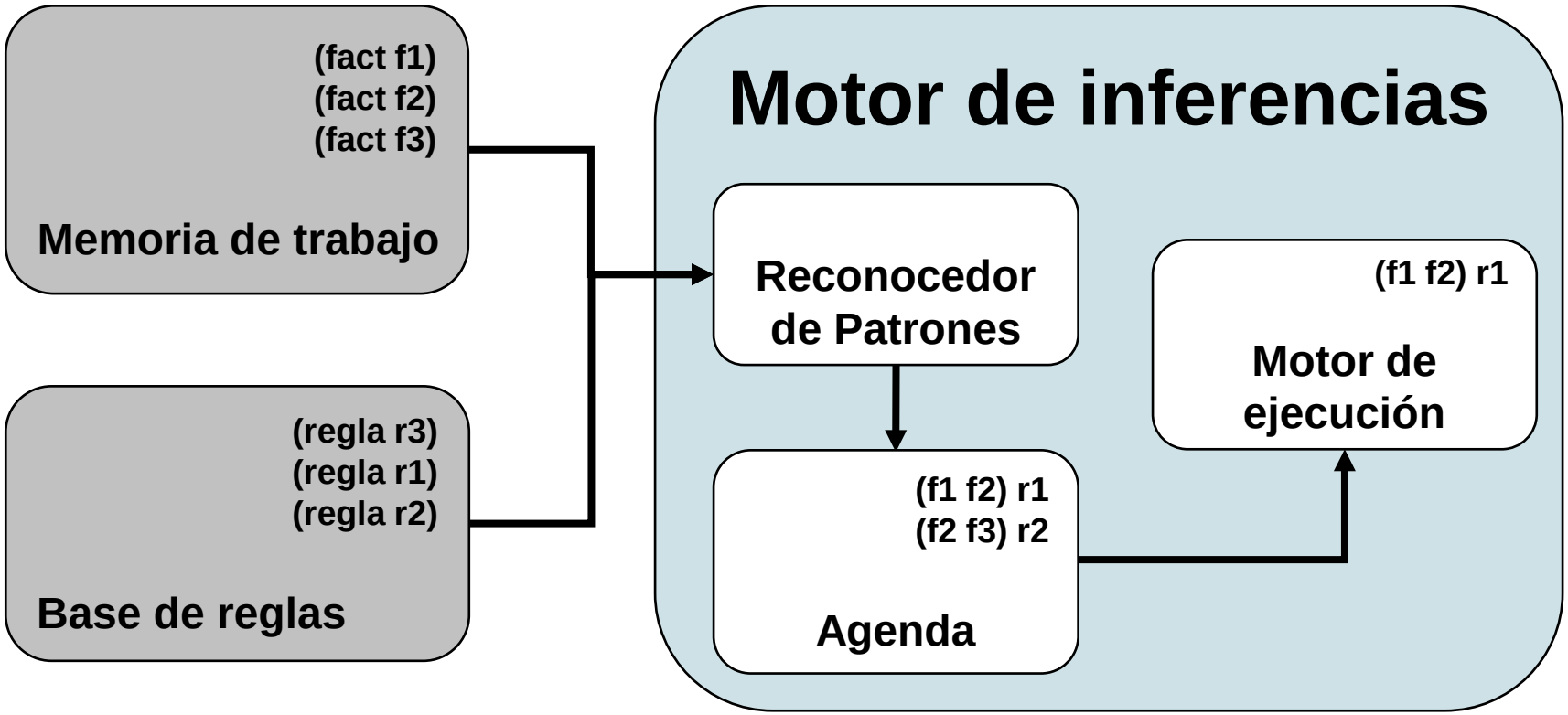
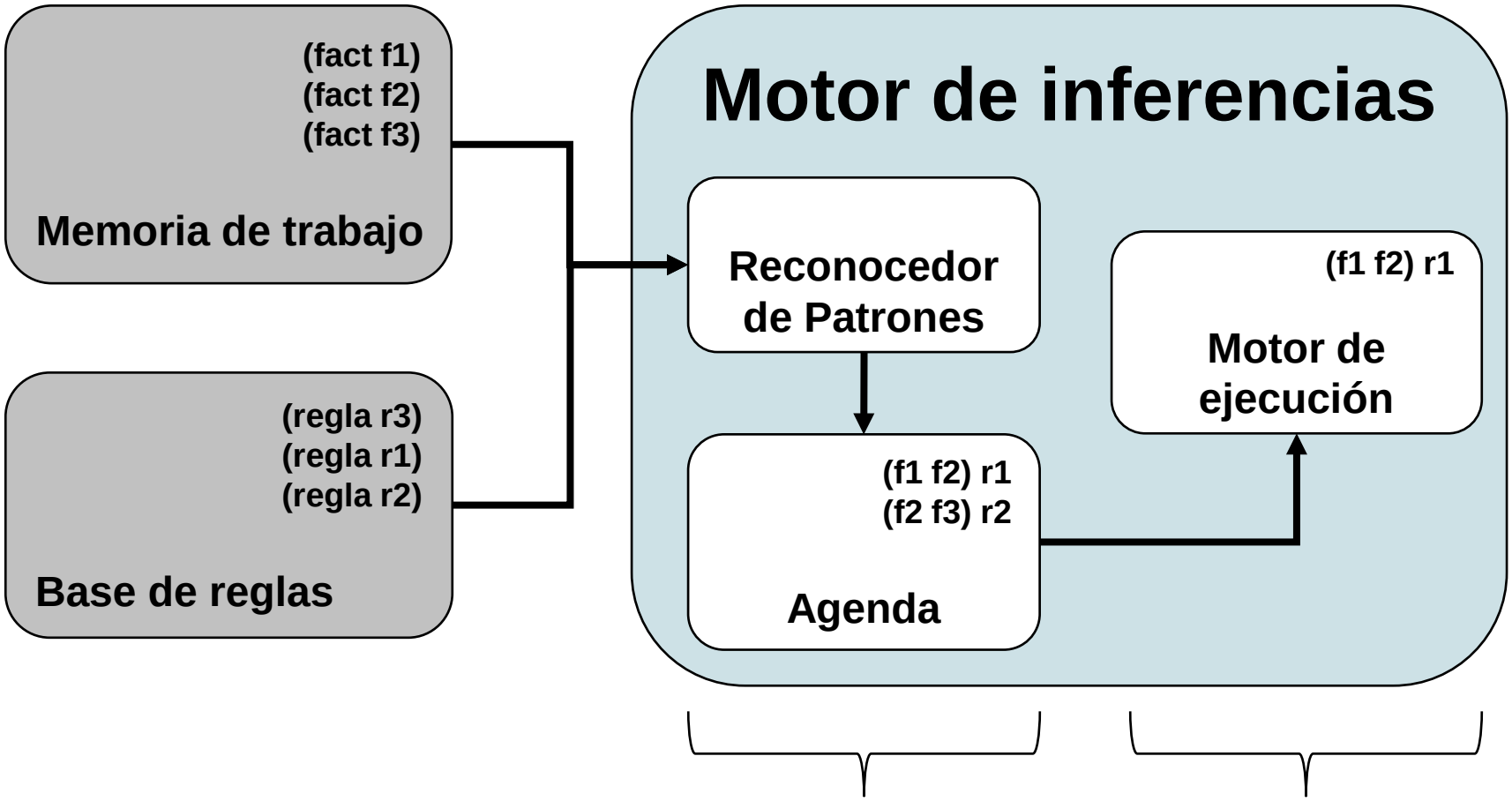
Red RETE

Arquitectura del Interprete Reglas



Adaptado de E. Friedman-Hill , “Jess in Action”, 2003

Arquitectura del Interprete Reglas



Elementos del motor de inferencias

- Reconocedor Patrones
  - Decidir que reglas se activan en base al contenido de la memoria de trabajo (filtrado)
- Agenda
  - Aplicar estrategia de resolución de conflictos para decidir que regla se dispara primero (de las activadas)
- Motor de ejecución
  - Realiza la acción de la primera regla de la agenda
  - En Clips: entorno de ejecución del lenguaje de programación

Reconocedor patrones (I)

- Aproximación Naive
  - En cada ciclo, comparar LHS 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

Reconocedor patrones (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

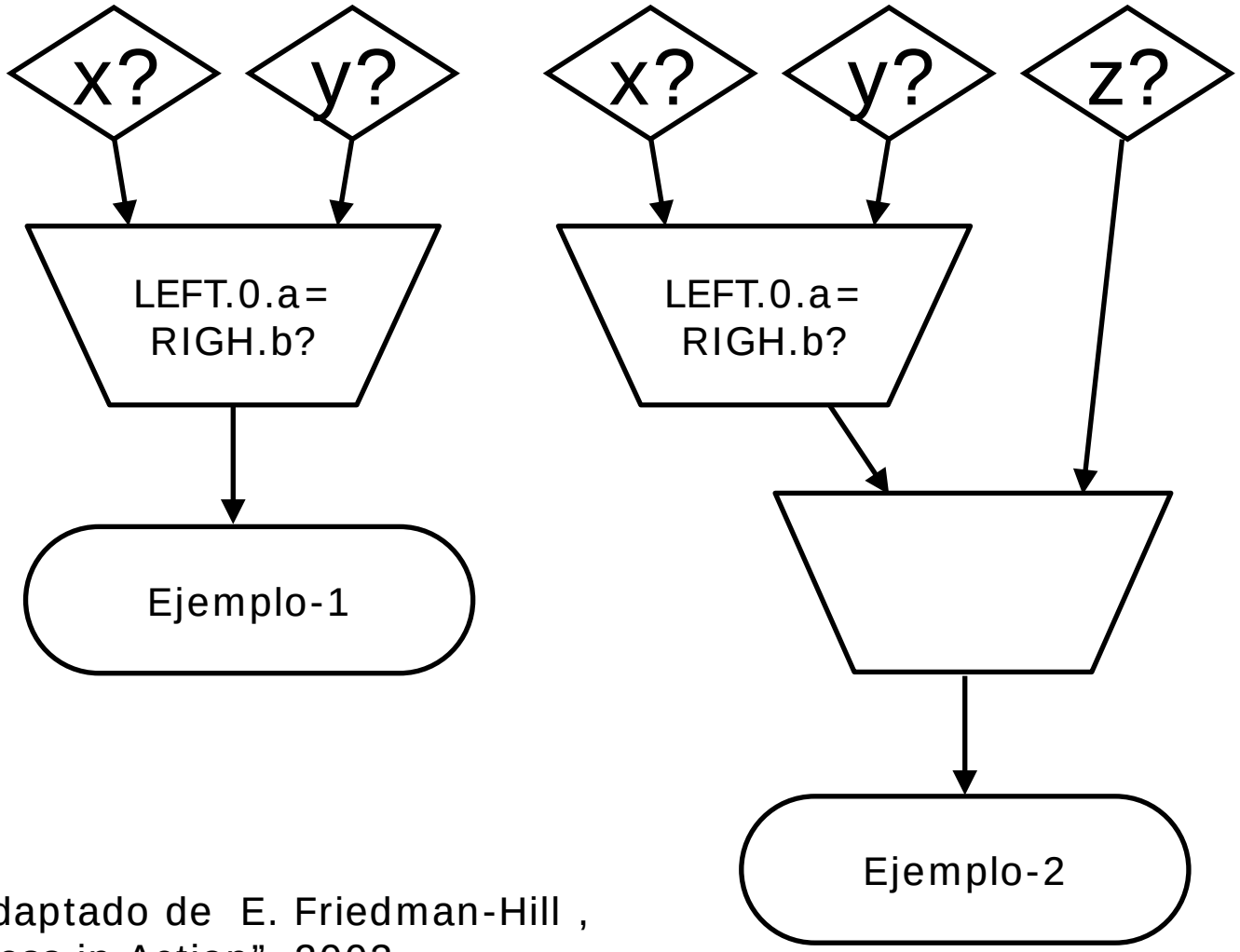
Charles L. Forgy, Artificial Intelligence 19 (1982), 17-37.

Ejemplo

```
(defrule ejemplo-1
   (x (a ?v1))
   (y (b ?v1))
=>
   (assert (p))
)
```

```
(defrule ejemplo-2
   (x (a ?v2))
   (y (b ?v2))
   (z)
=>
(assert (q))
)
```

Red RETE (sin optimizar)



Pattern network

Joint network

Adaptado de E. Friedman-Hill , "Jess in Action", 2003

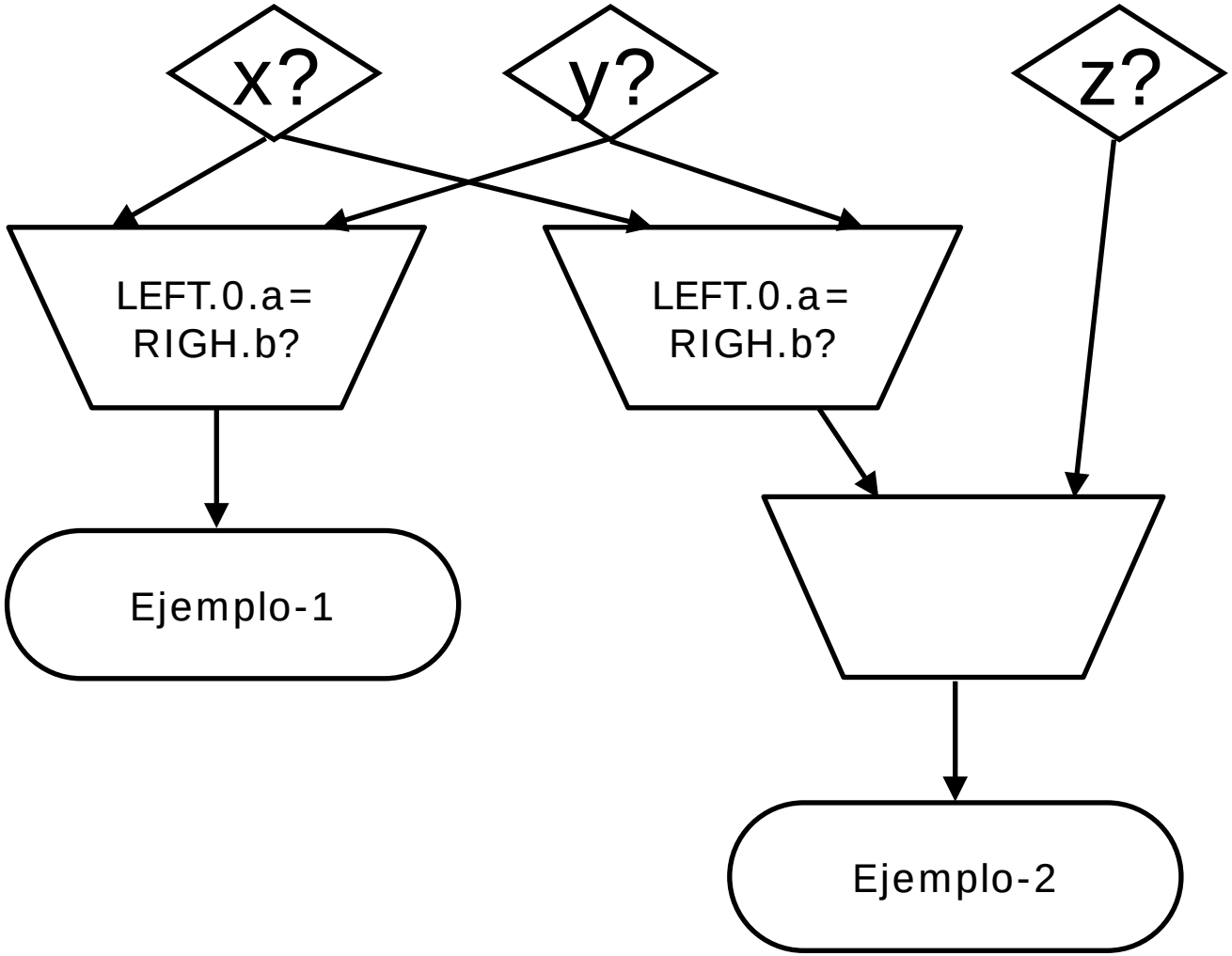
Elementos red RETE

- Nodos “patrón”: entrada red
  - =x? comprueba que la cabeza del hecho es x
  - Los hechos que superan el test se envían a la salida del nodo
- Nodos “Join”: representan confrontaciones
  - 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
  - Representan reglas
  - Registran activaciones en la agenda

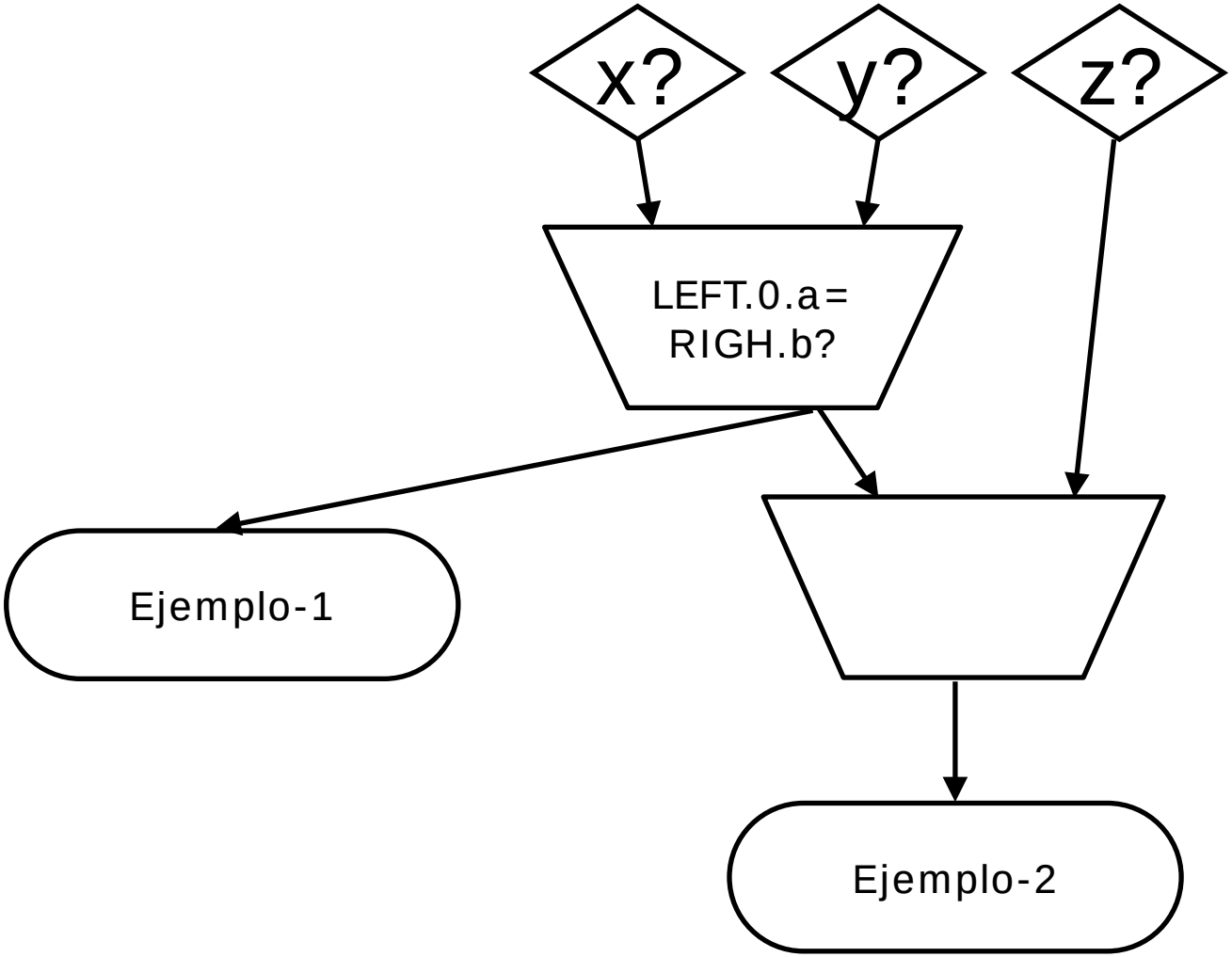
Funcionamiento de la red RETE

- Cada nuevo hecho se presenta a la “Red de Patrones”
  - Si añadimos f1 *(x (a 1))* y f2 *(y (b 1))* , son aceptados por los correspondientes nodos y enviados a los *Join*
- El nodo *Join*
  - Crea los pares de patrones que confrontan: f1,f2
  - Los envía a la salida: ejemplo-1 (activación) y siguiente *Join*
- Si añadimos nuevo hecho f3 *(z (c 2))*
  - El nodo *Join* crea el triplete f1, f2, f3 sin necesidad de procesar *x* e *y*.
  - Lo envía a la salida: se activa ejemplo-2

Optimización: compartir nodos patrón



Optimización: compartir nodos *Join*



Eficiencia algoritmo de RETE

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

Examinar red RETE

```
CLIPS>(watch compilations)
```

Ejemplo RETE

```
(deftemplate   x (slot a) )
(deftemplate   y (slot b) )
(deffacts hechos  (x (a 5))  (y (b 5)) )

(defrule ejemplo-1
    (x (a ?v1))
    (y (b ?v1))
=> (assert (p)))

(defrule ejemplo-2
    (x (a ?v2))
    (y (b ?v2))
    (z)
=> (assert (q)))
```

Construcción de la red: *Load*

```
CLIPS>(load “../EjemploRETE.CLP")
Defining deftemplate: x
Defining deftemplate: y
Defining deffacts: hechos
Defining defrule: ejemplo-1 +j+j
Defining defrule: ejemplo-2 =j=j+j
TRUE
```

Procesar hechos *deffacts: RESET*

```
CLIPS> (reset)
```

Memoria de la red (regla ejemplo-1)

```
CLIPS> (matches ejemplo-1)
Matches for Pattern 1
f-1
Matches for Pattern 2
f-2
Partial matches for CEs 1 - 2
f-1,f-2
Activations
f-1,f-2
```

Memoria de la red (regla ejemplo-2)

```
CLIPS> (matches ejemplo-2)
Matches for Pattern 1
f-1
Matches for Pattern 2
f-2
Matches for Pattern 3
 None
Partial matches for CEs 1 - 2
f-1,f-2
Partial matches for CEs 1 - 3
 None
Activations
 None
```