



Representación del conocimiento

Sistemas basados en Reglas.





Contenido

1. Introducción.
2. Componentes.
3. Lenguajes.
4. Inferencia en un sistema de producción.
5. Encadenamiento hacia adelante.
6. Encadenamiento hacia atrás.
7. Lenguajes con variables.
8. Algoritmo de Rete.



1. Introducción



Introducción

- Los Sistemas Basados en Reglas o Sistemas de producción se caracterizan por utilizar una única estructura para representar el conocimiento: REGLA
- Esquema de Regla:
 - Condición → Acción
 - Antecedente → Consecuente
 - Lado Izquierdo → Lado Derecho (LHS → RHS)
 - Si ... Entonces ...



Un poco de historia

- Post, 43: reglas de reescritura.
- Nilson, 60's: modelo de cómputo, soporte de la búsqueda en espacio de estados.
- Newell y Simon, 70's: modelo psicológico razonamiento humano.
- Representación conocimiento: mediados 70's.



Interés

- Formalismo adecuado para representar conocimiento heurístico: asociaciones entre los elementos del antecedente y del consecuente
 - Si el coche no arranca y las luces no se encienden
Entonces revisar la batería
- Existen numerosos dominios (parcialmente) gobernados por reglas deterministas:
 - Si el semáforo está rojo
Entonces los coches tienen que parar.
 - Si los ingresos anuales netos por rentas del trabajo son inferiores a 20,000.00€
Entonces no están sometidos al impuesto de la renta.



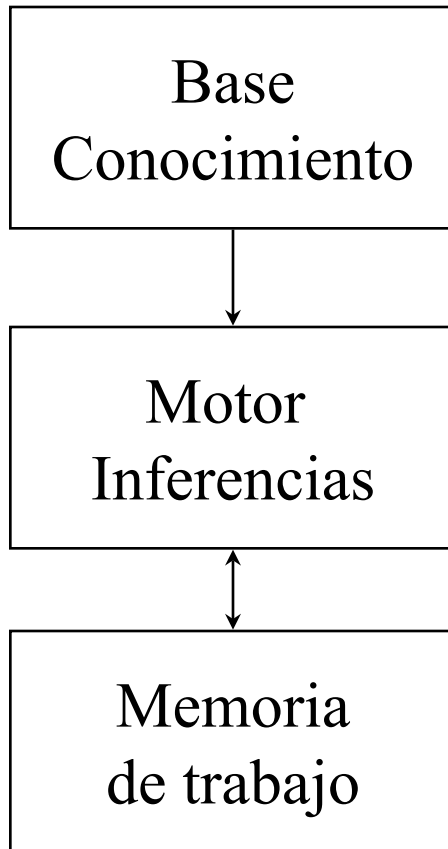
Popularidad

- Sencillez: una única estructura, simple.
- Eficiencia: mecanismos deductivos computacionalmente más eficientes que los de la lógica.

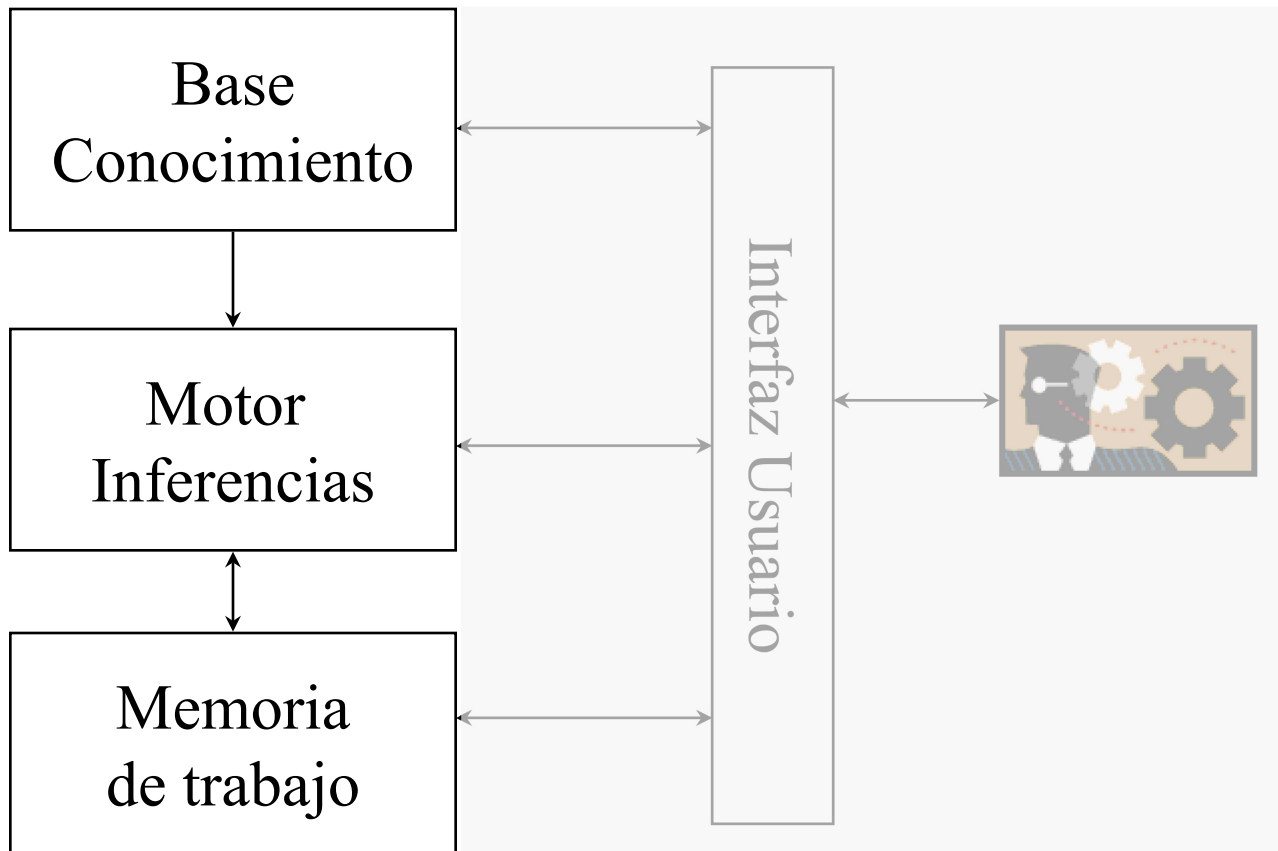


2. Componentes de una sistema de producción

2 .Componentes de una sistema de producción



2 .Componentes de una sistema de producción





Componentes de una sistema de producción

- Base de Conocimiento (BC, KB)
 - Base de Reglas (BR): conjunto de reglas de producción.
 - Declaración de Dominio: declaración de los elementos básicos que se referencian en hechos y reglas.
- Memoria de Trabajo (MT, WM)
 - Conjunto de hechos:
 - Información que se considera cierta.
- Motor de inferencias (MI, IE)
 - Generar nuevos hechos, a partir de MT, usando BR.



3. Lenguajes



Lenguajes

- Elementos básicos de representación.
 - Hechos: información que se posee relevante a una instancia o problema concreto
 - La casa es verde
 - La temperatura del paciente es de 39°C
 - Reglas: conocimiento sobre el dominio del problema que permite derivar hechos adicionales.
- Variantes
 - Según se referencien elementos del dominio.
 - Tripletes objeto-atributo-valor: `casa.color=verde`
 - Patrones simbólicos: `(casa color verde)`
 - Según funcionamiento motor inferencias.
 - encadenamiento hacia delante / hacia atrás-
 - Otros.
 - incertidumbre, meta-conocimiento, ...



Lenguaje objeto-atributo valor

- Conceptualización del universo en tripletes objeto, atributo y valor
 - Objeto: paciente
 - Atributo: fumador
 - Valor: si
- Ejemplo de hecho:
 - paciente.fumador=si
- Ejemplo de regla:
 - Si iguales(paciente, fumador, si)
 - Entonces añadir(paciente, grupo-de-riesgo, si)



Declaración de dominio

- Declaración de objetos (O)

$$O = \{O_1, O_2, \dots, O_n\}$$

- Declaración atributos (DA)

	univaluada	multivaluada
tipada	$O_i.a_j^s:\tau$	$O_i.a_j^m:2^\tau$
no tipada	$O_i.a_j^s$	$O_i.a_j^m$

- Declaración de dominio: $DD = O \cup DA$



Ejemplo declaración de dominio

- $O = \{\text{paciente-1}\}$
- $DA = \{\text{paciente-1.sexo}^s: \{\text{varón, hembra}\},$
 $\text{paciente-1.edad}^s: \text{int},$
 $\text{paciente-1.síntoma}^m,$
 $\text{paciente-1.enfermedad}^m: 2^{\{\text{aneurismaAortico, estenosisArterial}\}}\}$
- Declaración de dominio: $DD = O \cup DA$



Ejemplo declaración de dominio

- $O = \{l1, l2, s1, s2, s3, cb1, cb2, p1, p2, w1, w2, w3, w4, w5, w6, outside\}$
- $DA = \{$
 - $l1.live^s: boolean,$
 - $l1.light^s: boolean,$
 - $l1.lit^s: boolean,$
 - $l1.ok^s: boolean,$
 - $l1.conectado^s,$
 - $s1.estado^s: \{up, down\},$
 - $s1.ok^s: boolean,$
 - $cb1.ok^s: boolean,$
 - $p1.conectado^s,$
 - $w1.conectado^s, \dots \}$
- Declaración de dominio: $DD = O \cup DA$



Objeto-atributo-valor: hechos

- Sintaxis hechos:

$O_i.a=c$, si $x^s:\tau$, $c \in \tau$

$O_i.a=C$, si $x^m:2^\tau$, $C \subseteq \tau$

- Ejemplos:

paciente-1.sexo=varón

paciente-1.edad=25

paciente-1.síntoma={dolorPecho, calambresPiernas}

paciente-1.enfermedad={estenosisArterial}



Objeto-atributo-valor: hechos

- Ejemplos:
 - out.live=t
 - l1.ok=t
 - l1.light=t
 - l2.ok=t
 - l2.light=t
 - l1.concetado=w0
 - s1.estado=down
 - s2.estado=up



Memoria de trabajo

- Memoria de trabajo:

$$H = \{ O_1.x_1 = c_1, \dots, O_k.x_p = c_p, O_l.y_1 = C_1, \dots, O_r.y_q = C_q \}$$

con

- c_i constantes, C_i conjuntos de constantes.
- Declaración de objetos $\{O_1, \dots, O_r\}$.
- Declaración de atributos $O_j.x_i^s, O_j.y_i^m$.
- Si declaraciones tipadas, restricciones de tipo.
- Cada objeto-atributo solo ocurre una vez.



Memoria de trabajo: Ejemplo

- $O = \{\text{paciente-1}\}$
- $DA = \{\text{paciente-1.sexo}^s: \{\text{varón, hembra}\},$
 $\text{paciente-1.edad}^s: \text{int},$
 $\text{paciente-1.síntoma}^m,$
 $\text{paciente-1.enfermedad}^m: 2^{\{\text{aneurismaAortico, estenosisArterial}\}}\}$
- $H = \{\text{paciente-1.sexo} = \text{varón}, \text{paciente-1.edad} = 25,$
 $\text{paciente-1.síntoma} = \{\text{dolorPecho, calambresPiernas}\},$
 $\text{paciente-1.enfermedad} = \{\text{estenosisArterial}\}$
 $\}$



Memoria de trabajo: Ejemplo

- $O = \{l1, l2, s1, s2, s3, cb1, cb2, p1, p2, , w0, w1, w2, w3, w4, w5, w6, outside\}$
- $DA = \{l1.live^s: boolean, l1.light^s: boolean, l1.lit^s: boolean, l1.ok^s: boolean, l1.conectado^s, s1.estado^s: \{up, down\}, s1.ok^s: boolean, cb1.ok^s: boolean, p1.conectado^s, w1.conectado^s, \dots \}$
- $H = \{out.live=t, l1.ok=t, l1.light=t, l2.ok=t, l2.light=t, l1.concetado=w0, s1.estado=down, s2.estado=up, \dots \}$



Objeto-atributo-valor: reglas

■ Sintaxis

<reglaProducción>	::= if <antecedente> then <consecuente> fi
<antecedente>	::= <disyunción> { and <disyunción>}*
<disyunción>	::= <condición> { or <condición>}*
<consecuente>	::= <conclusión> { also <conclusión>}*
<condición>	::= <predicado> (<objeto> , <atributo> , <constante>)
<conclusión>	::= <acción> (<objeto> , <atributo> , <constante>)
<predicados>	::= iguales noiguales mayorque ...
<acción>	::= añadir eliminar ...



Ejemplo regla

- Lenguaje natural

Si el paciente tiene un dolor abdominal y
en la auscultación se percibe un rumor abdominal y
se siente una masa pulsante en el abdomen
entonces el paciente tiene un aneurisma aórtico

- Regla de producción

```
if iguales(paciente-1, síntoma, dolorAbdominal) and  
    iguales(paciente-1, evidencia, rumorAbdominal) and  
    iguales(paciente-1, evidencia, masaPulsante)  
then  
    añadir(paciente-1, enfermedad, aneurismaAortico)  
fi
```




Ejemplo regla

- Si una bombilla tiene tensión y funciona correctamente, entonces la bombilla luce:

```
if  iguales(l1, light, t) and  
    iguales(l1, live, t) and  
    iguales(l1, ok, t)  
then  
    añadir(l1, lit, t)  
fi
```



Condición

- Se construyen con un predicado y sus argumentos: objeto, atributo y constante:
 - iguales(paciente, edad, 60)
- Expresa una comparación (según el predicado) entre la constante especificada y el valor asociado al atributo del objeto en la memoria de trabajo:
 - Hecho: paciente.edad=50
 - Condición: iguales(paciente, edad, 60)
- Los predicados son funciones booleanas:
 - iguales(paciente, edad, 60) se evaluará a falso



Condición

- Ejemplo dominio enfermedades

$H = \{\text{paciente.edad} = 50, \text{paciente.síntoma} = \{\text{fiebre}, \text{dolorAbdominal}\}\}$

$\text{iguales}(\text{paciente}, \text{edad}, 60) \Rightarrow F$

$\text{iguales}(\text{paciente}, \text{edad}, 50) \Rightarrow T$

$\text{iguales}(\text{paciente}, \text{síntoma}, \text{fiebre}) \Rightarrow T$



Condición

- Ejemplo asistente diagnóstico

$H = \{ \text{out.live} = t, \text{l1.ok} = t, \text{l1.light} = t, \text{l2.ok} = t, \text{l2.light} = t, \text{l1.concetado} = w0, \text{s1.estado} = \text{down}, \text{s2.estado} = \text{up}, \dots \}$

$\text{iguales}(\text{l1}, \text{light}, t) \Rightarrow T$

$\text{iguales}(\text{l1}, \text{ok}, t) \Rightarrow T$

$\text{iguales}(\text{s1}, \text{estado}, \text{up}) \Rightarrow F$



Antecedente de la regla

- Se satisface si se evalúa a cierto
- $H = \{out.live=t, l1.ok=t, l1.light=t, l2.ok=t, l2.light=t, l1.concetado=w0, s1.estado=down, s2.estado=up, \dots\}$
if **iguales**(l1, light, t) **and**
 iguales(l1, live, t) **and**
 iguales(l1, ok, t)
then
 añadir(l1, lit, t)
fi
- Antecedente:
 - **iguales**(l1, light, t) **and** **iguales**(l1, live, t) **and** **iguales**(l1, ok, t)
 - ¿Se satisface?



Semántica de los predicados

- La interpretación de los predicados en un sistema de producción es OPERACIONAL.
- iguales(o,a,c): cierto si \exists hecho o.a=c en memoria de trabajo.
 - El comportamiento depende del motor de inferencias:
 - Hacia adelante: sólo consulta contenido actual de memoria de trabajo.
 - Hacia atrás: puede forzar la búsqueda del hecho.
- Diferencia con la lógica: iguales(o,a,c) es cierto si y solo si se puede derivar o.a=c



Predicado noiguales

- ¿Qué tipo de negación?
- Hacia adelante: habitualmente, variante limitada de mundo cerrado
 - noiguales(o, a, c) cierto si no $\exists o.a=c$ en la memoria de trabajo cuando se evalúa el predicado
- Hacia atrás: habitualmente, negación por fallo.
 - Si se ha buscado y ha fracasado, éxito
 - Si no se ha buscado,
 - Fuerza la búsqueda
 - Si fracaso, éxito



Semántica de predicados

Predicado	Univaluado	Multivaluado
iguales(o, a, c)	$o.a = c$	$o. a = C, c \in C$
noiguales(o, a, c)	$o.a \neq c$ ($o.a = c$ o no \exists hecho para o.a)	$o.a \neq c$ ($o.a = C, c \notin C$ o no \exists hecho para o.a)
menorque(o, a, c)	$o.a < c$	-
mayorque(o, a, c)	$o.a > c$	-



Conclusiones

- Secuencia de acciones que pueden modificar la memoria de trabajo.
- Solo se ejecutan si el antecedente de la regla se satisface.
- Disparar una regla: realizar la(s) acción(es) de la conclusión.



Acción básica: añadir

- añadir(o,a,c)
- Atributos univaluados:
 - Tras realizar la acción, la memoria de trabajo contiene $o.a=c$ y ningún otro hecho para o.a
- Atributos multivaluados:
 - Si antes de realizar la acción no $\exists o.a=C$ en la memoria de trabajo, se añade $o.a=\{c\}$
 - Si antes de realizar la acción $\exists o.a=C$ en la memoria de trabajo, se añade $o.a=C \cup \{c\}$



Ejemplo añadir

$H = \{\text{paciente.edad} = 50, \text{paciente.síntoma} = \{\text{fiebre}\}\}$

`añadir(paciente, edad, 60)`

$H = \{\text{paciente.edad} = 60, \text{paciente.síntoma} = \{\text{fiebre}\}\}$

`añadir(paciente, síntoma, mareos)`

$H = \{\text{paciente.edad} = 60, \text{paciente.síntoma} = \{\text{fiebre}, \text{mareos}\}\}$

`añadir(paciente, sexo, varón)`

$H = \{\text{paciente.edad} = 60, \text{paciente.sexo} = \text{varón},$
 $\text{paciente.síntoma} = \{\text{fiebre}, \text{mareos}\}\}$



Acción eliminar

- eliminar(o, a, c)
- Acción no disponible en un lenguaje lógico.
- Elimina el hecho de la memoria de trabajo (el valor, si multivaluado y varios valores).



Ejemplo eliminar

$H = \{\text{paciente.edad} = 60, \text{paciente.sexo} = \text{varón}, \text{paciente.síntoma} = \{\text{fiebre}, \text{mareos}\}\}$

`eliminar(paciente, síntoma, fiebre)`

$H = \{\text{paciente.edad} = 60, \text{paciente.sexo} = \text{varón}, \text{paciente.síntoma} = \{\text{mareos}\}\}$

`eliminar(paciente, síntoma, mareos)`

$H = \{\text{paciente.edad} = 60, \text{paciente.sexo} = \text{varón}\}$

`eliminar(paciente, edad, 60)`

$H = \{\text{paciente.sexo} = \text{varón}\}$

Ejercicio 1: versión reducida del asistente al diagnóstico

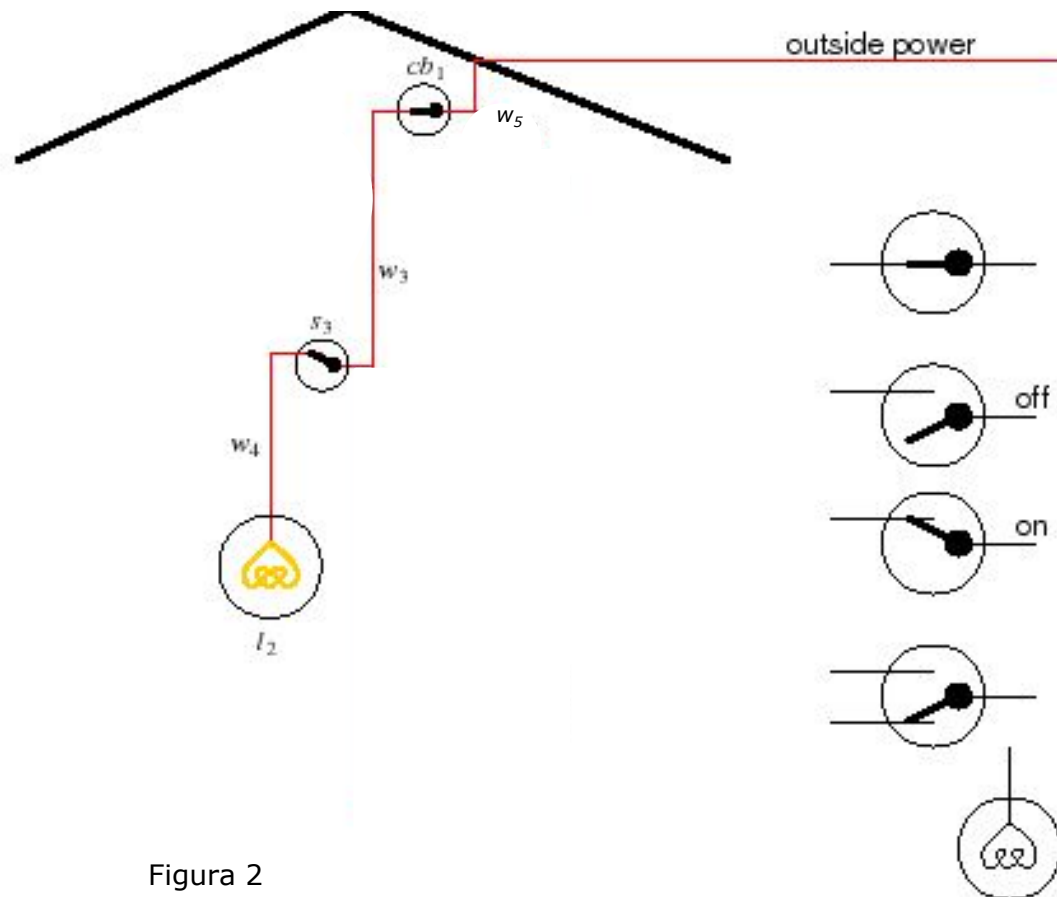


Figura 2



Base de conocimiento para la versión reducida del asistente al diagnóstico

- Desarrollar una base de conocimiento de un sistema basado en reglas con el formalismo o-a-v y la sintaxis y semántica de hechos y reglas presentada en estas notas. La base de conocimiento debe permitir determinar a que elementos llega la señal y si la bombilla luce en el ejemplo reducido del asistente al diagnóstico, descrito en la transparencia anterior.
- Recordar que se deben proporcionar:
 - Declaración de dominio.
 - Reglas generales: reglas de la base de conocimiento.
 - Descripción de la instancia específica:
 - Hechos iniciales de la memoria de trabajo.
 - Si la descripción incluye reglas, también se incluyen en la base de conocimiento.



4. Inferencia en un sistema de producción



Inferencia en un sistema de producción

- Esencialmente, seleccionar reglas cuyo antecedente se satisface y realizar su acción.
- Utilizan la diferencia entre reglas y hechos:
 - Los hechos se gestionan globalmente, en la memoria de trabajo
 - Hechos iniciales: conocimiento primitivo.
 - Las reglas se utilizan para derivar nuevos hechos
 - Nuevos hechos: conocimiento derivado.
- Un proceso de búsqueda que examina primero los hechos, después las reglas.



Objetivo

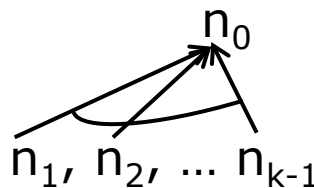
- El concepto de pregunta se reemplaza por el de meta: "valor del atributo de un objeto".
- Declaración de metas: ampliación declaración de dominio
 - metas: subíndice **g**

	univaluada	multivaluada
tipada	$O_i.a_g^s:\tau$	$O_i.a_g^m:2^\tau$
no tipada	$O_i.a_g^s$	$O_i.a_g^m$

- No es imprescindible declarar la meta
 - El sistema dispara todas las reglas posibles

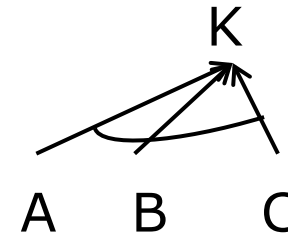
Espacio de búsqueda: hipergrafos o grafos y/o

- Generalización del concepto de grafo: Nodos e hiperarcos o k-conectores
- $HG = \{N, HA\}$, N conjunto de nodos, HA conjunto de hiperarcos
- Nos interesan hipergrafos **dirigidos**
- 1-conector: arco de un grafo dirigido convencional
 - (n_0, n_1)
 - Gráficamente: $n_0 \leftarrow n_1$
- k-conector, $k > 1$: generaliza el concepto de arco
 - k-conector: $(n_0, n_1, n_2, \dots, n_{k-1})$
 - Gráficamente

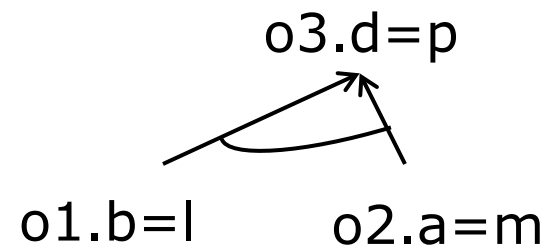


Representación de reglas mediante hiperarcos: conjunción

- $A, B, C \dashrightarrow K$

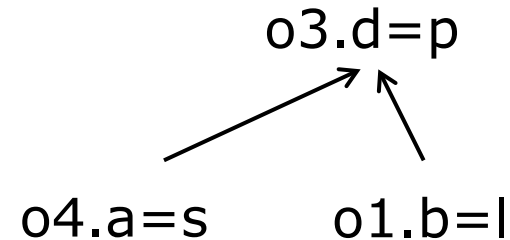


if iguales(o1,b, l) **and**
iguales(o2,a, m)
then añadir (o3, d, p) **fi**



Representación de reglas mediante hiperarcos: disyunción

if iguales(o4,a, s) **or**
iguales(o1,b, l)
then añadir (o3, d, p) **fi**



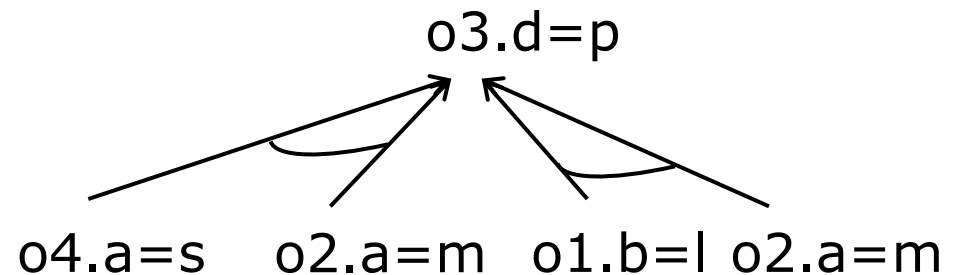
if iguales(o4, a, s) **or**
iguales(o1,b, l) **and**
iguales(o2,a, m)
then añadir (o3, d, p) **fi**


?



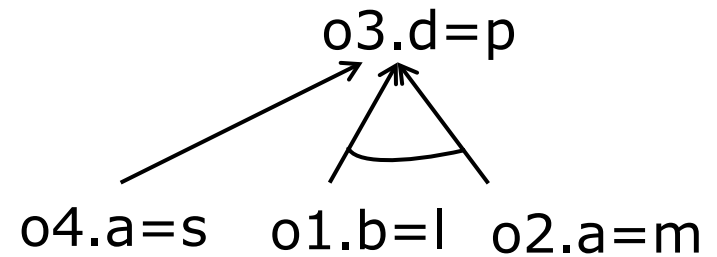
if iguales(o4,a,s) **or**
iguales(o1,b,l) **and**
iguales(o2,a,m)
then añadir (o3,d,p) **fi**

if (iguales(o4,a,s) **or**
iguales(o1,b,l)) **and**
iguales(o2,a,m)
then añadir (o3,d,p) **fi**





```
if      iguales(o4, a, s) or  
        (iguales(o1,b, l) and  
          iguales(o2,a, m) )  
then    añadir (o3, d, p) fi
```



Hipergrafo implícito definido por un conjunto de reglas

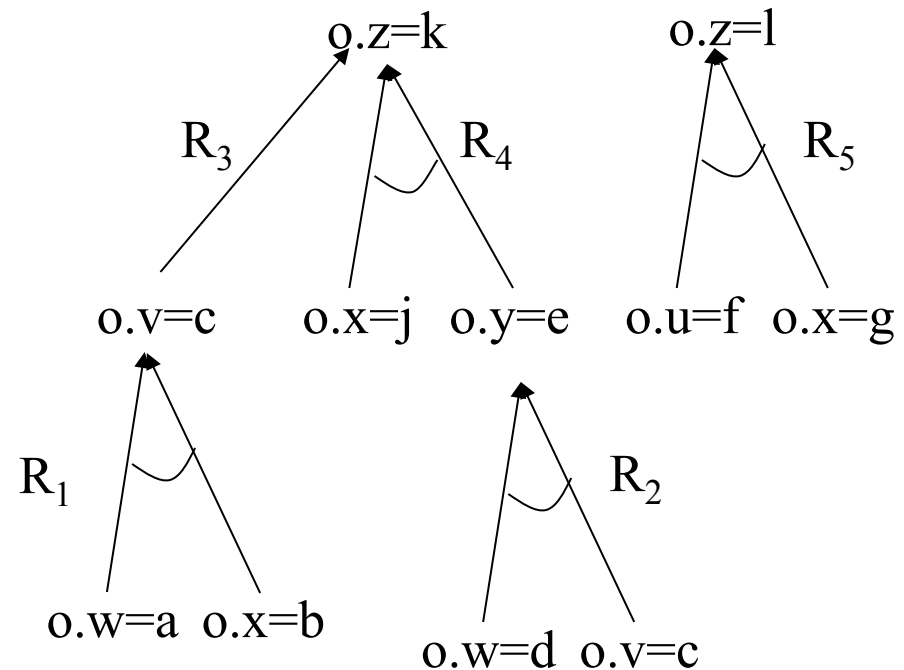
R_1 **if** iguales(o,w,a) **and**
iguales(o,x,b)
then añadir(o,v, c) **fi**

R_2 **if** iguales(o,w,d) **and**
iguales(o,v,c)
then añadir(o,y, e) **fi**

R_3 **if** iguales(o,v, c)
then añadir(o,z, k) **fi**

R_4 **if** iguales(o,x,j) **and**
iguales(o,y,e)
then añadir(o,z, k) **fi**

R_5 **if** iguales(o,u,f) **and**
iguales(o,x,g)
then añadir(o,z, l) **fi**





Ejercicio: obtener el Hipergrafo implícito definido por el siguiente conjunto de reglas

R1 **if** iguales(o,u,a) **and** iguales(o,v,b) **then** añadir(o,w,c) **fi**

R2 **if** iguales(o,x,d) **and** iguales(o,y,e) **then** añadir(o,z,f) **fi**

R3 **if** iguales(o,t,h) **then** añadir(o,s,j) **fi**

R4 **if** iguales(o,w,c) **and** iguales(o,y,e) **then** añadir(o,z,g) also
añadir(o,x,a) **fi**

R5 **if** noiguales(o,t,b) **then** añadir(o,t,h) **fi**



Métodos básicos de inferencia: Encadenamiento hacia adelante

- Interpretación Directa de las Reglas
 - $A, B, C \rightarrow K$
 - “si la memoria de trabajo satisface A, B y C, entonces añadir K”
- Planteamiento básico
 - Parte del conjunto de Hechos iniciales, H
 - Genera nuevos hechos disparando reglas
 - Parada: se alcanza la meta o no hay reglas activadas
- Uso
 - Inicialmente se dispone de suficientes datos
 - No hay una meta clara
 - Naturaleza de la tarea



Métodos básicos de inferencia: Encadenamiento hacia atrás

- Interpretación Inversa de las Reglas
 - A, B, C --->K
 - “si queremos obtener la meta K, obtener primero las submetas A, B y C, “
- Planteamiento básico
 - Parte del conjunto de metas, M
 - Intenta disparar reglas que concluyan la meta, convirtiendo sus antecedentes en nuevas submetas
 - Parada: se alcanza la meta o no hay reglas activadas
- Uso
 - Inicialmente se dispone de pocos datos
 - Es razonable plantear una meta
 - Naturaleza de la tarea



Control de la búsqueda

- Dirección de Búsqueda
 - Encadenamiento hacia adelante: de los hechos a las metas
 - Encadenamiento hacia atrás: de las metas a los hechos
- Régimen Tentativo/Irrevocable
 - Típicamente, adelante irrevocable, atrás tentativo
- Primero Anchura/Profundidad
 - Atrás, generalmente profundidad (*backtracking*)
- Estrategias de Resolución de Conflictos
- Orden Evaluación Premisas (operacional)



5. Encadenamiento hacia adelante



Encadenamiento hacia adelante

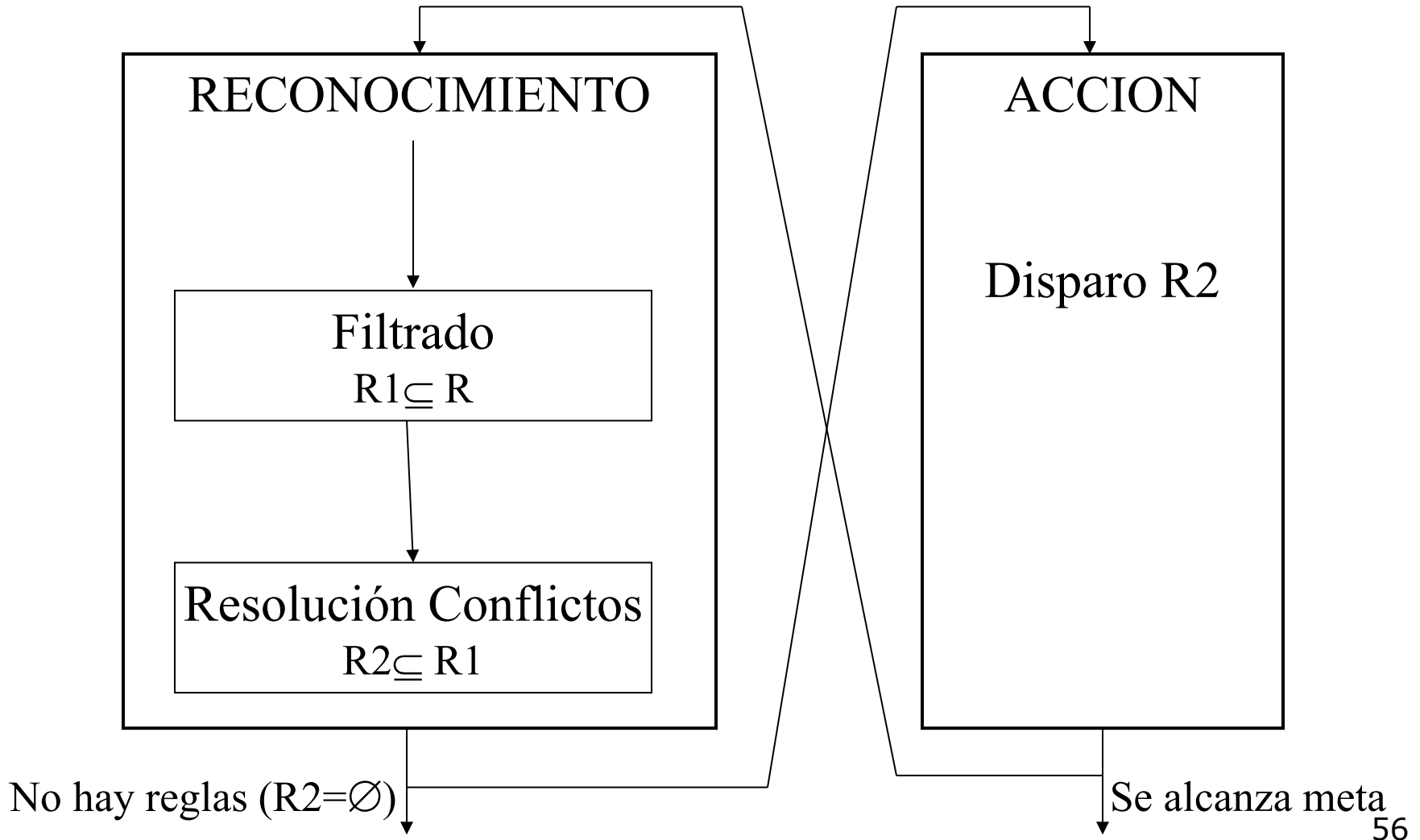
- Interpretación Directa de las Reglas:
A, B, C --->K
"si la memoria de trabajo satisface A, B y C, entonces añadir K"
- Planteamiento básico:
 - Parte del conjunto de Hechos iniciales, H.
 - Genera nuevos hechos disparando reglas.
 - Parada: se alcanza la meta o no hay reglas activadas.
- Uso:
 - Inicialmente se dispone de suficientes datos.
 - No hay una meta clara.
 - Naturaleza de la tarea.



Modelo básico encadenamiento hacia adelante

- Régimen de control irrevocable:
 - Algoritmos de búsqueda simples, que no generan explícitamente grafos de búsqueda.
- Funcionamiento del motor de inferencias: iterar sobre el ciclo básico **reconocimiento-acción**.
- Control adicional:
 - Estrategias de resolución de conflictos.
 - Orden de las premisas.

Ciclo básico Reconocimiento-Acción





Ciclo básico y encadenamiento hacia adelante

- Filtrado o equiparación (*symbolic pattern matching*)
 - Seleccionar todas las reglas activadas: reglas cuyo antecedente se satisfacen según H.
 - R1: conjunto conflicto.
- Resolución de Conflictos:
 - Seleccionar mediante criterios adicionales (estrategias de resolución de conflictos) un subconjunto R2 de R1.
- Acción:
 - Disparar las reglas de R2, realizando su acción.



Estrategias de resolución de conflictos

- Criterios adicionales para decidir que regla(s) de las activadas se va(n) a disparar.
- Es habitual aplicar, secuencialmente, varios criterios
 - Hasta única regla (salvo estrategia todas).



Algunas estrategias de Resolución de Conflictos

- Refracción
 - Cada regla sólo se puede disparar una vez con los mismos elementos de la memoria de trabajo. (desencadenar si hacia atrás)
- Reciencia
 - Seleccionar la regla que se satisfaga con los hechos más recientemente añadidos a la memoria de trabajo.
- Especificidad
 - Seleccionar la regla que contenga más premisas.
- Prioridad
 - Seleccionar la regla con máxima prioridad, fijada por el programador.
- Orden
 - Primera regla, según orden en la base.
- Todas
 - Disparar todas las reglas activadas



Comentarios sobre las estrategias de resolución de conflictos

- Principio de refracción:
 - Esta estrategia es necesaria para evitar ciclos.
- Prioridad:
 - Evitar utilizar demasiados niveles de prioridad.
 - Se suele utilizar para agrupar reglas por tareas.
- Orden:
 - Más bien un criterio para obtener un comportamiento determinista.
- El disparo de reglas debe ser oportunista y no estar prefijado por el programador
 - Oportunista: en función del contenido de la memoria de trabajo.



Un algoritmo básico de encadenamiento hacia adelante

Procedimiento HaciaAdelante(Meta)

```
globales H, BR;                                /* H hechos, BR base de reglas */
locales CC, R;                                  /* CC conjunto conflicto */
CC=Equiparar(Antecedente(BR), H);
R=Resolver(CC);
mientras NoContenida(Meta, H) y NoVacio(R) hacer
    Aplicar(R);
    CC=Equiparar(Antecedente(BR), H);
    R=Resolver(CC);
fin mientras
si Contenida(Meta, H) entonces
    devolver («éxito»);
fin si
end
```



Ejemplo I encadenamiento hacia adelante

$DD = \{o.x^m, o.y^m, o.z_g^s, o.u^m, o.v^m, o.w^m\}$

$BR = \{R_1, R_2, R_3, R_4, R_5\}$

$H = \{o.w = \{a\}, o.x = \{b, j, g\}, o.u = \{f\}\}$

Estrategias de Resolución de Conflictos: refracción, orden.

R_1	if iguales(o,w,a) and iguales(o,x,b) then añadir(o,v,c) fi
R_2	if iguales(o,w,d) and iguales(o,v,c) then añadir(o,y,e) fi
R_3	if iguales(o,v,c) then añadir(o,z,k) fi
R_4	if iguales(o,x,j) and iguales(o,y,e) then añadir(o,z,k) fi
R_5	if iguales(o,u,f) and iguales(o,x,g) then añadir(o,z,l) fi



En términos del ciclo básico

$H = \{o.w=\{a\}, o.x=\{b,j,g\}, o.u=\{f\}\}$

Estrategias de Resolución de Conflictos: refracción, orden.

R_1 **if** iguales(o,w,a) **and** iguales(o,x,b) **then** añadir(o,v,c) **fi**

R_2 **if** iguales(o,w,d) **and** iguales(o,v,c) **then** añadir(o,y,e) **fi**

R_3 **if** iguales(o,v,c) **then** añadir(o,z,k) **fi**

R_4 **if** iguales(o,x,j) **and** iguales(o,y,e) **then** añadir(o,z,k) **fi**

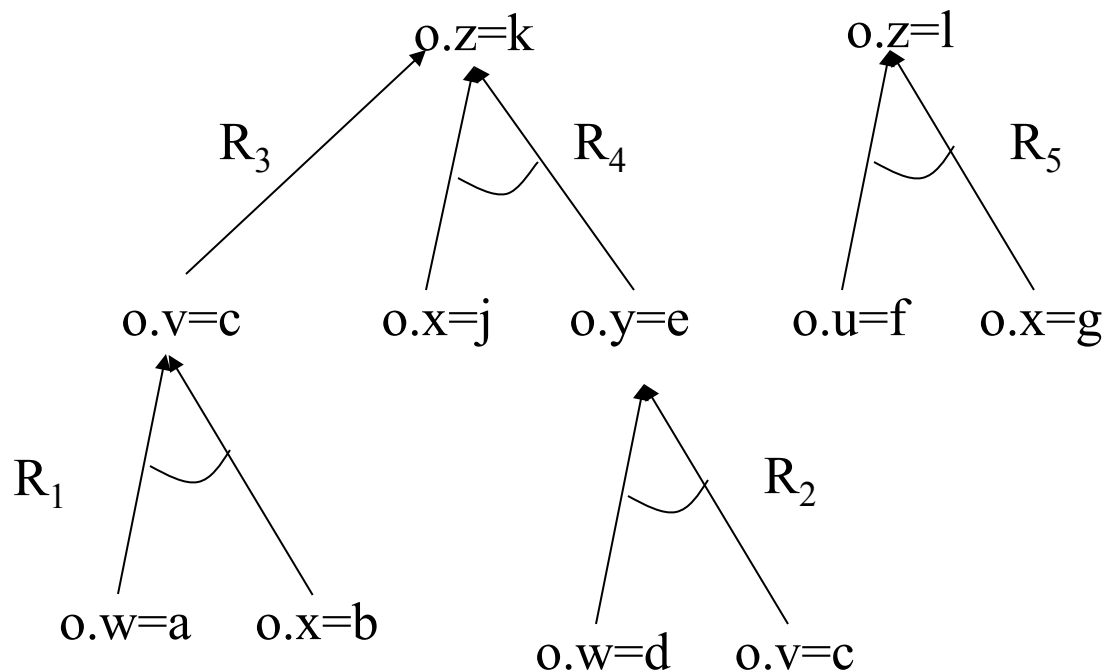
R_5 **if** iguales(o,u,f) **and** iguales(o,x,g) **then** añadir(o,z,l) **fi**

Resolución Conflictos: refracción, orden

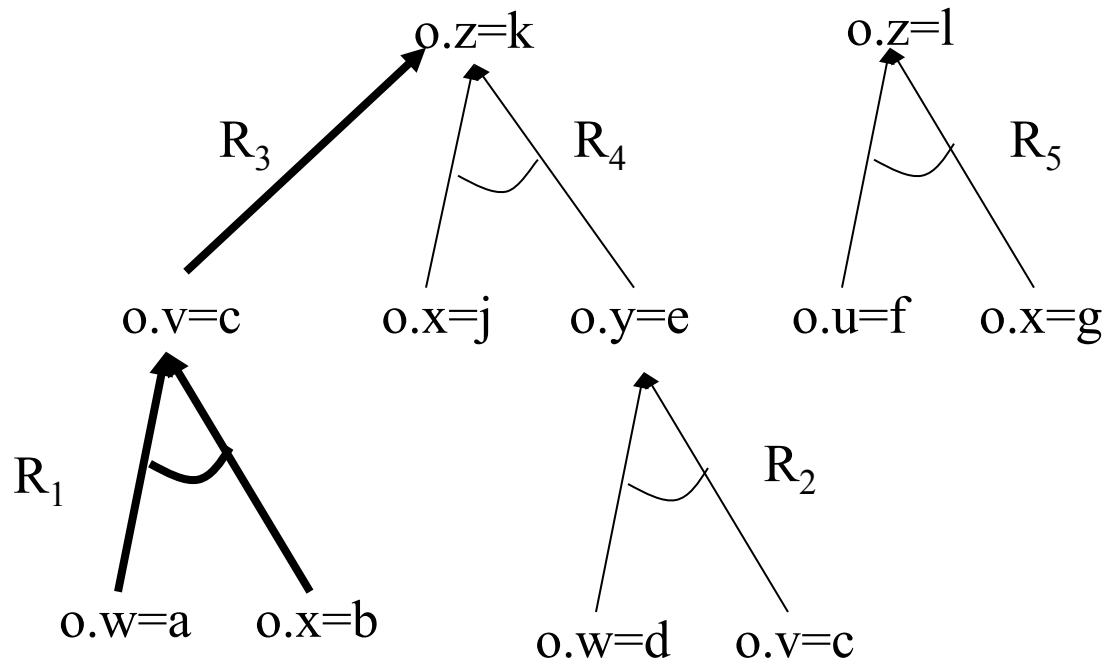
Iter.	Conjunto Conflictos	Res. Conflictos	Mod. Mem. Trabajo
1	R_1, R_5	R_1	+ o.v={c}
2	R_1, R_3, R_5	R_3	+ o.z=k

Repetir el ejercicio, suponiendo que no se declara la meta

En términos del hipergrafo implícito



Hipergrafo explícito ejemplo I



$$H = \{o.z=k, o.v=\{c\}, o.w=\{a\}, o.x=\{b, j, g\}, o.u=\{f\}\}$$

Orden de disparo: R_1, R_3



Ejercicio encadenamiento hacia adelante

$DD = \{o.x^m, o.y^m, o.z_g^s, o.u^m, o.v^m, o.w^m\}$

$BR = \{R_1, R_2, R_3, R_4, R_5\}$

$H = \{o.w = \{a\}, o.x = \{b, j, g\}, o.u = \{f\}\}$

Estrategias de Resolución de Conflictos: refracción, especificidad y orden.

R_1 **if** iguales(o, w, a) **and** iguales(o, x, b) **then** añadir(o, v, c) **fi**

R_2 **if** iguales(o, w, d) **and** iguales(o, v, c) **then** añadir(o, y, e) **fi**

R_3 **if** iguales(o, v, c) **then** añadir(o, z, k) **fi**

R_4 **if** iguales(o, x, j) **and** iguales(o, y, e) **then** añadir(o, z, k) **fi**

R_5 **if** iguales(o, u, f) **and** iguales(o, x, g) **then** añadir(o, z, l) **fi**

1. En términos del ciclo básico
2. En términos del hipergrafo implícito



6. Encadenamiento hacia atrás



Encadenamiento hacia atrás

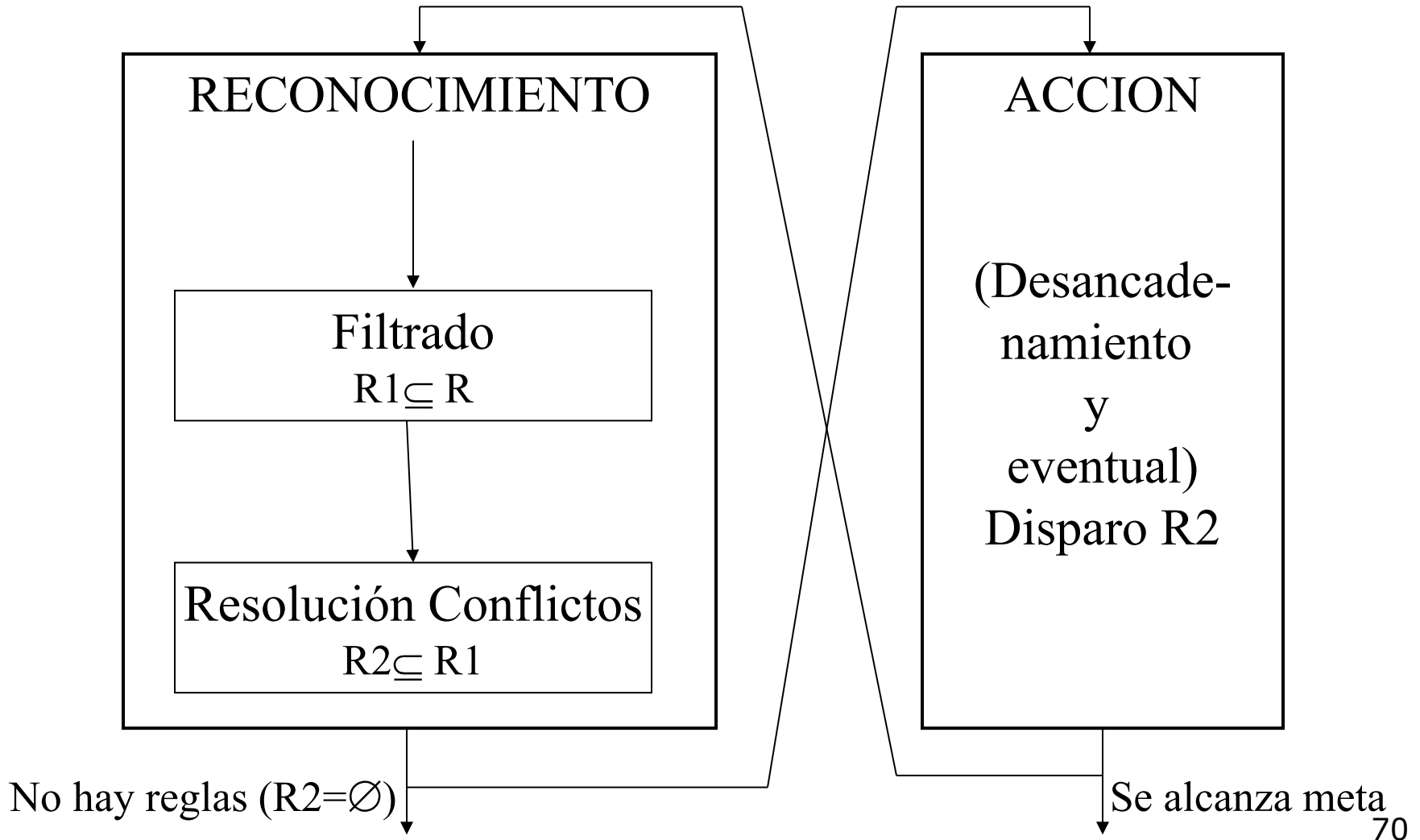
- Interpretación Inversa de las Reglas:
A, B, C --->K
"si queremos obtener la meta K, obtener primero las submetas A, B y C, "
 - Planteamiento básico:
 - Parte del conjunto de metas, M.
 - Intenta disparar reglas que concluyan la meta, convirtiendo sus antecedentes en nuevas submetas.
 - Parada: se alcanza la meta o no hay reglas activadas.
- Uso:
 - Inicialmente se dispone de pocos datos.
 - Es razonable plantear una meta.
 - Naturaleza de la tarea.



Modelo básico encadenamiento hacia atrás

- Régimen de control tentativo:
 - Es necesario explorar el grafo implícito.
 - Algoritmo básico: *backtracking*.
- Ciclo básico **reconocimiento-acción**: menor utilidad, pues no refleja los elementos de la búsqueda.
- Control adicional:
 - Estrategias de resolución de conflictos: menos críticas.
 - Orden de las premisas: mayor influencia.

Ciclo básico Reconocimiento-Acción





Ciclo básico y encadenamiento hacia atrás

- Filtrado o equiparación:
 - Selecciona las reglas cuyo consecuente permite obtener la meta actual.
 - Estas reglas no tienen que estar activadas.
 - R1: conjunto conflicto.
- Resolución de Conflictos:
 - Seleccionar mediante criterios adicionales (estrategias de resolución de conflictos) un subconjunto R2 de R1.
- Acción
 - Desencadenar reglas: reemplazar meta actual por submetas, obtenidas de las condiciones de las reglas.
 - Disparar las reglas cuyo antecedente se satisface, realizando su acción.



Un algoritmo básico de encadenamiento hacia atrás (I)

Procedimiento HaciaAtras(Meta)

si Inferir(Meta) **entonces**

devolver(«éxito»);

sino

devolver(«fracaso»);

fin si

end



Un algoritmo básico de encadenamiento hacia atrás (II)

Procedimiento Inferir(Meta)

```
globales H, BR; /* H hechos , BR Base de reglas */
locales Verificado; /* T cuando una regla concluye la meta */
si Contenida(Meta, H) entonces
    Si Satisface(Meta, H) entonces devolver(T)
    sino devolver(F)
fin si.
fin si
Verificado=F;
CC=Equiparar(Consecuentes(BR), Meta);
mientras NoVacio(CC) y No(Verificado) hacer
    R=Resolver(CC); Eliminar(R, CC);
    Verificado=Desencadenar(R);
fin mientras
devolver(Verificado)
end
```



Un algoritmo básico de encadenamiento hacia atrás (III)

Procedimiento Desencadenar(Regla)

si EvaluarAntecedente(Regla) **entonces**

 Aplicar(Regla);

devolver(T);

si no

devolver(F);

fin si

end



Un algoritmo básico de encadenamiento hacia atrás (IV)

Procedimiento EvaluarAntecedente(Regla)

locales Condicion, NuevaMeta;

para cada Condicion **en** Antecedente(Regla) **hacer**

 NuevaMeta=ExtraerMeta(Condicion);

 Inferir(NuevaMeta);

si No(EvaluarPredicado(Condicion)) **entonces**

devolver(F);

fin si

fin para

devolver (T)

end



Comentarios algoritmo básico

- Algoritmo muy simple.
- Ilustra encadenamiento hacia atrás mediante un algoritmo de *bactraking* con reglas proposicionales.
- Se pueden añadir numerosas mejoras:
 - Estrategia de resolución de conflicto “todas”.
 - No solo reglas conjuntivas.
 - Otros métodos de resolver metas:
 - Preguntar al usuario.
 - Consulta base de datos.
 - Ejecución procedimiento.



Ejemplo II encadenamiento hacia atrás

$DD = \{o.x^m, o.y^m, o.z_g^s, o.u^m, o.v^m, o.w^m\}$

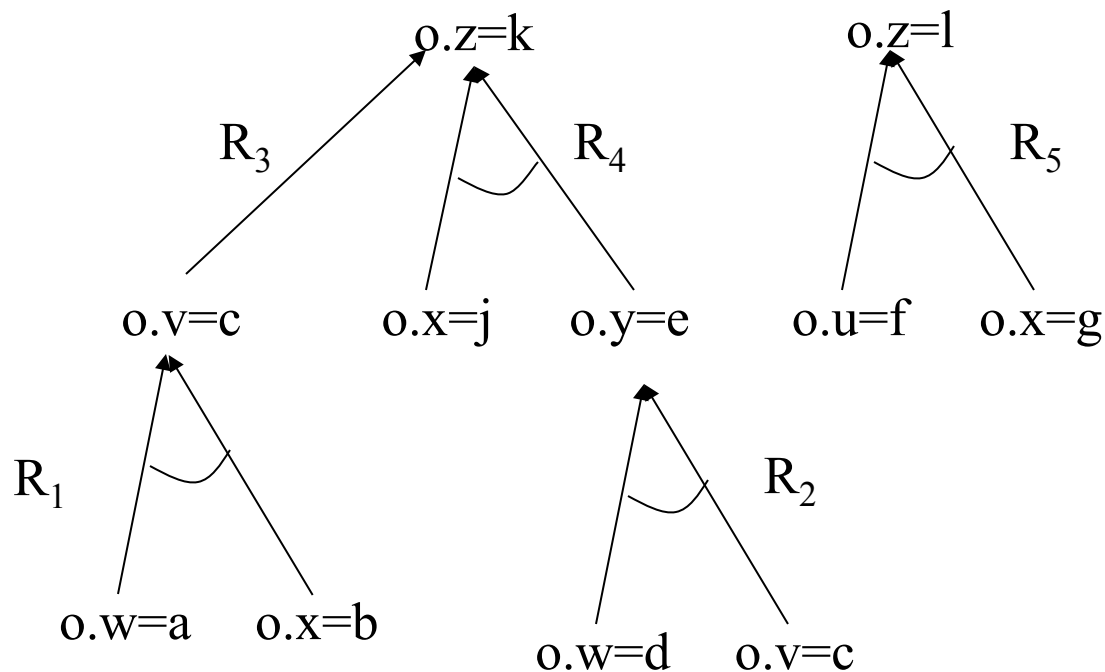
$BR = \{R_1, R_2, R_3, R_4, R_5\}$

$H = \{o.w = \{a\}, o.x = \{b, j, g\}, o.u = \{f\}\}$

Estrategias de Resolución de Conflictos: refracción, orden.

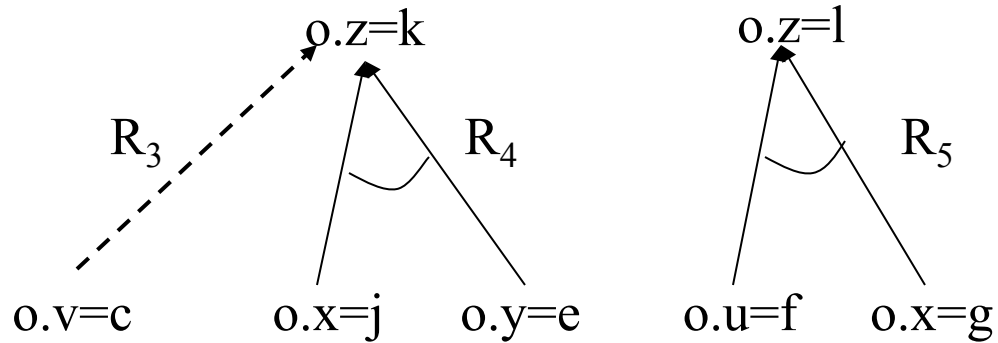
R_1	if iguales(o,w,a) and iguales(o,x,b) then añadir(o,v,c) fi
R_2	if iguales(o,w,d) and iguales(o,v,c) then añadir(o,y,e) fi
R_3	if iguales(o,v,c) then añadir(o,z,k) fi
R_4	if iguales(o,x,j) and iguales(o,y,e) then añadir(o,z,k) fi
R_5	if iguales(o,u,f) and iguales(o,x,g) then añadir(o,z,l) fi

En términos del hipergrafo implícito



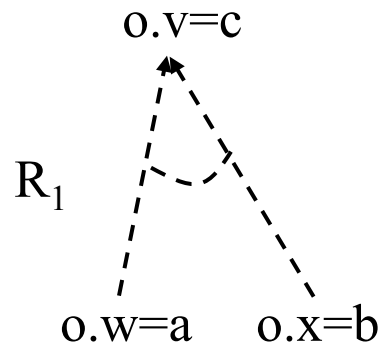
Primer ciclo

- Meta: $o.z$
- $CC = \{R_3, R_4, R_5\}$
- Resolución de conflictos
 - $R = R_3$



Segundo ciclo

- Meta: $o.v$
- $CC = \{R_1\}$
- Resolución de conflictos
 - $R = R_1$





Tercer ciclo

- Meta: $o.w$
- Se satisfice: $o.w=a \in H$

$$o.w=a$$



Cuarto ciclo

- Meta: $o.x$
- Se satisface: $o.x=b \in H$

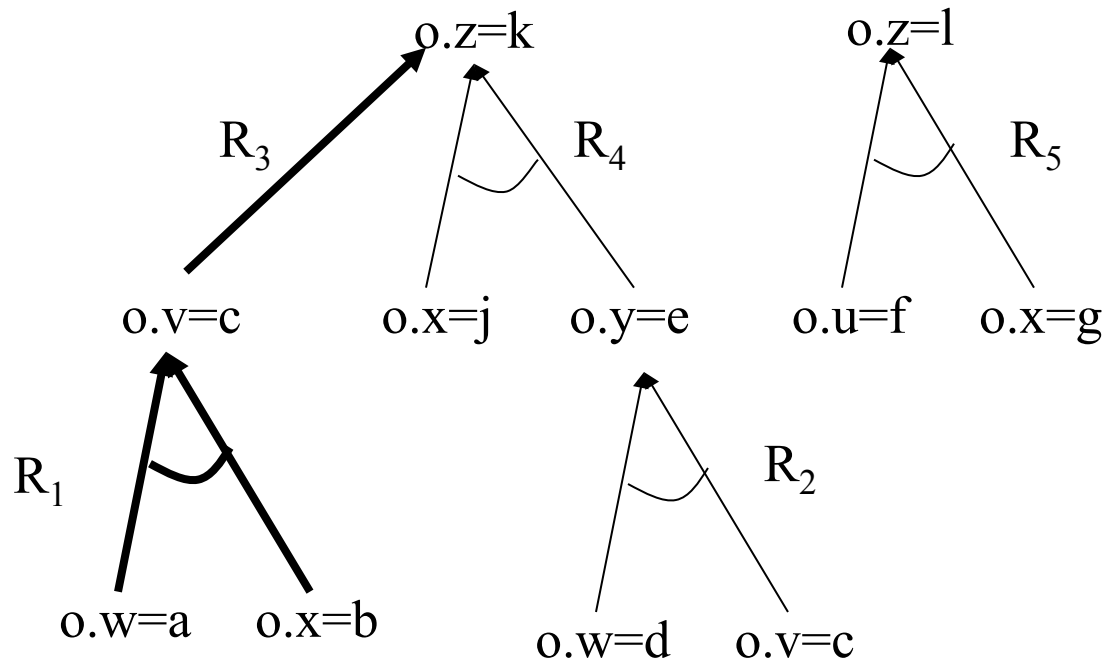
$$o.x=b$$



Disparo de reglas

- Tras el cuarto ciclo, el procedimiento *Desencadenar* dispara las reglas
- Primera, R_1
 - $H = \{o.v = \{c\}, o.w = \{a\}, o.x = \{b, j, g\}, o.u = \{f\}\}$
- Segunda, R_3
 - $H = \{o.z = k, o.v = \{c\}, o.w = \{a\}, o.x = \{b, j, g\}, o.u = \{f\}\}$

Hipergrafo explícito ejemplo II



$$H = \{o.z=k, o.v=\{c\}, o.w=\{a\}, o.x=\{b, j, g\}, o.u=\{f\}\}$$

Orden de disparo: R_1, R_3



Ejercicio encadenamiento hacia atrás

$DD = \{o.x^m, o.y^m, o.z_g^s, o.u^m, o.v^m, o.w^m\}$

$BR = \{R_1, R_2, R_3, R_4, R_5\}$

$H = \{o.w = \{a\}, o.x = \{j, g\}, o.u = \{f\}\}$

Estrategias de Resolución de Conflictos: refracción, orden.

R_1 **if** iguales(o,w,a) **and** iguales(o,x,b) **then** añadir(o,v,c) **fi**
 R_2 **if** iguales(o,w,d) **and** iguales(o,v,c) **then** añadir(o,y,e) **fi**
 R_3 **if** iguales(o,v,c) **then** añadir(o,z,k) **fi**
 R_4 **if** iguales(o,x,j) **and** iguales(o,y,e) **then** añadir(o,z,k) **fi**
 R_5 **if** iguales(o,u,f) **and** iguales(o,x,g) **then** añadir(o,z,l) **fi**



7. Lenguajes con variables



Lenguajes con variables

- Las variables extienden el poder de representación de los lenguajes de reglas.
- Las reglas con variables permiten representar conocimiento general.
- Semántica de la variables: similar a LPO.
- Lenguajes:
 - Extensión formalismo Objeto-Atributo-Valor.
 - Patrones simbólicos.



Lenguaje de patrones simbólicos

- Patrón simbólico: secuencia de constantes y/o variables
- Hechos: patrones de constantes.
 - (Juan edad 50 peso 71)
- Reglas: admiten patrones con variables.
 - Ejemplo patrón de regla: (Juan edad 50 peso ?y).
- No limitado a la estructura O-A-V.
 - Que sigue siendo la conceptualización recomendable.



Sintaxis de patrones y hechos

$\langle \text{patrón} \rangle ::= (\langle \text{término} \rangle \{ \text{término} \}^*)$

$\langle \text{término} \rangle ::= \langle \text{constante} \rangle \mid \langle \text{variableUnivaluada} \rangle \mid$
 $\langle \text{variableMudaUnivaluada} \rangle$

$\langle \text{variableMudaUnivaluada} \rangle ::= ?$

Constante: cualquier palabra que no empiece por ?

VariableUnivaluada: palabra que empieza por ?

$\langle \text{hecho} \rangle ::= (\langle \text{constante} \rangle \{ \langle \text{constante} \rangle \}^*)$



Ejemplos patrones, hechos

- Patrones
 - (edad paciente 60)
 - (edad paciente ?x)
 - (edad paciente ?)
 - (light l1 t)
 - (l1 ligh t OK t)
 - (s1 estado ?x)
- Hechos
 - (edad paciente 60)
 - (light l1 t)
 - (l1 ligh t OK t)



Ligadura de variables

- Las variables de los patrones se pueden reemplazar por constantes.
- Se definen:
 - Ligadura: constante, denotada por d , que reemplaza a una variable.
 - Variable ligada: variable para la que existe una ligadura, denotado por $?x=d$.
 - Substitución: reemplazar una variable por su ligadura, en su alcance.
 - Alcance de una ligadura:
 - variable muda: la ocurrencia de la variable.
 - variable no muda: regla en la que ocurre el patrón.

Equiparación de Patrones

- Equiparación, confrontación o *Pattern Matching*

Sea p un patrón y h un hecho

Se dice que el patrón p y el hecho h se equiparan (confrontan) si existen ligaduras para las variables que ocurren en p tales que al sustituir las variables por sus ligaduras, p y h son sintácticamente iguales

$p : (c \text{ ?}x \text{ ?}y \ a)$	$h : (c \ a \ b \ a)$	con $?x=a, ?y=b$, confrontan
$p : (c \text{ ?}x \text{ ?}x \ a)$	$h : (c \ a \ b \ a)$	no confrontan
$p : (c \text{ ?} \text{ ?} \ a)$	$h : (c \ a \ b \ a)$	confrontan
$p : (c \text{ ?}x \text{ ?} \ a)$	$h : (c \ a \ b \ a)$	con $?x=a$, confrontan

Patrones y reglas de producción

- Los patrones , con o sin variables, pueden formar parte de las reglas de producción.

- Sintaxis reglas:

<reglaProducción>	::= if <antecedente> then <consecuente> fi
<antecedente>	::= <disyunción> { and <disyunción>}*
<disyunción>	::= <condición> { or <condición>}*
<consecuente>	::= <conclusión> { also <conclusión>}*
<condición>	::= <predicado> <patrón>
<conclusión>	::= <acción> <patrón>
<predicados>	::= iguales noiguales mayorque ...
<acción>	::= añadir eliminar ...





Ejemplo reglas

if **iguales**(paciente-1 síntoma dolorAbdominal) **and**
 iguales(paciente-1 evidencia rumorAbdominal) **and**
 iguales(paciente-1 evidencia masaPulsante)
then
 añadir(paciente-1 enfermedad aneurísmaAortico)
Fi

if **iguales**(light ?objeto t) **and**
 iguales(live ?objeto t) **and**
 iguales(ok ?objeto t)
then
 añadir(lit ?objeto t)
fi



Semántica de los predicados

<i>Predicado</i>	<i>Cierto si</i>
iguales <patrón>	<patrón> confronta con algún hecho en MT
noiguales <patrón>	<patrón> no confronta con ningún hecho de la memoria de trabajo
mayorque <patrón>	Todas las variables están ligadas y al sustituir todas las variables se obtiene una secuencia de números estrictamente decreciente
menorque <patrón>	Todas las variables están ligadas y al sustituir todas las variables se obtiene una secuencia de números estrictamente creciente



Semántica de las acciones

<i>Acción</i>	<i>Efecto</i>
añadir <patrón>	Si todas las variables de <patrón> están ligadas, substituir todas las variables y añadir hecho resultante a la memoria de trabajo
eliminar <patrón>	Si todas las variables de <patrón> están ligadas, substituir todas las variables y eliminar hecho resultante a la memoria de trabajo



Ejemplo regla y disparo

$H = \{ \text{(persona nombre Juan edad 16)} \}$

if iguales(persona nombre ?x edad ?y) **and**
 mayorque(?y 12) **and**
 menorque(?y 18)

then

 añadir(?x es un adolescente)

fi

tras disparar la regla

$H = \{ \text{(persona nombre Juan edad 16), (Juan es un adolescente)} \}$



Particularización de Regla

Sea H el conjunto de hechos de la MT, R una regla y $M \subset H$ un conjunto minimal de hechos con los que se satisface el antecedente de R :

- el antecedente de R se satisfacen con los hechos de M .
 - el antecedente de R no se satisface con los hechos de $M - \{h_i / h_i \in M\}$.
-
- Se denomina particularización de R al par (M, R) .
 - Se denomina regla particularizada R' , a la regla que se obtiene a partir de R al sustituir las variables por las ligaduras que hacen que los patrones de R confronte con los hechos de M .



Ejemplo de Regla Particularizada

H: {(persona nombre Juan edad 16), (persona nombre Luis edad 17)}

M: {(persona nombre Juan edad 16)}

R: **if** iguales(persona nombre ?x edad ?y) **and**
 mayorque(?y 12) **and**
 menorque(?y 18)
 then
 añadir(?x es un adolescente)
 fi

R' : **if** iguales(persona nombre Juan edad 16) **and**
 mayorque(16 12) **and**
 menorque(16 18)
 then
 añadir(Juan es un adolescente)
 fi



Múltiples particularizaciones

En general, el conjunto M no es único y existe una particularización por cada posible conjunto M.

H: {(persona nombre Juan edad 16), (persona nombre Luis edad 17)}

M1: {(persona nombre Luis edad 17)}

```
R1':  if           iguales(persona nombre Luis edad 17) and
      mayorque(17 12) and
      menorque(17 18)
      then
      añadir(Luis es un adolescente)
      fi
```



Variables y ciclo básico, encadenamiento hacia adelante

- Filtración:
 - El conjunto conflicto está formado por todas las posibles particularizaciones de reglas
- Resolución de conflictos:
 - Las estrategias de resolución de conflictos trabajan sobre reglas particularizadas.
 - Refracción: no disparar dos veces la misma particularización de regla.
- Disparo:
 - Realizar la acción de reglas particularizadas.



Ejercicio equiparación 1

- (caballo ?x): ?x es un caballo
- (padre ?x ?y): el caballo ?x es el padre de ?y
- (rapido ?x): el caballo ?x es rápido
- $H = \{(\text{caballo Cometa}), (\text{caballo Bronco}), (\text{caballo Veloz}), (\text{caballo Rayo}), (\text{padre Cometa Veloz}), (\text{padre Cometa Bronco}), (\text{padre Veloz Rayo}), (\text{rapido Bronco}), (\text{rapido Rayo})\}$
- **R = if iguales(caballo ?x) and iguales(padre ?x ?y) and iguales(rapido ?y) then añadir(seleccionar ?x) fi**



Determinar

- Ligaduras para los patrones aislados:
 - (caballo ?x)
 - (padre ?x ?y)
 - (rapido ?y)
- Ligaduras para secuencias de patrones, tal y como se encuentran en la regla:
 - **iguales**(caballo ?x) **and** **iguales**(padre ?x ?y)
 - **iguales**(caballo ?x) **and** **iguales**(padre ?x ?y) **and** **iguales**(rapido ?y)
- Reglas particularizadas.



Ejercicio equiparación 2

- $H = \{(\text{caballo Cometa}), (\text{caballo Bronco}), (\text{caballo Veloz}), (\text{caballo Rayo}), (\text{padre Cometa Veloz}), (\text{padre Cometa Bronco}), (\text{padre Veloz Rayo}), (\text{rapido Bronco}), (\text{rapido Rayo})\}$
- $R = \text{if iguales}(\text{caballo ?x}) \text{ and noiguales}(\text{rapido ?x}) \text{ then añadir}(\text{reserva ?x}) \text{ fi}$



Determinar

- Ligaduras para los patrones aislados:
 - (caballo ?x)
 - (rapido ?x)
- Ligaduras para secuencias de patrones, tal y como se encuentran en la regla:
 - **iguales**(caballo ?x) **and noiguales**(rapido ?x)
- Reglas particularizadas.



8. Algoritmo de Rete



Equiparación de reglas y hechos (I)

- Aproximación Naive:
 - En cada ciclo, comparar antecedente de las reglas con hechos en Memoria de Trabajo.
- Comportamiento en el peor caso: $O(RF^P)$
 - R: número total de reglas.
 - F: número total de hechos.
 - P: número medio de patrones por regla.



Equiparación de reglas y hechos (II)

- Motivo de la ineficiencia:
 - El conjunto de reglas es estable.
 - La memoria de trabajo cambia.
 - PERO: (habitualmente) el porcentaje de cambio por ciclo es (muy) pequeño.
- Ineficiencia de la aproximación Naive:
 - La mayoría de los patrones que se satisfacen en un ciclo también lo hacen en el siguiente.
- Alternativa:
 - Algoritmo que recuerde activaciones entre ciclos, actualizando patrones que confronten con los hechos que han cambiado.



Algoritmo de Rete

- Representa las reglas como datos (red Rete).
- El compilador crea la red Rete a partir de las reglas .
- La red Rete se puede asimilar a máquina de estados que consume modificaciones de hechos.
- La red recuerda estados anteriores.

Charles L. Forgy, Artificial Intelligence 19 (1982), 17-37.



Ejemplo

Regla R1

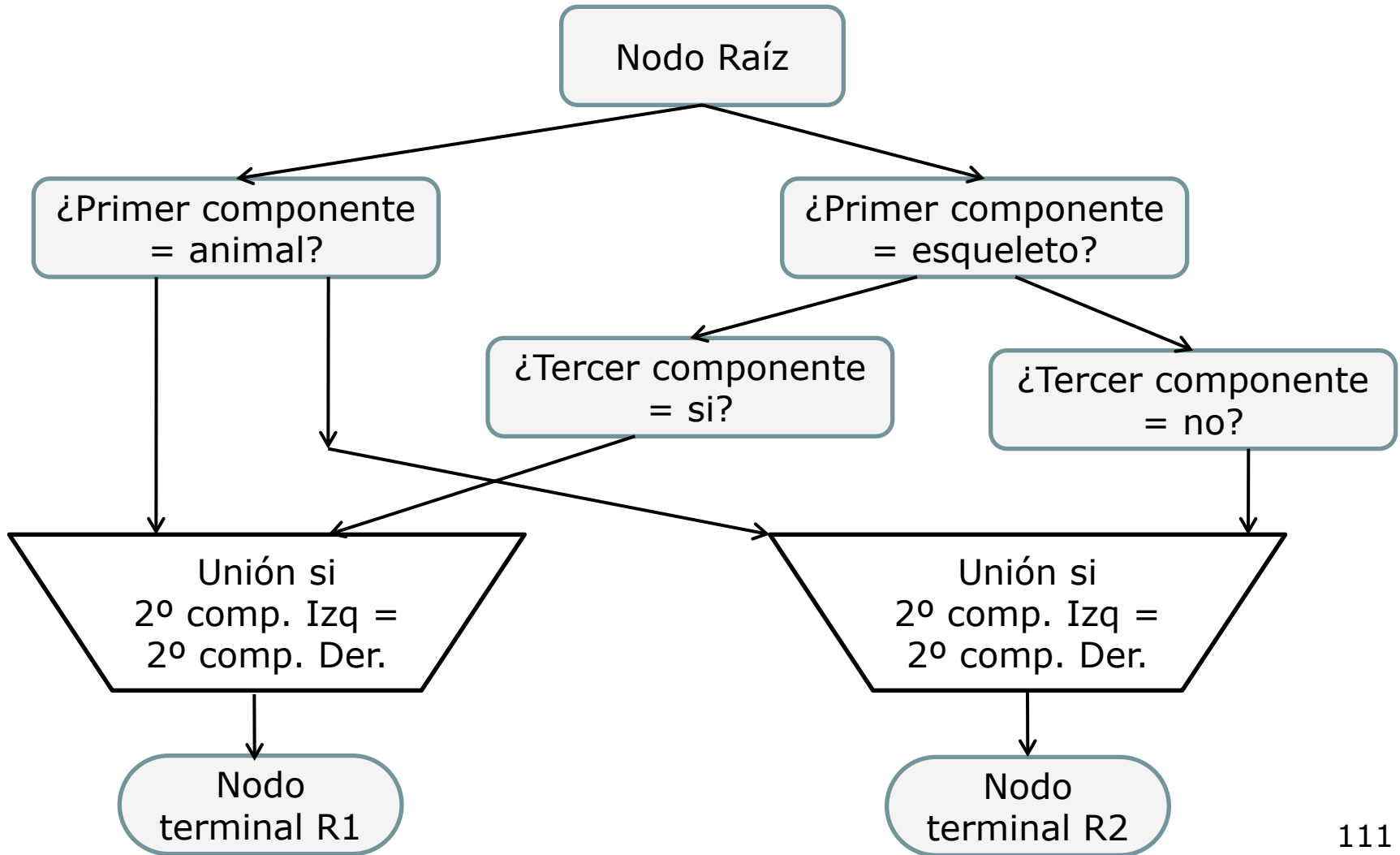
```
if iguales(animal ?a) and  
    iguales(esqueleto ?a si)  
then  
    añadir(vertebrado ?a)  
fi
```

Regla R2

```
if iguales(animal ?a) and  
    iguales(esqueleto ?a no)  
then  
    añadir(invertebrado ?a)  
fi
```

Primero: compilar las reglas y generar la red Rete

Red Rete

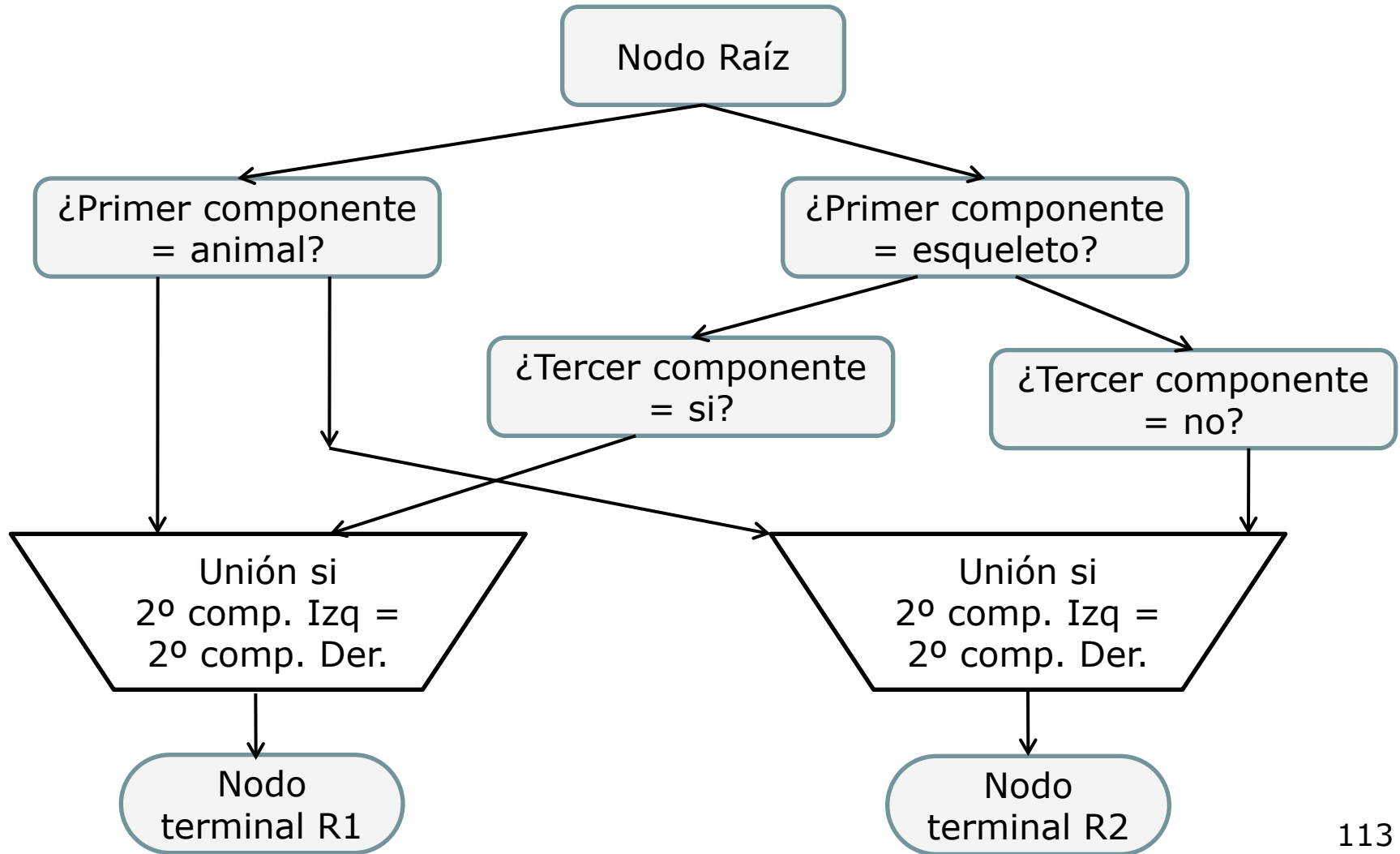




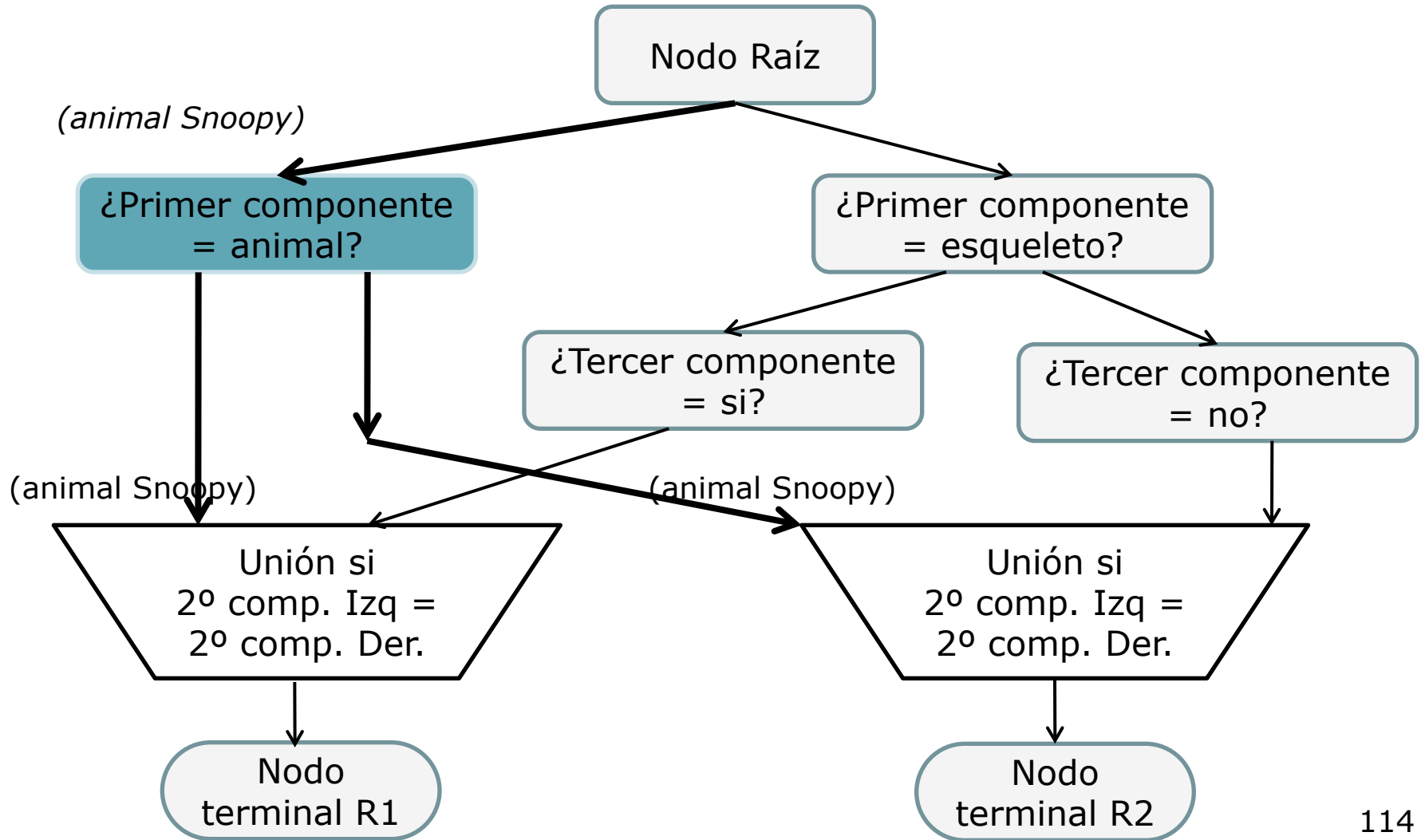
Elementos de la red Rete

- Nodos “patrón”: entrada red
 - Analizan términos constantes de un patrón.
 - ¿Primer componente = animal? comprueba que el primer componente del hecho es “animal”.
 - Los hechos que superan el test se envían a la salida del nodo.
- Nodos “Unión”: representan confrontaciones
 - Comprueban la igualdad de las ligaduras de variables comunes.
 - Dos entradas (y dos memorias)
 - Izquierda: grupos de uno o más hechos que confrontan
 - Derecha: un único hecho
 - Salida: grupos de dos o más hechos que confrontan
- Nodos terminales: representan reglas.
 - Registran activaciones de reglas particularizadas

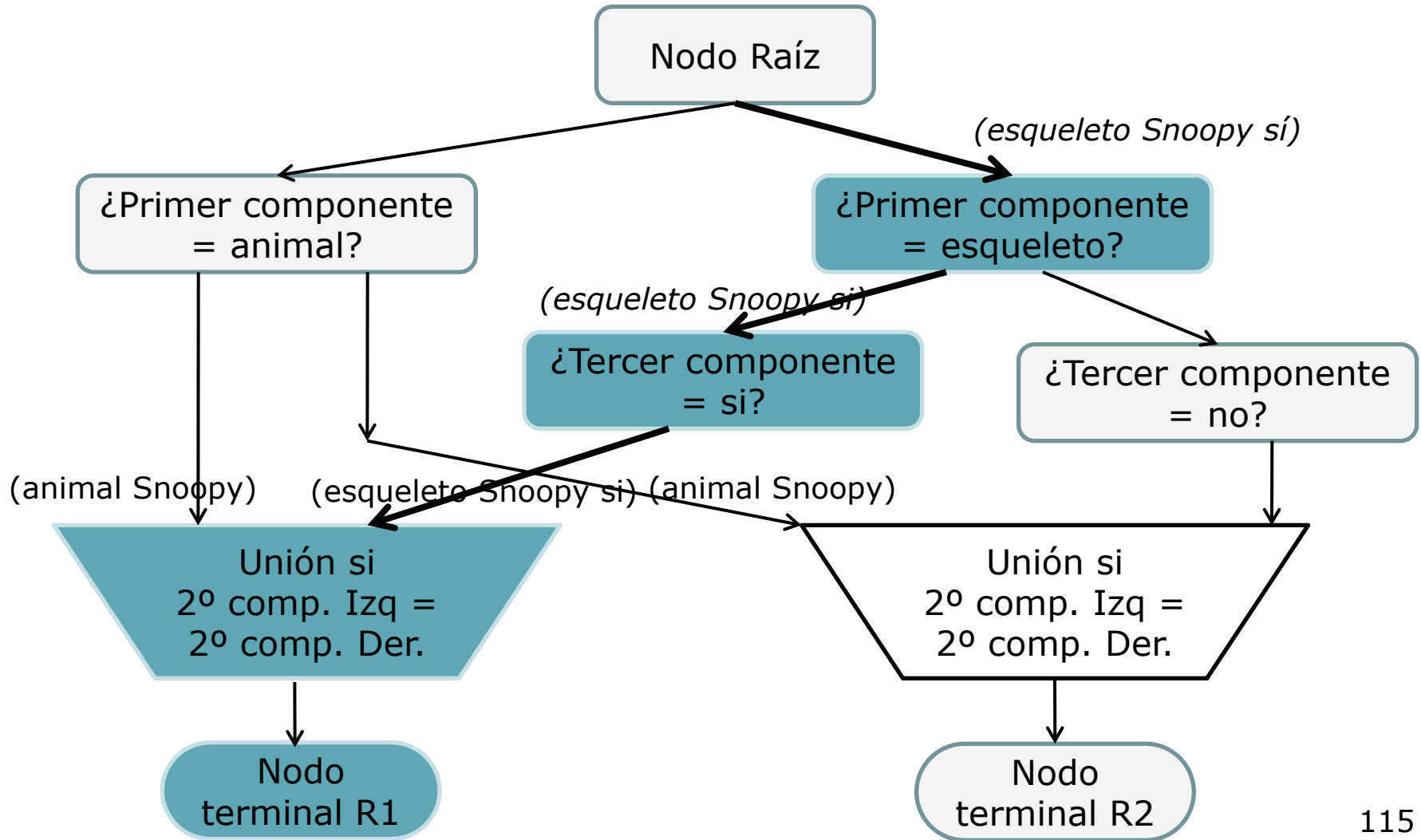
Red Rete. Creación de la red: compilación de la base de reglas.



Procesar nuevo hecho: (animal Snoopy)



Procesar nuevo hecho: (esqueleto Snoopy sí)



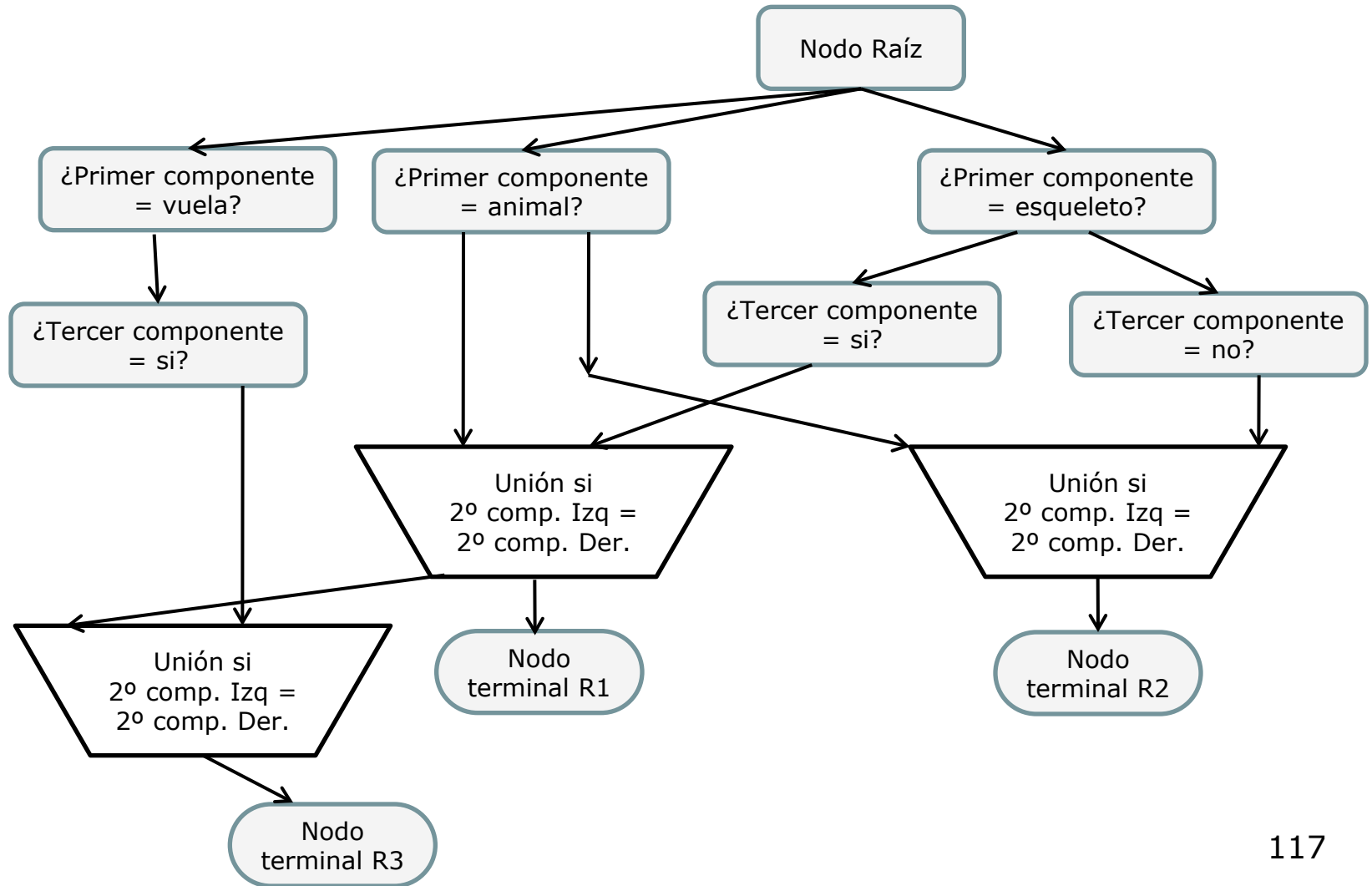


Los nodos “Unión” explotan redundancia de patrones en reglas

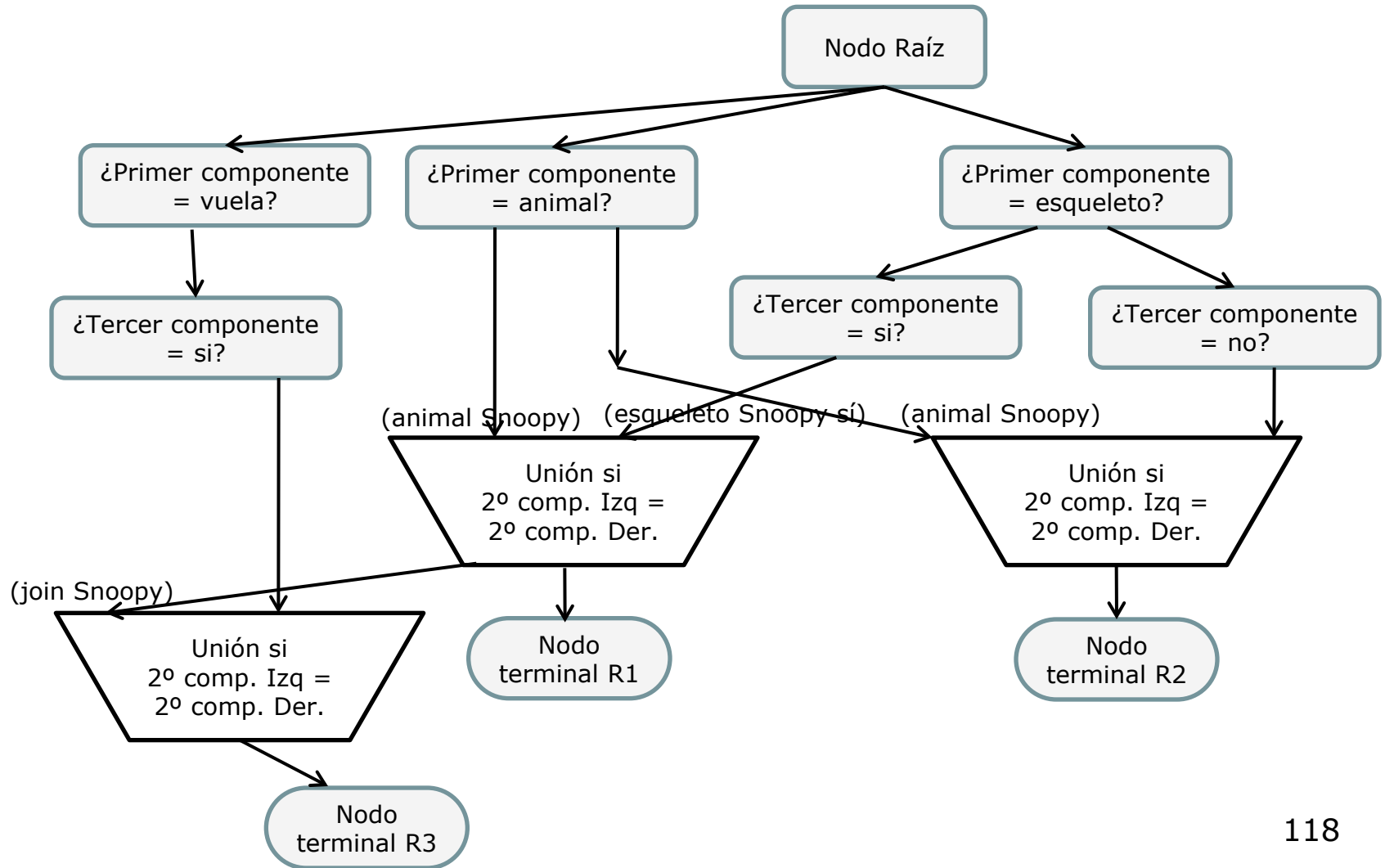
Regla R3

```
if iguales(animal ?a) and  
    iguales(esqueleto ?a si)  
    iguales(vuela ?a si)  
then  
    añadir(ave ?a)  
fi
```

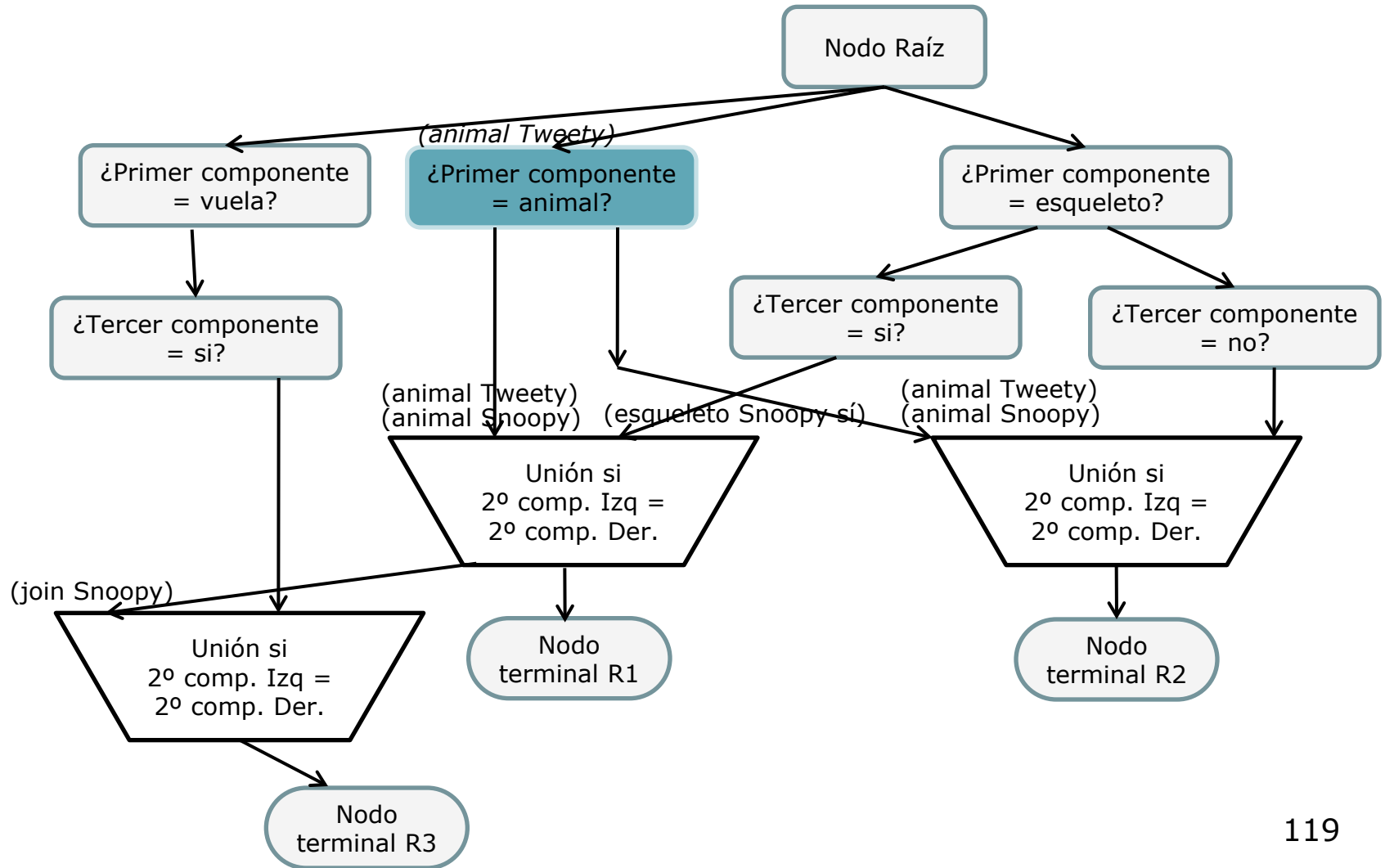
Red Rete



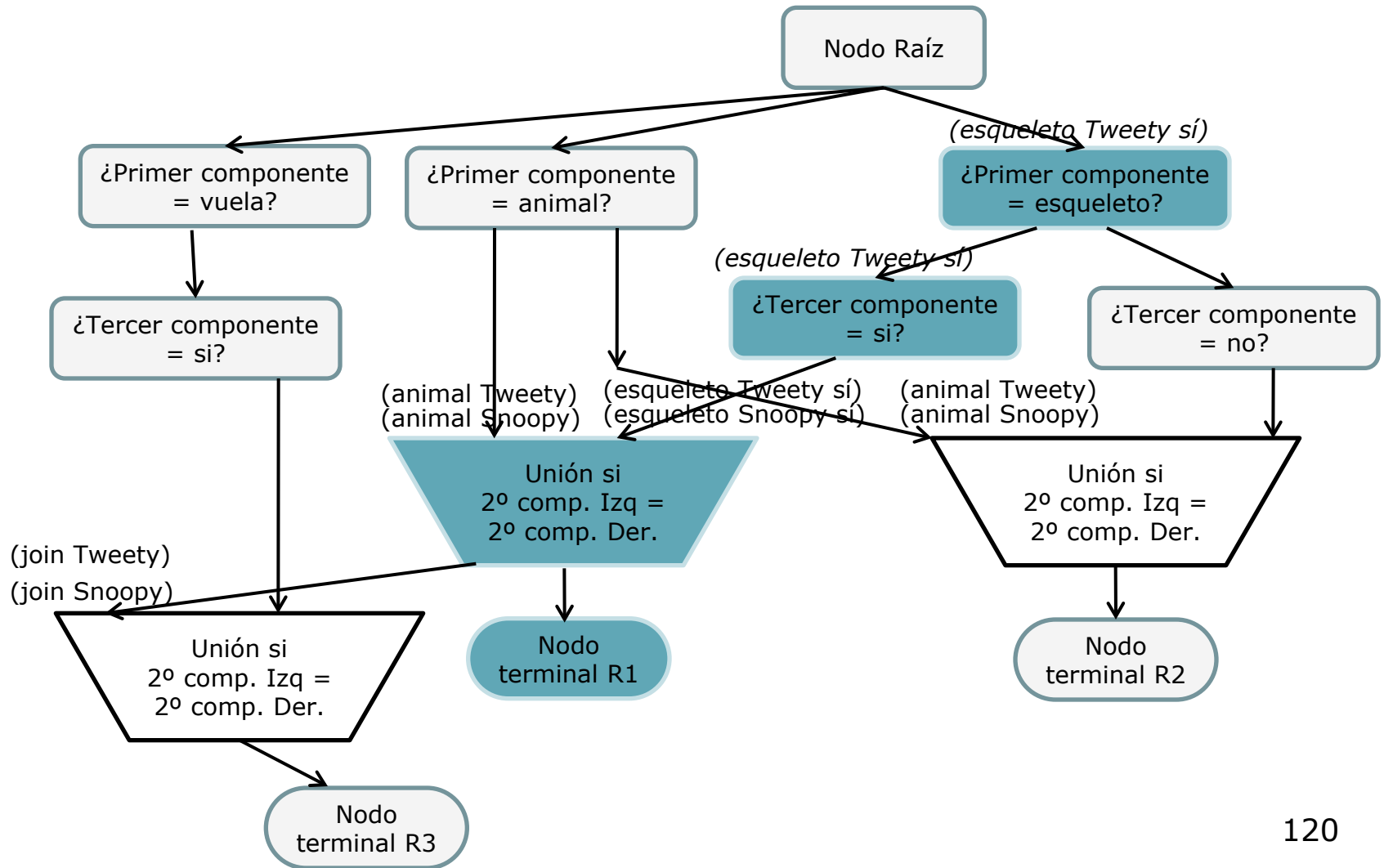
Tras procesar (animal Snoopy), (esqueleto Snoopy sí)



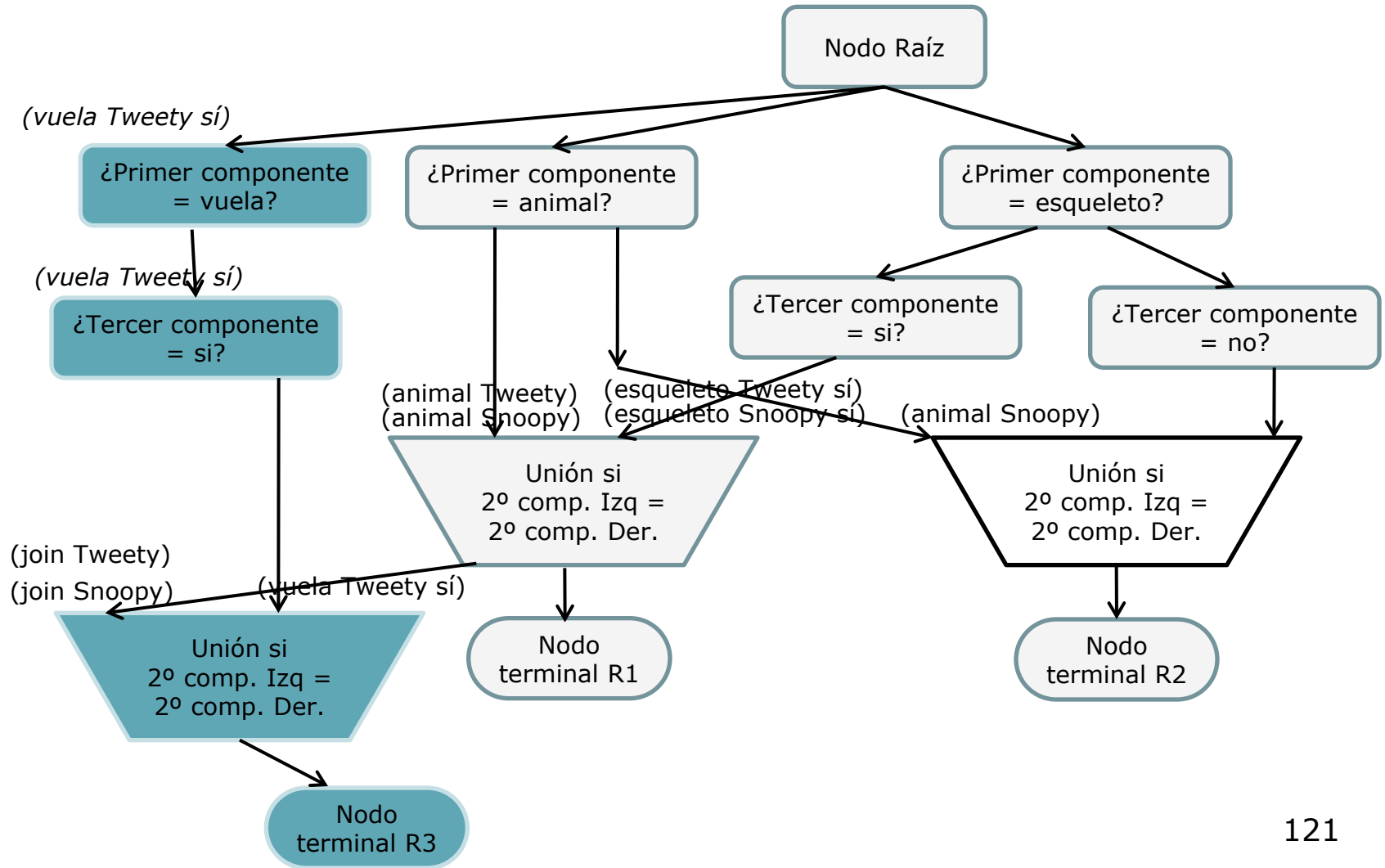
Procesar nuevo hecho (animal Tweety)



Procesar nuevo hecho (esqueleto Tweety sí)



Procesar nuevo hecho (vuela Tweety sí)





Eficiencia algoritmo de RETE

- Primer ciclo
 - Similar aproximación Naive
- Caso medio:
 - Dependencia con la implementación (indexación, estructuras y tamaño memoria)
 - Caso medio Jess: $O(R'F'P')$
 - $R' < R$
 - $F' < \text{número de hechos que cambian en cada iteración}$
 - $1 < P' < P$