

Bloque 1– Representación del conocimiento y búsqueda

Tema 2: Inferencia en SBR: encadenamiento y control. RETE.

Bloque 1, Tema 2- Indice

1. Motor de inferencia
2. Estrategias de resolución de conflictos.
3. Ejemplos.
4. Algoritmo de matching RETE.
5. Ejercicios.
6. Anexo: sintáxis BNF de las reglas en CLIPS

Bibliografía

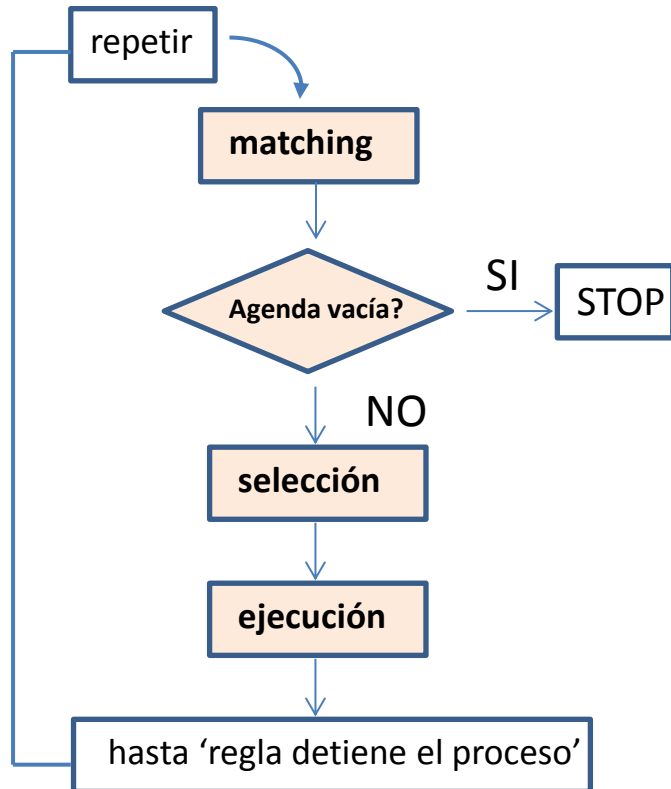
- Capítulo 3: *Sistemas Basados en Reglas*. Inteligencia Artificial. Técnicas, métodos y aplicaciones. McGraw Hill, 2008.
- Charles Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", [*Artificial Intelligence*](#), 19, pp 17–37, 1982.
- CLIPS User's Guide.
- CLIPS Basic Programming Guide.

1. Motor de inferencia

La programación lógica sigue una semántica declarativa. En cambio, un SBR tiene una semántica procedural. Esta diferencia se debe a la naturaleza procedural de la RHS de las reglas en un SBR.

CLIPS utiliza un motor de inferencia hacia delante basado en el algoritmo de matching Rete (motor de inferencia basado en Rete)

El mecanismo de control que siguen los motor de inferencia hacia delante se denomina **ciclo reconocimiento-acción** (también ciclo matching-selección-acción):



Matching de las reglas: genera un Conjunto conflicto (o Agenda) con todas las reglas aplicables o instancias de reglas encontradas

Si la Agenda está vacía, el proceso finaliza. En caso contrario, se selecciona una instancia del Conjunto Conflicto o Agenda de acuerdo a un criterio predeterminado (Estrategia de Resolución de Conflictos).

Se ejecuta la RHS de la instancia seleccionada, actualizando la BH y/o ejecutando las acciones externas

1. Motor de inferencia

BH = base de hechos; BR= base de reglas; CC= conjunto conflicto; InstRule=instancia de regla;

BH=hechos iniciales

CC=Matching(BH,BR)

while objetivo $\not\subseteq$ BH y $CC \neq \emptyset$

InstRule=Seleccionar(CC)

BH=BH \setminus hechos eliminados por InstRule ;; Ejecución de la RHS (1)

BH=BH \cup hechos añadidos por InstRule ;; Ejecución de la RHS (2)

CC=CC \setminus instancias de reglas eliminadas resultado de hechos eliminados (3)

CC=CC \cup Matching(BH,BR) (4)

end_while

if objetivo \subseteq BH ;; disparo de una regla que encuentra el objetivo y para el MI

then EXITO

else FALLO

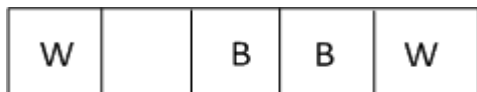
(1) (2): ejecución de la RHS de InstRule ; ejecutar los comandos 'retract' y 'assert' y actualizar la BH en consonancia

(3) Actualizar el CC las instancias que dependen de los hechos eliminados (retract)

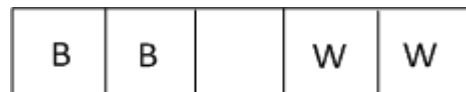
(4) Activación de la fase de matching con los nuevos hechos generados (assert)

1. Motor de inferencia (ejemplo 1)

Puzzle lineal:



situación inicial



situación final

Base de Reglas

```
(defrule empty-1-left
  ?f1 <- (puzzle $?x ?y E $?z)
  =>
  (retract ?f1)
  (assert (puzzle $?x E ?y $?z)))

(defrule empty-2-left
  ?f1 <- (puzzle $?x ?y1 ?y2 E $?z)
  =>
  (retract ?f1)
  (assert (puzzle $?x E ?y2 ?y1 $?z)))

(defrule empty-1-right
  ?f1 <- (puzzle $?x E ?y $?z)
  =>
  (retract ?f1)
  (assert (puzzle $?x ?y E $?z)))

(defrule empty-2-right
  ?f1 <- (puzzle $?x E ?y1 ?y2 $?z)
  =>
  (retract ?f1)
  (assert (puzzle $?x ?y2 ?y1 E $?z)))
```

(deffacts data
(puzzle W E B B W))

Base de Hechos

f-1: (puzzle W E B B W)

Matching

empty-1-left: f-1, ?y=W, \$?x=(), \$?z=(B B W), ?f1=1
empty-1-right: f-1, ?y=B, \$?x=(W), \$?z=(B W), ?f1=1
empty-2-right: f-1, ?y1=B, ?y2=B, \$?x=(W), \$?z=(W), ?f1=1

Agenda ó
Conjunto
Conflicto
(instancias de
reglas)

Selección

empty-1-left: f-1, ?y=W, \$?x=(), \$?z=(B B W), ?f1=1
empty-1-right: f-1, ?y=B, \$?x=(W), \$?z=(B W), ?f1=1
empty-2-right: f-1, ?y1=B, ?y2=B, \$?x=(W), \$?z=(W), ?f1=1

1. Motor de inferencia (ejemplo 1)

Ejecución

~~empty-1-right: f-1, ?y=B, \$?x=(W), \$?z=(B W), ?f1=1~~
~~empty-2-right: f-1, ?y1=B, ?y2=B, \$?x=(W), \$?z=(W), ?f1=1~~

Base de Hechos

f-2: (puzzle E W B B W)

Matching

empty-1-right: f-2, ?y=W, \$?x=(), \$?z=(B B W), ?f1= 2
empty-2-right: f-2, ?y1=W, ?y2=B, \$?x=(), \$?Z=(B W), ?f1= 2

Selección

empty-1-right: f-2, ?y=W, \$?x=(), \$?z=(B B W), ?f1= 2
empty-2-right: f-2, ?y1=W, ?y2=B, \$?x=(), \$?Z=(B W), ?f1= 2

Ejecución

~~empty-2-right: f-2, ?y1=W, ?y2=B, \$?x=(), \$?Z=(B W), ?f1= 2~~

Base de Hechos

f-3: (puzzle W E B B W)

1. Motor de inferencia (ejemplo 2)

Puzzle lineal:

W		B	B	W
---	--	---	---	---

situación inicial

B	B		W	W
---	---	--	---	---

situación final

Base de Hechos

(defacts data
 (cell 1 W)
 (cell 2 E)
 (cell 3 B)
 (cell 4 B)
 (cell 5 W))



f-1: (cell 1 W)
f-2: (cell 2 E)
f-3: (cell 3 B)
f-4: (cell 4 B)
f-5: (cell 5 W)

Base de Reglas

```
(defrule empty-1-left
  ?f1 <- (cell ?x E)
  ?f2 <- (cell ?y ?cell)
  (test (= ?y (- ?x 1)))
=>
  (retract ?f1 ?f2)
  (assert (cell ?y E))
  (assert (cell ?x ?cell)))
```

```
(defrule empty-2-left
  ?f1 <- (cell ?x E)
  ?f2 <- (cell ?y ?cell)
  (test (= ?y (- ?x 2)))
  . . . . .
=>
  (retract ?f1 ?f2)
  (assert (cell ?y E))
  (assert (cell ?x ?cell)))
```

1. Motor de inferencia (ejemplo 2)

Matching

empty-1-left: f-2, f-1, ?x=2, ?y=1, ?f1=2, ?f2=1
empty-1-right: f-2, f-3, , ?x=2, ?y=3, ?f1=2, ?f2=3
empty-2-right: f-2, f-4, ?x=2, ?y=4, ?f1=2, ?f2=4

Selección

empty-1-left: f-2, f-1, ?x=2, ?y=1, ?f1=2, ?f2=1
empty-1-right: f-2, f-3, , ?x=2, ?y=3, ?f1=2, ?f2=3
empty-2-right: f-2, f-4, ?x=2, ?y=4, ?f1=2, ?f2=4

Ejecución

~~empty-1-right: f-2, f-3, , ?x=2, ?y=3, ?f1=2, ?f2=3~~
~~empty-2-right: f-2, f-4, ?x=2, ?y=4, ?f1=2, ?f2=4~~

Base de Hechos

f-3: (cell 3 B)
f-4: (cell 4 B)
f-5: (cell 5 W)
f-6: (cell 1 E)
f-7: (cell 2 W)

Matching

empty-1-right: : f-6, f-7, ?x=1, ?y=2, ?f1=6, ?f2=7
empty-2-right: f-6, f-3, ?x=1, ?y=3, ?f1=6, ?f2=3

2. Estrategias de resolución de conflictos

El **Conjunto Conflicto** o **Agenda**: colección de **instancias de reglas** o **activaciones** resultante del proceso de matching entre las reglas de la BR y los hechos de la BH. En la Agenda puede haber cero o más instancias de reglas.

Estrategia de resolución de conflictos: criterio para seleccionar la activación de la Agenda

Refracción

Una regla solo se puede instanciar una vez con los mismos hechos (hechos con los mismos índices) y mismas instanciaciones de variables. Cuando la BH se modifica, todas las reglas se pueden volver a utilizar de nuevo siempre que se haya producido al menos un nuevo hecho que genere una nueva activación de la regla.

CLIPS aplica refracción, no permitiendo que una regla se dispare dos veces con los mismos datos, evitando así que se disparen repetidamente reglas con los mismos hechos (bucle infinito).

Este comportamiento se complementa con la opción por defecto de CLIPS de **no permitir duplicidad de hechos**. CLIPS no acepta hechos duplicados, es decir, hechos idénticos pero con diferentes índices (este comportamiento se puede, no obstante, anular seleccionando otra opción).

2. Estrategias de resolución de conflictos: ejemplo refracción y hechos duplicados

(defrule R1

(lista \$?x ?y ?z \$?w)

(test (evenp ?z))

=>

(assert (lista \$?x ?z ?y \$?w)))

BH= { f-1: (lista 12 5 24 7)}

Matching

~~R1: f-1, \$?x=(12), ?y=5, ?z=24, \$?w=(7)~~

Selección Ejecución

BH= { f-1: (lista 12 5 24 7)
f-2: (lista 12 24 5 7)}

Matching

~~R1: f-2, \$?x=(), ?y=12, ?z=24, \$?w=(5 7)~~

Selección Ejecución

BH= { f-1: (lista 12 5 24 7)
f-2: (lista 12 24 5 7)
f-3: (lista 24 12 5 7)}

La activación R1: f-1, \$?x=(12), ?y=5, ?z=24, \$?w=(7)
no se vuelve a insertar

Matching

~~R1: f-3, \$?x=(), ?y=24, ?z=12, \$?w=(5 7)~~

Selección Ejecución

BH= { f-1: (lista 12 5 24 7)
f-2: (lista 12 24 5 7)
f-3: (lista 24 12 5 7)}

CLIPS no vuelve a insertar el hecho (lista 12 24 5 7)

2. Estrategias de resolución de conflictos: agenda

Cuando hay más de una instancia en la Agenda, la estrategia de resolución de conflictos decide la activación a escoger, de modo que **la instancia que aparece en el top de la agenda es siempre la instancia que se selecciona para su ejecución.**

Prioridades explícitas

Valor numérico del **salience** de las reglas

En CLIPS, el rango del salience va desde -10,000 hasta 10,000. Cuanto mayor sea el valor, más alta es la prioridad de la regla. Si no se especifica salience en la regla entonces su prioridad es 0.

```
(defrule <rule name> ["comment"]  
  (declare (salience <integer>))  
    <conditional-element>* ; left-hand side (LHS) de la regla  
    ; condiciones a satisfacer  
=>  
....
```

2. Estrategias de resolución de conflictos: agenda

Ejemplo: queremos seleccionar 5 personas para jugar al baloncesto, preferiblemente gente con una altura mayor de 2m; sino, escoger gente por debajo de los 2m.

```
(defrule over-2m
  (declare (salience 30))
  ?f1 <- (height ?per ?tall)
    (test (>= ?tall 2))
  ?f2 <- (count ?numper)
  =>
  (retract ?f1 ?f2)
  (assert (count (+ ?numper 1))))
```

```
(defrule below-2m
  (declare (salience 10))
  ?f1<- (height ?per ?tall)
  ?f2 <- (count ?numper)
  =>
  (retract ?f1 ?f2)
  (assert (count (+ ?numper 1))))
```

```
(defrule final-1
  (declare (salience 100))
  (count 5)
  =>
  (halt)
  (printout t "We already have 5 people to play basket " crlf))
```

```
(defrule final-2
  =>
  (halt)
  (printout t "We were not able to find 5 people to play basket ", crlf))
```

```
(deffacts data
  (height John 2.02) (height Peter 1.92)(height Terry 1.86)
  (height Lucas 2.01)(height Nick 2.05)(height Joshua 1.94)
  (count 0))
```

2. Estrategias de resolución de conflictos: agenda

Profundidad

Las nuevas activaciones se manejan como una cola de prioridades. Los empates se resuelven mediante una estrategia LIFO (más prioridad para los hechos e instancias más recientes).

Anchura

Las nuevas activaciones se manejan como una cola de prioridades. Los empates se resuelven mediante una estrategia FIFO (más prioridad para los hechos e instancias más antiguos).

Aleatoria

A cada activación se le asigna un número aleatorio que se usa para determinar la posición de la activación entre aquellas activaciones que tienen el mismo salience. CLIPS permite seleccionar **Random** como estrategia de resolución de conflictos.

Especificidad

Usar la regla más específica, la que contiene más restricciones. En CLIPS, la especificidad de una regla viene determinada por el número de comparaciones que deben realizarse en la LHS de una regla. Cada comparación con una constante o a con una variable ligada a un valor añade un punto de especificidad a la regla. Del mismo modo, las llamadas a una función dentro de un test condicional también añaden un punto de especificidad a la regla.

CLIPS permite seleccionar **Simplicity** ó **Complexity** como estrategias de resolución de conflictos.

Simplicity: En caso de empate se le da más prioridad a las reglas con igual o menor especificidad

Complexity: En caso de empate se le da más prioridad a las reglas con igual o mayor especificidad

3. Ejemplos

En las siguientes transparencias se muestran 4 trazas de un SBR y se analiza el funcionamiento del Motor de Inferencia de CLIPS.

Para cada ejemplo se define: a) la BH inicial; b) las reglas que forman la BR; y c) la estrategia de resolución de conflictos

El objetivo es analizar la sucesiva aplicación del ciclo matching-selección-ejecución hasta que el problema finaliza (en estos ejemplos el problema finaliza cuando la agenda se queda vacía).

3. Ejemplo 1

Asumiendo que la estrategia de la Agenda es **ANCHURA**, realiza una traza del siguiente SBR y muestra el contenido final de la BH. Este SBR ordena una lista de números naturales dada.

```
BH={{(list 3 2 7 5)}}  
  
(defrule R1  
  ?f <- (list $?x ?y ?z $?w)  
  (test (< ?z ?y))  
=>  
  (assert (list $?x ?z ?y $?w)))
```

BH (hechos)	Agenda (instancias de reglas)	
f-1: (list 3 2 7 5)	R1: f-1 {\$?x=(3 2),?y=7, ?z=5, \$?w=(), ?f=1}	←selección (1)
	R1: f-1 {\$?x=(),?y=3, ?z=2, \$?w=(7 5), ?f=1}	←selección (2)
f-2: (list 3 2 5 7)	R1: f-2 {\$?x=(),?y=3, ?z=2, \$?w=(5 7), ?f=2}	←selección (3)
f-3: (list 2 3 7 5)	R1: f-3 {\$?x=(2 3),?y=7, ?z=5, \$?w=(), ?f=3}	←selección (4)
f-4: (list 2 3 5 7)	no hay matching	
no hay nuevos hechos (hecho duplicado)		

BH final={{(list 3 2 7 5) (list 3 2 5 7) (list 2 3 7 5)(list 2 3 5 7)}}

3. Ejemplo 2

Asumiendo que la estrategia de la Agenda es **ANCHURA**, realiza una traza del siguiente SBR y muestra el contenido final de la BH. Este SBR ordena una lista de números naturales dada.

BH={{(list 3 2 7 5)}}

```
(defrule R1
  ?f <- (list $?x ?y ?z $?w)
  (test (< ?z ?y))
=>
  (retract ?f)
  (assert (list $?x ?z ?y $?w)))
```

	BH (hechos)	Agenda (instancias de reglas)
eliminar (1) →	f-1: (list 3 2 7 5)	R1: f-1 {\$?x=(3 2),?y=7, ?z=5, \$?w=(), ?f=1} ← selección (1) R1: f-1 {\$?x=(),?y=3, ?z=2, \$?w=(7 5), ?f=1} ← eliminar(1)
eliminar (2) →	f-2: (list 3 2 5 7)	R1: f-2 {\$?x=(),?y=3, ?z=2, \$?w=(5 7), ?f=2} ← selección (2)
	f-3: (list 2 3 5 7)	no hay matching

BH final={{(list 2 3 5 7)}}

3. Ejemplo 3

Asumiendo que la estrategia de la Agenda es **ANCHURA** (mayor prioridad para las instancias y los hechos más antiguos comenzando por la regla 'move'), realiza una traza del siguiente SBR y muestra el contenido final de la BH.

BH={{(list a b c d e) (move-to-front c)}}

```
(defrule move
  ?m<- (move-to-front ?who)
  ?l <- (list $?front ?who $?rear)
=>
  (assert (list ?who $?front $?rear))
  (assert (change-list yes)))
```

```
(defrule print
  ?ch <- (change-list yes)
  (list $?list)
=>
  (retract ?ch)
  (printout t "List is " $?list crlf))
```

BH (hechos)	Agenda (instancias de reglas)
f-1: (list a b c d e) f-2: (move-to-front c)	move: f-2,f-1 {?who=c, \$?front=(a b), \$?rear=(d e) ?m=2, ?l=1} ← selección (1)
f-3: (list c a b d e) f-4: (change-list yes)	move: f-2,f-3 {?who=c, \$?front=(), \$?rear=(a b d e) ?m=2, ?l=3} ← selección (2) print: f-4,f-1 {\$?list=(a b c d e), ?ch=4} ← selección (3) print: f-4,f-3 {\$?list=(c a b d e), ?ch=4} ← eliminar (3)
(hecho duplicado)	

CLIPS> (run)

List is (a b c d e)

BH final={{(list a b c d e)(move-to-front c)(list c a b d e)}}

3. Ejemplo 4

Asumiendo que la estrategia de la Agenda es **PROFUNDIDAD** (mayor prioridad para las instancias y los hechos más recientes comenzando por la regla 'move'), realiza una traza del siguiente SBR y muestra el contenido final de la BH.

BH={{(list a b c d e) (move-to-front c)}}


```
(defrule move
  ?m<- (move-to-front ?who)
  ?l <- (list $?front ?who $?rear)
=>
  (assert (list ?who $?front $?rear))
  (assert (change-list yes)))
```

```
(defrule print
  ?ch <- (change-list yes)
  (list $?list)
=>
  (retract ?ch)
  (printout t "List is " $?list crlf))
```

BH (hechos)	Agenda (instancias de reglas)	
f-1: (list a b c d e) f-2: (move-to-front c)	move: f-2,f-1 {?who=c, \$?front=(a b), \$?rear=(d e) ?m=2, ?l=1}	← selección (1)
eliminar (2) → f-3: (list c a b d e) f-4: (change-list yes)	print: f-4,f-3 {\$?list=(c a b d e), ?ch=4} print: f-4,f-1 {\$?list=(a b c d e), ?ch=4} move: f-2,f-3 {?who=c, \$?front=(), \$?rear=(a b d e) ?m=2, ?l=3}	← selección (2) ← eliminar (2) ← selección (3)
(hecho duplicado)		

CLIPS> (run)
List is (c a b d e)

3. Ejemplo 4 (continuación)

BH (hechos)	Agenda (instancias de reglas)
f-1: (list a b c d e) f-2: (move-to-front c)	
f-3: (list c a b d e)	
 eliminar (4) f-5: (change-list yes)	print: f-5,f-3 {\$?list=(c a b d e), ?ch=5} ←selección (4) print: f-5,f-1 {\$?list=(a b c d e), ?ch=5} ←eliminar (4)

CLIPS> (run)

List is (c a b d e)

BH final={{(list a b c d e)(move-to-front c)(list c a b d e)}

3. Ejemplos: conclusiones

La elección de la estrategia de resolución de conflictos, especialmente cuando existen eliminaciones de hechos en la RHS de las reglas, puede conducir a ejecuciones distintas y marcar así una diferencia en la evaluación del SBR.

Debido a los efectos de la estrategia de resolución de conflictos, las reglas interaccionan entre sí de un modo u otro y el orden de disparo de las reglas marca la diferencia.

Para entender el funcionamiento de un SBR es necesario no solo entender el significado semántico de las reglas sino considerar también el funcionamiento del MI y tener en cuenta las condiciones bajo las cuales se disparan las reglas.

No es posible entender el significado de una regla como una unidad independiente dentro de un SBR. Hay que considerar el funcionamiento de la estrategia de resolución de conflictos y las relaciones que surgen entre las reglas.

4. Algoritmo de matching *Rete*

El **algoritmo** de matching *Rete* fue diseñado por Charles Forgy de la Universidad de Carnegie Mellon. El término *Rete* en latín indica 'red', y describe una arquitectura software para el proceso de pattern-matching [Forgy 82].

Rete representa internamente las reglas en la llamada *red Rete*. El compilador crea la red *Rete* a partir de las reglas especificadas en el SBR.

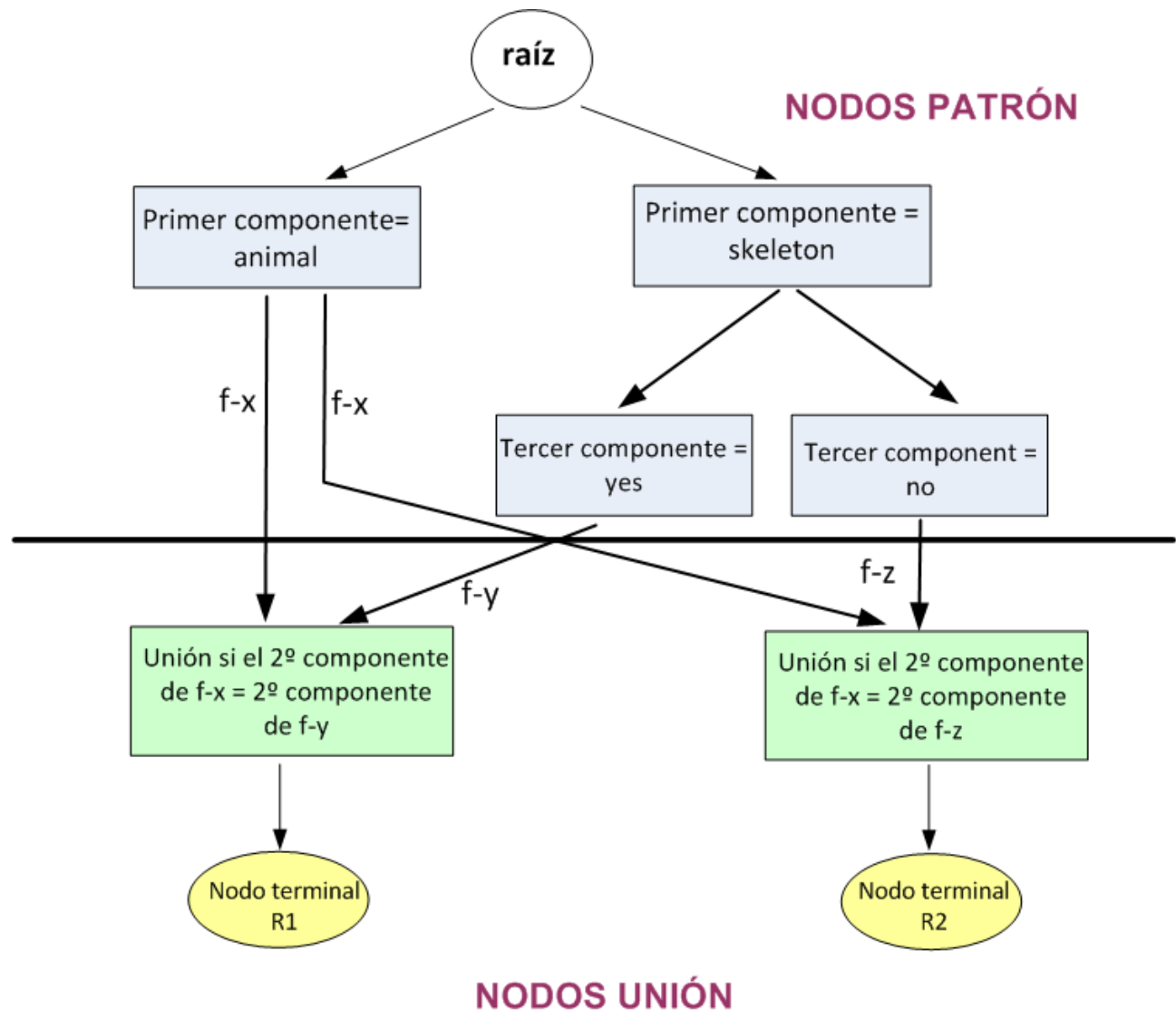
La red *Rete* consiste en una red de patrones y una red de unión de patrones.

La red *Rete* es equivalente a una máquina de estados finita que consume modificaciones de hechos.

4. Red Rete

```
(defrule R1
  (animal ?a)
  (skeleton ?a yes)
=>
  (assert (vertebrate ?a)))

(defrule R2
  (animal ?a)
  (skeleton ?a no)
=>
  (assert (invertebrate ?a)))
```



4. Algoritmo Rete: propiedades

Rete explota **dos propiedades** que son comunes a todos los SBR:

Similitud estructural

- Las reglas de un problema suelen tener patrones similares aunque no idénticos
- Rete aprovecha la *similitud estructural* de las reglas (el hecho de que con frecuencia muchas reglas contienen patrones similares o grupos de patrones similares) extrayendo los componentes comunes en los diferentes patrones de modo que solo tengan que calcularse una vez
- El compilador agrupa los patrones comunes antes de ejecutar un SBR generando la red Rete.

Redundancia temporal

- En cada ciclo del motor de inferencia, los cambios en la BH suelen ser muy pocos, aunque los contenidos sean muchos
- Rete aprovecha la redundancia temporal de cada ciclo y guarda el resultado de la fase de matching durante un ciclo, para que pueda ser usado nuevamente en el ciclo siguiente. Rete solo calcula los cambios necesarios para los nuevos hechos añadidos o los hechos eliminados.
- De esta manera, el coste computacional depende de la velocidad de cambio de la BH, que suele ser baja, y no de su tamaño.

5. Ejercicios propuestos

Ejercicio 1

Escribir **una única regla** en clips que, dada una lista de números, obtenga una nueva lista en la que se reemplace por un 0 todos aquellos números cuyo valor no coincida con su orden de posición (asumiendo que el primer número a la izquierda ocupa la posición uno). Por ejemplo, dada la lista (lista 15 2 4 6 5 3 7 8 15), obtendría la lista (lista 0 2 0 0 5 0 7 8 0).

Ejercicio 2

Dada una BH que represente una lista de naturales no ordenados y posiblemente repetidos un número indefinido de veces cada uno de los números, escribir una **única** (si es posible) regla de producción que obtenga, como BH-meta, la lista inicial de números, pero en los que no haya ningún número repetido.

Ejemplo:

BH-Inicial (lista 1 3 4 5 6 7 4 6 4 4 3 4 5 6 4 5 6 4 8 5 7 3 2 4 4 4 1 5 6 7 1 2 3 2 3 4 6 5 3 4 5 1 4 3)

y se obtendría: (lista 2 7 6 5 4 3 1 8)

Ejercicio 3

Dadas dos secuencias de ADN, escribe un SBR (una única regla, si es posible) que cuente el número de mutaciones (se puede utilizar una variable global para almacenar el número de aciertos o bien un hecho para registrar este valor). Por ejemplo, para las dos secuencias de ADN

(A A C C T C G A A A) y (A G G C T A G A A A)

hay tres mutaciones

5. Ejercicios propuestos

Ejercicio 4

Sea un SBR cuya BH inicial es $BH=\{(lista\ 1\ 2\ 3\ 4)\}$ y cuya Base de Reglas se compone de las siguientes reglas:

```
(defrule R1
  ?f <- (lista ?x $?z)
=>
  (retract ?f)
  (assert (lista ?z))
  (assert (elemento ?x)))
```

```
(defrule R2
  ?f <- (elemento ?x)
  (elemento ?y)
  (test (< ?x ?y))
=>
  (retract ?f)
  (assert (lista-new ?x ?y)))
```

asumiendo una estrategia de búsqueda en anchura (mayor prioridad para los hechos e instancias de reglas más antiguas, comenzando por R1):

- ¿cuál sería el estado final de la BH?. Realiza una traza que muestre el proceso inferencial.
- Si la BH inicial fuera $BH=\{(lista\ 1\ 2\ 2\ 4)\}$, ¿cuántos hechos (lista-new) habría en la BH final ?. ¿Cuáles?.
- Si se eliminara el comando (retract ?f) de R1, ¿qué cambios se producirían con la BH inicial= $\{(lista\ 1\ 2\ 3\ 4)\}$ respecto a los hechos (lista-new) ?
- Si se eliminara el comando (retract ?f) de R2, ¿qué cambios se producirían con la BH inicial= $\{(lista\ 1\ 2\ 3\ 4)\}$ respecto a los hechos (lista-new) ?

5. Ejercicios propuestos

Ejercicio 5

a) Diseña un sencillo SBR (**una única regla si es posible**) para determinar el número de aciertos de los boletos sellados de la Lotería Primitiva. En la BH se dispone de un hecho que determina la combinación ganadora y otro hecho que especifica la combinación jugada (se puede utilizar una variable global para almacenar el número de aciertos o bien un hecho para registrar este valor). Ejemplo de BHinicial:

BHinicial= {(boleto-ganador 2 5 8 13 24 35) (combinacion 3 5 15 24 26 37)}

b) Asumiendo que los números de la combinación no se introducen en orden creciente, ¿funcionaría el S.P. que has diseñado en el apartado a)? ¿Porqué ?.

Ejercicio 6

Un cartón de bingo se compone de 5 líneas de 5 números cada una. La BH de un SP contiene un hecho que representa una línea ganadora, por ejemplo, (linea-ganadora 12 21 34 56 77). Escribe un hecho que represente un cartón de bingo y una regla de producción que permita determinar si hay alguna línea ganadora en el cartón.

6. Anexo: sintáxis BNF de las reglas en CLIPS

```
(defrule <rule-name> [<comment>]
  [<declaration>]
  <conditional-element>*
=>
  <action>*)
```


6. Anexo: sintáxis BNF de las reglas en CLIPS (RHS)

<action> ::= (assert <ordered-pattern>) | (retract <term-retract>*) |
(bind <variable> <term-bind>) | <action-multi-value>

<term-retract> ::= <integer> | <single-vble>

<term-bind> ::= <constant> | <variable> | <function-call>

<action-multi-value> ::= (printout ...) | (read) | (readline)