Riley
- ❑ Algoritmo que realiza un encaje de patrones eficiente (muchos a muchos) entre la base de reglas y la base de hechos
 - ❑ Evita reevaluar condiciones ya evaluadas
 - ❑ Aprovecha la **redundancia temporal**: los posibles cambios en la agenda de un ciclo al siguiente son pocos, están perfectamente acotados y pueden ser fácilmente determinados
 - ❑ Aprovecha la **similitud estructural**: muchas reglas comparten condiciones
 - ❑ Construye un grafo de dependencias estableciendo las reglas que comparten condiciones
 - ❑ En un ciclo de operación, las nuevas reglas que se añaden a la agenda son aquéllas que dependían de condiciones que acaban de hacerse ciertas al aplicar la regla anterior

Implementación: fase de selección

- ❑ Estrategias de control
 - ❑ En cada ciclo de operación del sistema puede haber más de una instancia de regla candidata a la ejecución
 - ❑ ¿Qué regla seleccionar?
- ❑ Hay dos aproximaciones generales al control de los sistemas basados en reglas
 - ❑ **Control global**: control independiente del dominio de aplicación
 - ❑ Estrategias implementadas en el intérprete
 - ❑ No son modificables por el programador
 - ❑ **Control local**: control dependiente del dominio de aplicación
 - ❑ Reglas especiales que permiten razonar sobre el control: META-REGLAS
 - ❑ El programador escribe reglas explícitas para controlar el sistema (o son aprendidas automáticamente por éste)

Implementación: fase de selección

☐ Control global

- ☐ Determina qué reglas participan en el proceso de reconocimiento de patrones y cómo se elegirá entre las instancias de regla en el caso de que exista más de una candidata

☐ Estrategias de resolución del conjunto conflicto

- ☐ Estrategias para la selección de la regla del conjunto conflicto que será disparada en cada ciclo
- ☐ Los criterios de selección persiguen que el sistema sea sensible y estable
 - ☐ Sensible: responde a cambios reflejados en la memoria de trabajo
 - ☐ Estable: hay continuidad en la línea de razonamiento
- ☐ Hay diversos mecanismos de resolución de conflictos

Criterios de selección de reglas

☐ Los **criterios** de selección más populares son

- ☐ **Orden** en la base de reglas
 - ☐ Se selecciona la primera regla aplicable, suponiendo que exista un orden lineal explícito en la BR, lo cual no siempre es deseable
- ☐ **Prioridades** asociadas a las reglas (*conocimiento heurístico*)
 - ☐ Se selecciona la regla de mayor prioridad
 - ☐ La prioridad la establece el experto del dominio
- ☐ **Especificidad**: tienen prioridad las reglas más específicas, con mayor número de condiciones (*búsqueda más convergente*)
- ☐ **Novedad**: tienen prioridad las instancias de reglas que utilizan hechos más recientemente añadidos a la memoria de trabajo
- ☐ **Refracción**: una regla no debe poderse disparar más de una vez con los mismos hechos (*evita bucles deductivos*)
- ☐ **Arbitrariedad**
 - ☐ Cuando no existen criterios adicionales disponibles (*no sistemática*)

Organización de reglas

❑ Módulos

- ❑ Podemos tener la base de reglas dividida en bloques o módulos
 - ❑ Ejemplo: módulo de averías mecánicas y módulo de averías eléctricas
Se pregunta al usuario qué tipo de avería tiene para ir a uno u otro módulo o se usa una regla de diagnóstico general
- ❑ Los módulos permiten estructurar la base de reglas, haciendo más eficiente la selección de reglas

❑ Agregación de reglas (*chunking*)

- ❑ Las reglas se corresponden con pasos muy elementales en la resolución de problemas
- ❑ Algunos sistemas poseen capacidad de aprendizaje de macro-reglas formadas por agregación (aplicación sucesiva) de reglas elementales

Selección de reglas

❑ Uso de **meta-reglas** para el control local

- ❑ Problema (resolución de conflictos): tenemos varias reglas y hay que seleccionar una
- ❑ Conocimiento de control:

SI **condición** ENTONCES **seleccionar Regla X**

- ❑ Mecanismo declarativo para establecer estrategias de control
- ❑ Número mucho menor de meta-reglas que de reglas
- ❑ Pueden ser introducidas por el diseñador del sistema
- ❑ O pueden ser aprendidas automáticamente
(si el sistema dispone de capacidad de aprendizaje)

Sistemas de producción en Prolog

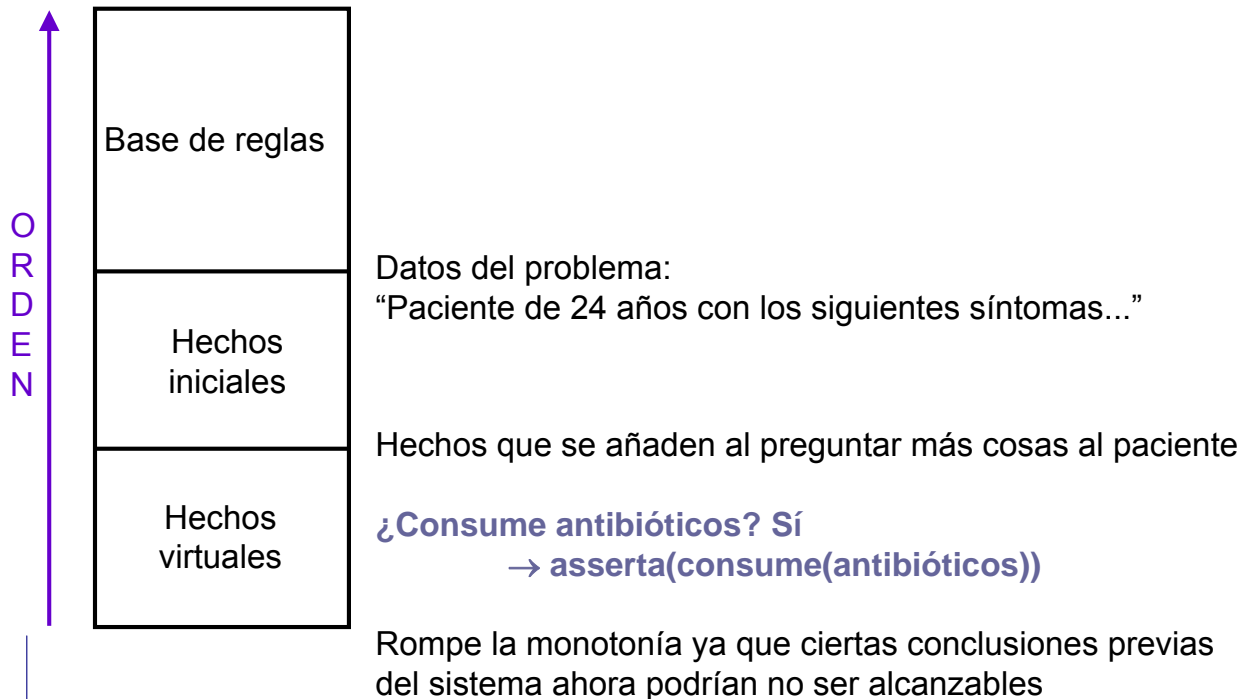
- ❑ Componentes de un sistema de producción
 - ❑ Base de reglas
 - ❑ Memoria de trabajo o base de hechos
 - ❑ Intérprete de reglas
- ❑ Prolog nos proporciona el intérprete
 - ❑ “Sólo falta” construir la base de reglas y la base de hechos
- ❑ Características del intérprete de reglas de Prolog
 - ❑ Encadenamiento hacia atrás
 - ❑ Selección de subobjetivos de izquierda a derecha
 - ❑ Primero en profundidad con *backtracking*

Sistemas de producción en Prolog

- ❑ Predicados extra-lógicos útiles en la gestión del conocimiento
 - asserta(Cláusula)** Se coloca la primera en la BC
 - assertz(Cláusula)** Se coloca la última en la BC
- ❑ Uso: posibilidad de añadir cláusulas en tiempo de ejecución
 - ❑ Permite añadir conocimiento para mejorar la eficiencia
 - ❑ Si demostramos un cierto predicado podemos añadirlo dinámicamente para no tener que demostrarlo cada vez
 - ❑ Ejemplo: **$a(X) :- b(X), c(X), d(X).$**
Para no demostrar *b*, *c*, *d* cada vez podemos cambiar la regla por
 $a(X) :- b(X), c(X), d(X), \text{asserta}(a(X)).$
 - ❑ El sistema sigue siendo monótono: no se añade nada nuevo
 - ❑ Permite añadir **hechos virtuales** que pueden romper la monotonía del sistema
 - ❑ Información añadida por el usuario como respuesta a las preguntas realizadas por el sistema

Sistemas de producción en Prolog

Base de conocimiento



Sistemas de producción en Prolog

☐ Predicados de entrada/salida

`write(Término)` `nl` `read(Respuesta)` `tab(N)`
`consult(Fichero)` `[Fichero]`

(añade el contenido del fichero al final de la BC actual)

☐ Es menos eficiente (y mucho menos modular ;-) tener todas las reglas en la base de conocimiento

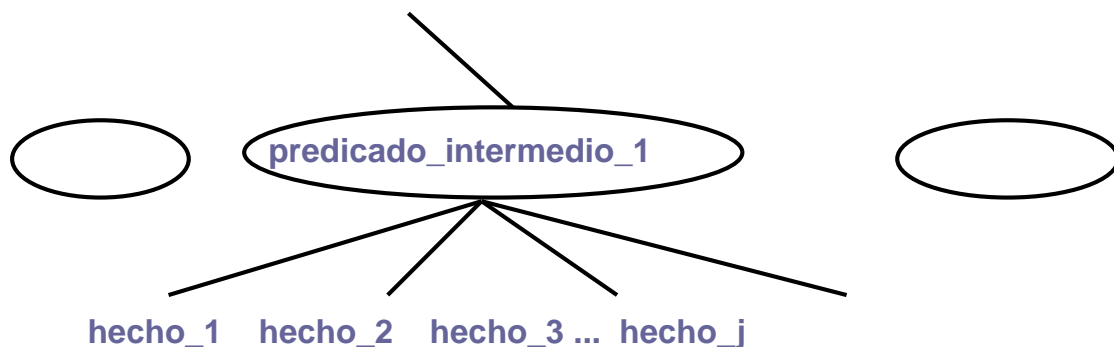
- ☐ Repartir las reglas en varios ficheros
- ☐ En la base de reglas inicial se pueden tener las reglas básicas para clasificar problemas
- ☐ Y, en función del problema, cargar su fichero asociado

Sistemas de producción en Prolog

- ❑ Sistemas con encadenamiento hacia atrás
 - ❑ Por ejemplo, sistemas de diagnóstico: **diagnóstico(X)**
X: problema de batería, problema de motor de arranque....
- ❑ Sintaxis: **si** condición **entonces** acción
acción :- condición
 - ❑ Prolog indexa las reglas por el consecuente
 - ❑ El predicado **diagnóstico** aparecerá a la izquierda
 - ❑ Los objetivos correspondientes a hechos observables en el problema (iniciales) y resultantes de las preguntas al usuario (virtuales) aparecerán a la derecha de las reglas (en las condiciones)
- ❑ El sistema de producción tiene una estructura en varios niveles
 - ❑ Nivel superior: predicados de salida
 - ❑ Niveles intermedios: predicados intermedios que agrupan características comunes e influyen en la eficiencia del problema
 - ❑ Nivel inferior: hechos que son la entrada al sistema

Sistemas de producción en PROLOG

diagnostico(a) diagnostico(b) diagnostico(c) ... diagnostico(k)



Los predicados intermedios aparecen tanto a la izquierda como a la derecha de las reglas y hacen que el sistema sea más eficiente

predicado_intermedio_1 :-

hecho_1, hecho_2, hecho_3, ..., hecho_j.

Ejemplo

- ❑ Sistema experto basado en reglas para detectar averías
 - ❑ Se supone que se usará introduciendo:
?- diagnóstico(X).
 - ❑ Ejemplos de reglas de diagnóstico
diagnóstico('Fusible fundido') :-
no_funciona, no_hay_luz.
diagnóstico('Fusible fundido') :-
ruido(pop).
diagnóstico('Cable roto') :-
no_funciona, cable_en_mal_estado.
 - ❑ Cadenas de caracteres en Prolog: alternativa a usar constantes
como *fusible_fundido*, *cable_roto*
 - ❑ Por supuesto, hay que definir estos predicados intermedios

Ejemplo

- ❑ Implementación de hechos virtuales
 - ❑ Para las reglas que necesiten solicitar información
pregunta(P, R) :-
write('¿'), write(P), write('?'), read(R), nl.
 - ❑ Para no hacer la misma pregunta más de una vez, debemos
guardar las preguntas contestadas
pregunta(P, R) :-
write('¿'), write(P), write('?'), read(R), nl,
asserta(pregunta(P, R)).
 - ❑ Las asertamos por delante de esta regla
 - ❑ Ejemplo
diagnóstico('Fusible fundido') :-
pregunta('El aparato está inoperante', sí),
pregunta('Se ha ido la luz en la casa', sí).
 - ❑ Demasiadas repeticiones en el código para un sistema grande...

Ejemplo

- ❑ Codificación de las respuestas
 - ❑ Permite agrupar las respuestas de los usuarios en categorías
 - ❑ Respuestas afirmativas y negativas (*lo más simple*)

afirmativa(sí).	negativa(no).
afirmativa(s).	negativa(n).
afirmativa(si).	negativa(nunca).
afirmativa(ok).	negativa(imposible).
afirmativa(claro).	negativa(jamás).

- ❑ Nuevo predicado *pregunta_si/1*: éxito si respuesta afirmativa, fallo si negativa

pregunta_si(P) :-

pregunta(P, R), respuesta_afirmativa(P, R).

Ejemplo

- ❑ Si esperamos una respuesta negativa en lugar de una afirmativa

pregunta_si_no(P) :-

\+pregunta_si(P).

- ❑ Tendremos en cuenta también la posibilidad de que la respuesta no esté clara: no se sepa si es afirmativa o negativa

respuesta_afirmativa(P, R) :- afirmativa(R).

respuesta_afirmativa(CódigoP, R) :-

**\+negativa(R), \+afirmativa(R),
write('Por favor, responda sí o no. '),
read(R2),
retract(preguntado(CódigoP, R)),
asserta(preguntado(CódigoP, R2)),
respuesta_afirmativa(CódigoP, R2).**

Ejemplo

❑ Codificación de las preguntas

- ❑ Incrementa la legibilidad del código
- ❑ Evita repeticiones en el código de la misma pregunta

diagnóstico('Fusible fundido') :-

pregunta_si(no_funciona),

pregunta_si(no_hay_luz).

códigoP(no_funciona, 'El aparato está inoperante').

códigoP(no_hay_luz, 'Se ha ido la luz en la casa').

❑ Necesitamos redefinir *pregunta/2*

pregunta(CódigoP, R) :- preguntado(CódigoP, R).

pregunta(CódigoP, R) :-

\+preguntado(CódigoP, R), write('¿'),

códigoP(CódigoP, P), write(P), write('? '),

read(R).

Ejemplo

- ❑ Preguntas similares: códigos con argumentos
- ❑ En lugar de un hecho, usamos una regla que se encarga de escribir la parte común
- ❑ El argumento que se use es lo que diferencia la pregunta

códigoP(ruido(X), X) :-

write('Ha escuchado un ruido como ').

?- pregunta_si(ruido(pop)).

¿Ha escuchado un ruido como pop? sí.

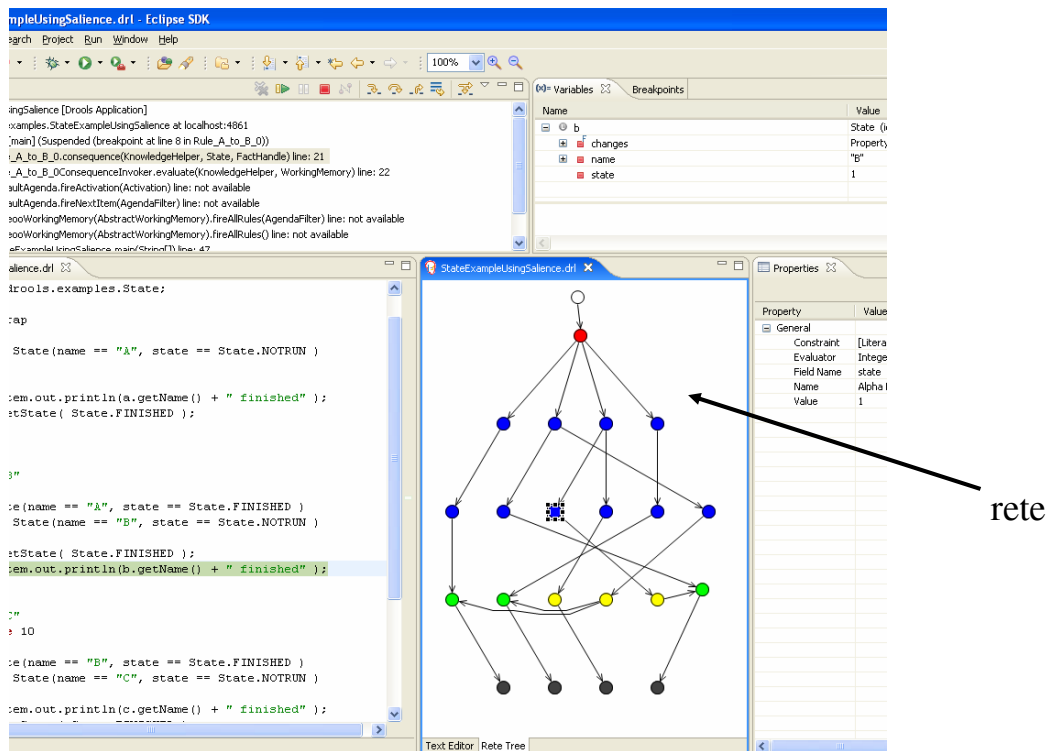
diagnóstico('Fusible fundido') :-

pregunta_si(ruido(pop)).

- ❑ Lo único que falta es declarar *preguntado/2* como dinámico y usar *Sicstus Prolog* (la E/S no funciona muy bien en SWI-Prolog)

:- dynamic preguntado/2.

Sistemas de Reglas: Drools



Example of Drools application interface showing the Rete tree structure and associated variables.

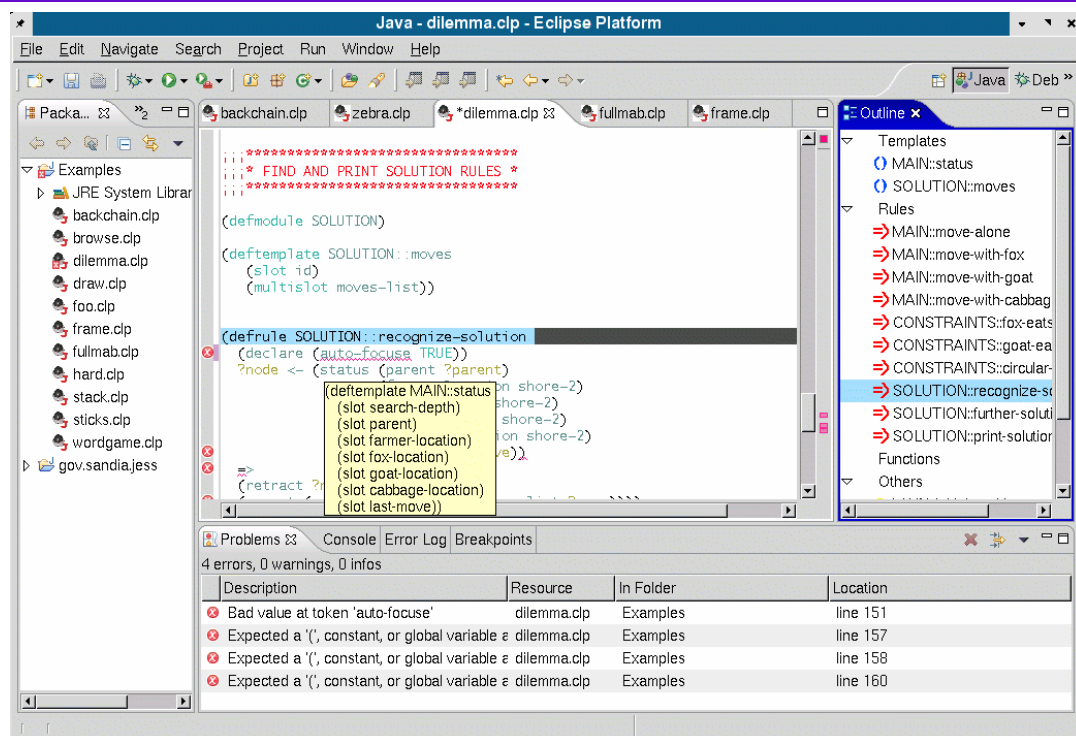
<http://www.jboss.org/drools/>

Características: <http://www.jboss.org/drools/featuresandscreenshots.html>

IAIC – Curso 2008-09

Tema 3.3 - 65

Sistemas de Reglas: Jess



Example of Jess application interface showing the rule definition and associated variables.

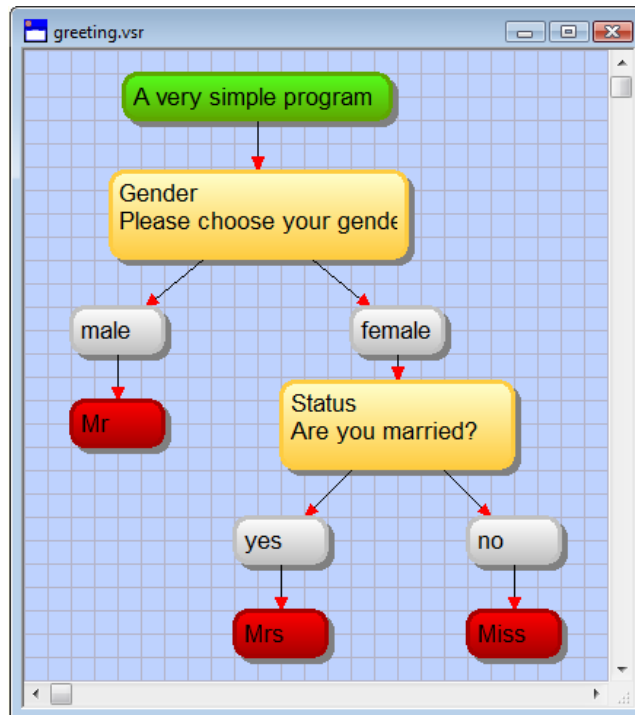
<http://www.jessrules.com/>

Demo: <http://www.jessrules.com/jessdemo/>

IAIC – Curso 2008-09

Tema 3.3 - 66

Sistemas de Reglas: VisiRules (LPA)



<http://www.lpa.co.uk/vsr.htm>

Demos online : http://www.lpa.co.uk/vrs_dem.htm

Bibliografía

- ❑ **Rich, E. y Knight, K.**
Artificial Intelligence.
McGraw-Hill, 1991, 2ª edición.
- ❑ **Russell, S. y Norvig, P.**
Inteligencia Artificial: Un Enfoque Moderno.
Prentice Hall, 2004, 2ª edición.
- ❑ **Luger, G.F.**
Artificial Intelligence.
Addison-Wesley, 2005, 5ª edición.
- ❑ **Nilsson, J.**
Artificial Intelligence: A New Synthesis.
Prentice Hall, 2004, 2ª edición.

- ❑ **Jackson, P.**
Introduction to Expert Systems.
Addison-Wesley, 1999.
- ❑ **Gonzalez, A. J. y Dankel, D. D.**
The Engineering of Knowledge Based Systems:
Theory and Practice.
Prentice Hall, 1993.
- ❑ **Rowe, Neil C.**
Artificial Intelligence through Prolog.
Prentice-Hall, 1988.
- ❑ **Giarratano, J. y Riley, G.**
Sistemas Expertos. Principios y Programación
Thomson, 2001.