Bloque 1 – Representación del conocimiento y búsqueda

Tema 5:
Sistemas Basados en Reglas (SBR).
Representación en SBR: hechos y reglas.
Pattern-matching.

Bloque 1, Tema 1: Índice

1. Representación del conocimiento
2. Sistemas Basados en Reglas (SBR)
3. CLIPS: una herramienta para la construcción de SBR
4. Base de Hechos en CLIPS
5. Base de Reglas en CLIPS
6. Pattern-matching
7. Ejercicios

Bibliografía

- Capítulo 3: *Sistemas Basados en Reglas*. Inteligencia Artificial. Técnicas, métodos y aplicaciones. McGraw Hill, 2008.

1. Representación del Conocimiento

Todo problema es más sencillo de resolver si disponemos de conocimiento específico sobre él. Este conocimiento dependiente del dominio se combina con el conocimiento general sobre cómo resolver problemas.

Tipos conocimiento: declarativo, procedural

Descripción declarativa

- Conjunto de claúsulas que describen propiedades de los objetos del problema: CÓMO es algo (por ejemplo, lista de ingredientes necesarios para hacer un pastel)
- El conocimiento se representa de forma independiente a su uso posterior
- Tipos de conocimiento declarativo: relacional, heredable, deducible

Descripción procedural ó procedimental

- Conjunto de procedimientos que describen cómo utilizar el conocimiento: CÓMO hacer algo (por ejemplo, receta para hacer un pastel)
- El conocimiento representado implica la inclusión de información sobre cómo utilizarlo

Representación de redes (*declarativa*): redes semánticas, grafos conceptuales, Mapas de concepto
Representación estructurada (*declarativa & procedural*): scripts, marcos y demonios, objetos
Representación basada en la lógica (*declarativa*): lógica de primer orden, programación lógica
Representación basada en reglas (*declarativa & procedural*): reglas, Sistemas de Producción (Sistemas Basados en Reglas, Sistemas basados en el Conocimiento)

2. Sistemas Basados en Reglas (SBR)

También llamados Sistemas de Producción, son sistemas que utilizan reglas como la base de la representación del conocimiento

Se componen de tres elementos:

- Base de Reglas (BR) contiene reglas que representan cómo cambia el mundo (acciones del problema)
- Base de Hechos (BH) contiene hechos que representan el estado del mundo (situaciones)
- Motor de inferencia: mecanismo de control de un SBR

Reglas del tipo if-then:

- $p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow a1 \wedge a2 \wedge \ldots \wedge an$
- Antecedente (premisa o condición): conjunto de condiciones (sentencias en lógica de predicados, condiciones sobre objetos …)
- Consecuente (conclusión o acción): conjunto de operaciones. Una operación puede ser:
  - Añadir un hecho a la BH
  - Eliminar un hecho de la BH (**diferente de la programación lógica!**)
    - Negación por fallo (lo que no está en la BH es falso)
  - Consultas al usuario
  - Acciones externas (print, read, ….)

2. Sistemas basados en reglas: componentes

Base de Hechos

- También llamada memoria de trabajo ó base de datos
- Contiene ***hechos*** sobre el mundo (conocimiento declarativo), que pueden observarse directamente o deducirse a través de las reglas
- Ejemplos: *el bloque rojo está situado encima del bloque verde*, *la lista contiene los elementos a b c d e*, *el tanque está vacío*, etc.
- Se puede modificar mediante reglas (añadir hechos, eliminar hechos)

Base de Reglas

- También llamada Base de Conocimiento (permite representar conocimiento procedural)
- Contiene ***reglas***, cada regla representa un paso de la resolución del problema. Una regla es un conjunto de condiciones (precondiciones) junto con las operaciones a realizar si las condiciones se satisfacen (efectos).
- Las reglas son conocimiento duradero (persistente) sobre el dominio. Una vez definidas, no se modifican, solo el experto del dominio puede modificarlas.
- Las reglas se utilizan para representar las acciones del problema
- Ejemplo (regla '*coger bloque*'):

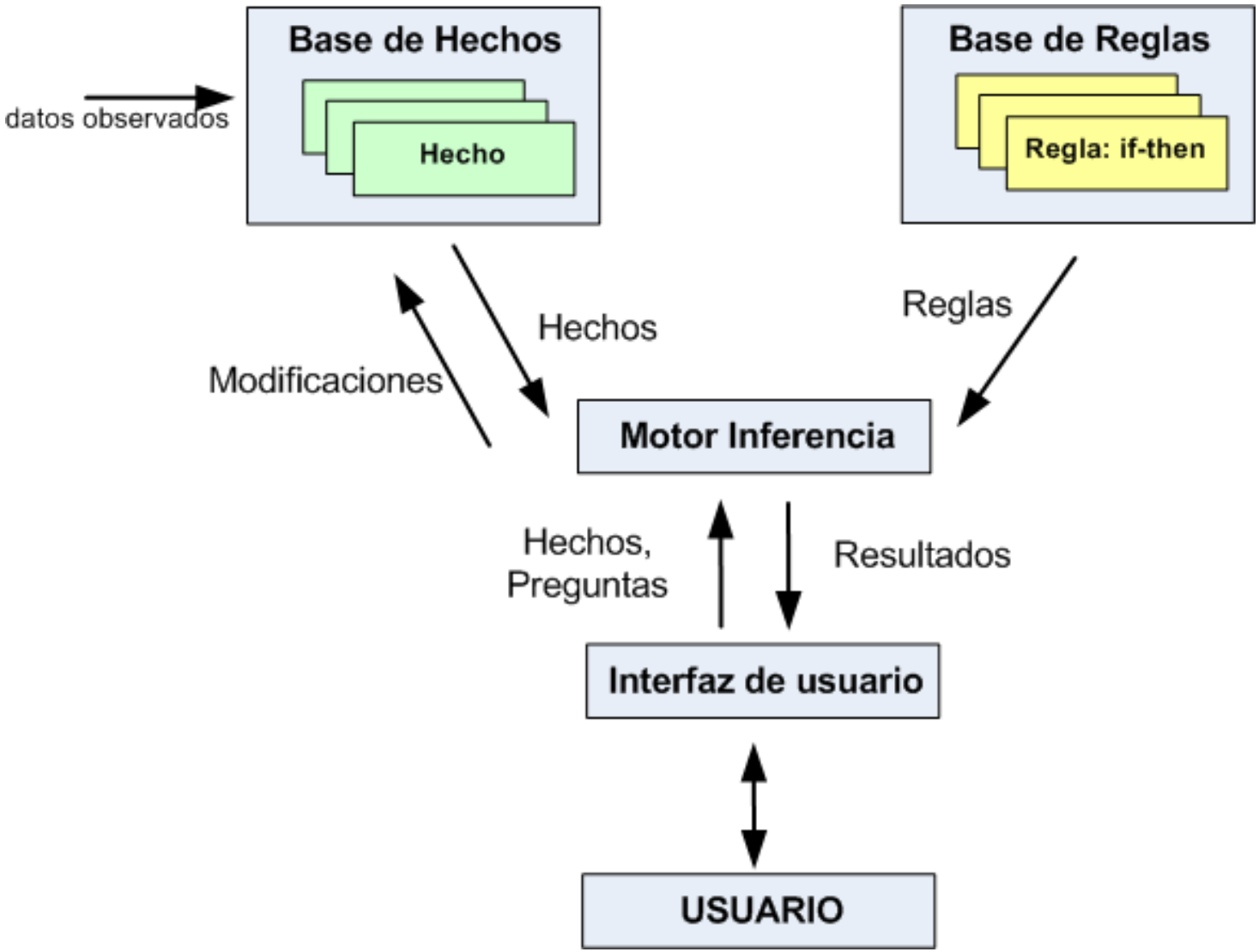
(si bloque rojo está encima del bloque verde) y
(no hay un bloque encima del bloque rojo)
=>
(bloque rojo sujeto por la grúa) y
(bloque rojo no está encima del bloque verde) y
(bloque verde no tiene un bloque encima)

2. Sistemas basados en reglas: componentes

Motor de inferencia

- Procesa la información de la BH y BR
- Es el mecanismo de razonamiento, independiente del dominio, para SBR
- Selecciona una regla de la BR y la aplica
- Un motor de inferencia se caracteriza por:
  - El tipo de encadenamiento
  - El proceso de unificación (pattern-matching)
  - El mecanismo de control para seleccionar y ejecutar reglas
- Dos tipos de motor de inferencia:
  - Encadenamiento hacia delante o razonamiento dirigido por los datos (las reglas unifican el antecedente e infieren el consecuente)
  - Encadenamiento hacia atrás o razonamiento dirigido por el objetivo (las reglas unifican el consecuente y prueban el antecedente)

2. Sistemas basados en reglas: arquitectura



3. CLIPS: una herramienta para la construcción de SBR

CLIPS (C Language Implementation Production System) es una herramienta que proporciona todas las funcionalidades para la construcción de SBR. CLIPS ha sido desarrollada por el Software Technology Branch (STB), NASA/Lyndon B. Johnson Space Center.

CLIPS proporciona:

- Mecanismos para definir los *hechos* de la BH
- Mecanismos para definir las *reglas* de la BR
- También proporciona soporte para representar conocimiento procedural y orientado a objetos
- Un motor de inferencia hacia delante basado en el algoritmo de match Rete
- Diferentes estrategias para la resolución de conflictos (selección de una instancia de regla)

4. Base de Hechos en CLIPS

Los hechos son los elementos de información básica. La BH es una lista de hechos.

Un hecho es una lista de valores atómicos que se referencian posicionalmente (hecho ordenado)

Hecho ordenado: símbolo seguido de cero o más elementos separados por espacios en blanco y delimitados por un paréntesis de apertura a la izquierda y un paréntesis de cierre a la derecha. El primer elemento de un hecho establece una 'relación' con el resto de elementos del hecho ordenado.

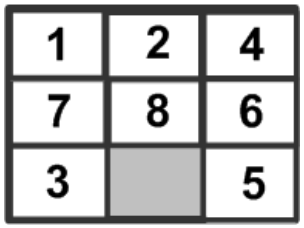
<fact> ::= (<symbol> <constant>*)
<constant> ::= <symbol> | <string> | <integer> | <float> | <instance-name>

Ejemplos de hechos:

(empty)
(on blockA blockB)
(on blockB table)
(grocery-list bread milk eggs)
(person name "John" age 24 activity journalist)
(water-jug jug X contents 0 capacity 4 jug Y contents 0 capacity 3)
(puzzle 1 2 4 7 8 6 0 3 5)

4. Base de Hechos en CLIPS

Constructor para definir un grupo inicial de hechos: **(deffacts <deffacts-name> [<comment>] <fact>*)**

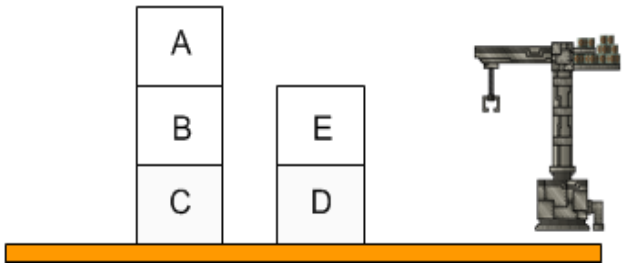



```
(deffacts puzzle_Rep1
     (puzzle 1 2 4 7 8 6 3 0 5)}
```

BH={ f-1: (puzzle 1 2 4 7 8 6 3 0 5)}

```
(deffacts puzzle_Rep2
   (puzzle 1 1 1)
   (puzzle 1 2 2)
   (puzzle 1 3 4)
   (puzzle 2 1 7)
    …)
```

BH={ f-1: (puzzle 1 1 1)
f-2: (puzzle 1 2 2)
f-3: (puzzle 1 3 4)
f-4: (puzzle 2 1 7)
… }




```
(deffacts blocks_Rep1
     (tower maximum height 3)
     (blocks tower 1 A B C tower 2 E D hoist nothing))
```

```
(deffacts blocks_Rep2
     (tower maximum height 3)
     (clear A)
     (on A B)
     (on B C)
     (on C table)
     (hoist nothing)    ….)
```

5. Base de reglas en CLIPS

La BR contiene reglas. Una regla tiene dos partes:

- El antecedente, premisa o condición (Left-Hand Side –LHS- de la regla): representa las condiciones que deben satisfacerse en una situación del problema para que la regla sea aplicable o ejecutable
- El consecuente, conclusión o acción (Right-Hand Side –RHS- de la regla): representa las acciones que se ejecutarán en la situación del problema cuando la regla sea aplicable y se seleccione para ser ejecutada (disparo de la regla)

```
(defrule <rule name> ["comment"]
   <conditional-element>*          ; left-hand side (LHS)  de la regla
                                   ; condiciones a satisfacer
=>
   <actions>*      ;; right-hand side (RHS)  de la regla
                   ;; acciones a ejecutar cuando la regla se dispara (cuando la regla
                   ;; se selecciona para ser ejecutada)
)
```

5. Base de reglas en CLIPS: parte izquierda de las reglas (LHS)

Los elementos condicionales de la LHS de una regla son de dos tipos:

- Patrón (pattern): restricción que se utiliza para determinar los hechos que satisfacen la condición especificada en el patrón. Un patrón es equivalente a un predicado en lógica de primer orden. Los patrones de una regla unifican con los hechos.
- Test: expresión booleana que se evalúa a TRUE o FALSE.

```
(defrule move_cell_to_left   ;; blank to right
      (puzzle  ?x0  ?y0  0)
      (puzzle  ?x1  ?y1  ?cell)
      (test (and (<  ?y0  3)(=  ?x0 ?x1)))
      (test (= ?y1 (+ ?y0 1)))
 =>
    ....
```

```
(defrule  pour_Y_into_X
      (contents jug X ?x jug Y ?y)
      (capacity jug X ?cap_X)
      (test  (and (< ?x ?cap_X)(> ?y 0)))
      (test (<= (+ ?x ?y) ?cap_X))
 =>
    ....
```

Todos los patrones y tests de la LHS de la regla se deben satisfacer para que la regla sea aplicable:

1. Un patrón se satisface si existe al menos un hecho en la BH que hace matching con el patrón
2. Un test se satisface si se evalúa a TRUE

5. Base de reglas en CLIPS: parte derecha de las reglas (RHS)

La RHS de una regla representa conocimiento procedural (comandos ejectuables).

- assert: añade hechos en la BH **(assert <fact>+)**
- retract: borra hechos de la BH **(retract <fact-index>+)**

BH= { f-1: (block A colour red) f-2: (block B colour blue) f-3: (available paint green) f-4: (available paint red) }

```
(defrule paint-block
    ?fa <- (block A colour red)
    ?fb <- (available paint green)
  =>
    (retract ?fa ?fb)
    (assert (block A colour green))
    (printout t "Bloque pintado de verde " crlf))
```

?fa y ?fb son las variables que almacenan los índices de los hechos que hacen matching con los patrones (?fa=1, ?fb=3)

cuando la regla se dispara, el comando 'retract' borra los hechos cuyos índices son ?fa y ?fb, los hechos que hacen matching con cada uno de los patrones

BH= {f-2: (block B colour blue) f-4: (available paint red) f-5: (block A colour green)}

6. Pattern-matching (unificación de patrones)

Unificación de patrones (pattern matching o *unificación sintáctica*). Es el proceso de unificación de los patrones de la parte izquierda de una regla con los hechos de la BH. El motor de inferencia de CLIPS unifica automáticamente los patrones de las reglas de la BR con los hechos de la BH y determina las reglas que son aplicables (aquellas que satisfacen su LHS).

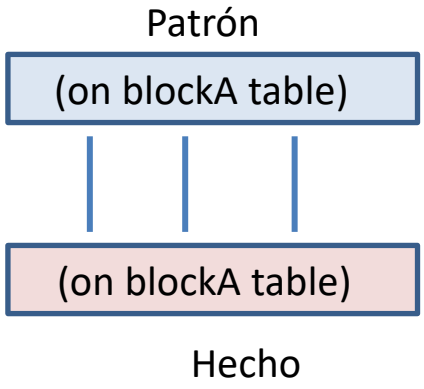
Los patrones pueden contener:

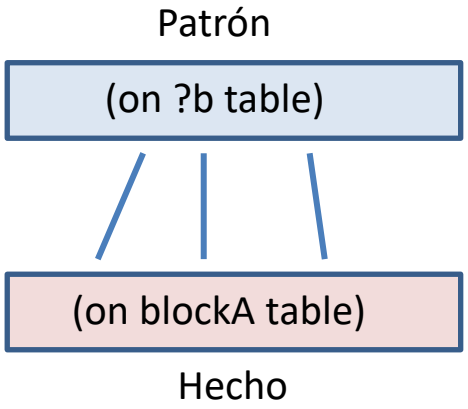
- Constantes
- Variables mono-valuadas ó simples: variable cuyo nombre comienza por un signo de interrogación '?'. Las variables mono-valuadas se ligan a un único valor.
- Variables multi-valuadas: variable cuyo nombre comienza con el signo '$?'. Las variables multi-valuadas se pueden ligar a cero o más valores.
- Comodín (wildcards). Su nombre es simplemente el símbolo '?' or '$?'. Son variables especiales que se pueden ligar a cualquier constante o conjunto de constantes pero sin mantener los valores ligados; unifican valores del hecho pero sin mantener los valores ligados a una variable (en caso de que esto no sea necesario).

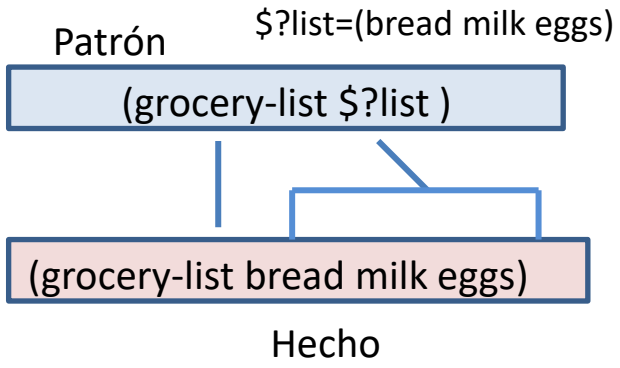
Unificación (matching)

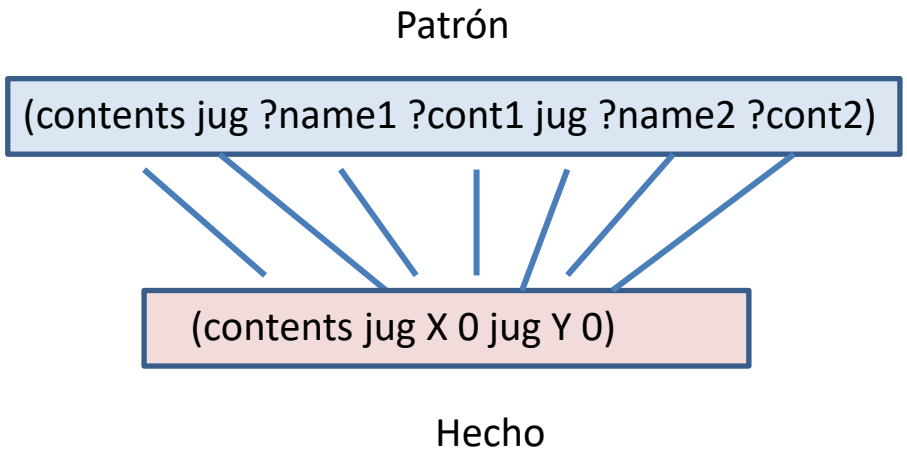
Elemento del patrón	Elemento del hecho
Constante	Constante
Variable mono-valuada (?x)	Un único valor
Variable multi-valuada ($?x)	Cero o más valores
Comodín mono-valuado (?)	Un único valor sin mantener la ligadura
Comodín multi-valuado ($?)	Cero o más valores sin mantener la ligadura

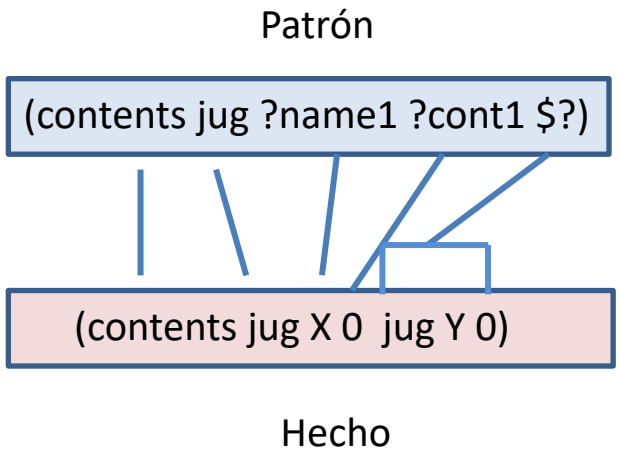
6. Pattern-matching (unificación de patrones)



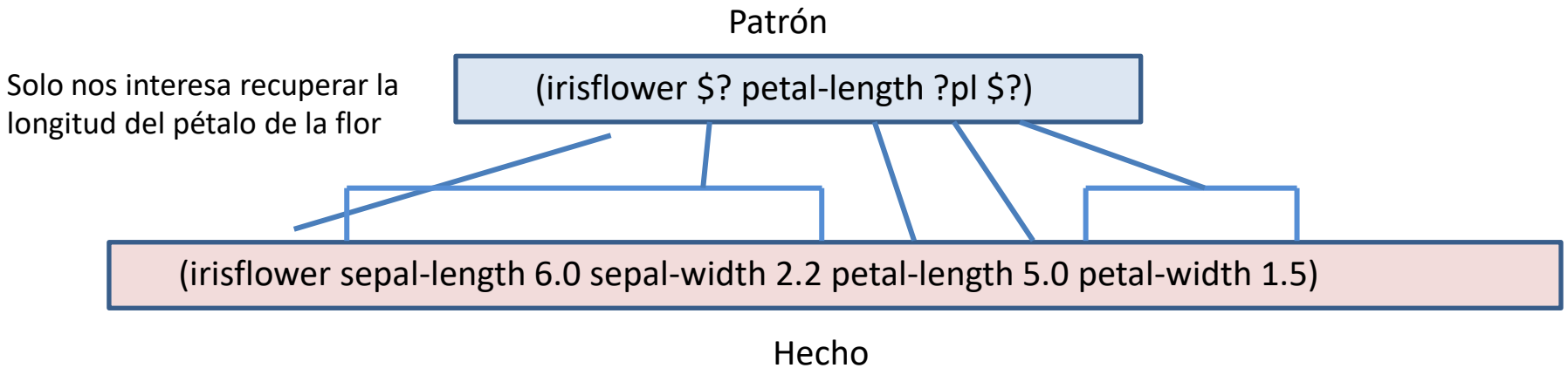




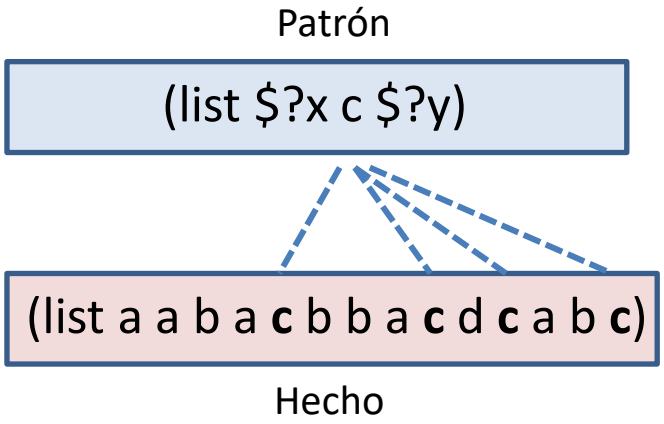




6. Pattern matching (unificación de patrones)



Cuando dos o más variables multi-valuadas aparecen en un patrón, el patrón y el hecho pueden unificar (match) más de una vez



#Match	$?x	$?y
1	(a a b a c b b a c d c a b)	()
2	(a a b a c b b a c d)	(a b c)
3	(a a b a c b b a)	(d c a b c)
4	(a a b a)	(b b a c d c a b c)

6. Pattern-matching (unificación de patrones)

Opciones de matching entre un patrón y los hechos de la BH

1	patrón	hecho	Solo una unificación
2	patrón	hecho 1 hecho 2	El mismo patrón unifica una vez con diferentes hechos
3	patrón 1 patrón 2	hecho	Diferentes patrones unifican con el mismo hecho
4	patrón	hecho	Varias unificaciones entre el patrón y el hecho debido al uso de variables multi-valuadas

Unificación de reglas (rule matching): la LHS de una regla unifica cuando existe al menos un hecho en la BH que unifica con cada patrón de la regla y todos los tests de la regla se evalúan a TRUE

Cuando una regla unifica, se dice que la regla es aplicable y que se ha encontrado una ***instancia de la regla***

Puede haber más de una unificación (más de una instancia) para la misma regla dependiendo del número de hechos que unifiquen con cada patrón

6. Pattern matching (unificación de reglas)

Base de Reglas

```
(defrule R1
     (number 2)
=>
 ...
(defrule R2
     (number ?num)
=>
 ...
(defrule R3
     (number ?n1)
     (number ?n2)
=>
 ...
(defrule R4
     (list $? ?n1 $? ?n2 $?)
     (test (= (mod ?n1 2) 0))
     (test (<> (mod ?n2 2) 0))
=>
 ...

(defrule R5
    (number ?n1)
    (list $?list)
    (test (member ?n1 $?list))
=>
 ...
```

```
(deffacts data
    (number 2)
    (number 5)
    (number 17)
    (list 26 2 17 18))
```

Base de Hechos

```
f-1: (number 2)
f-2: (number 5)
f-3: (number 17)
f-4: (list 26 2 17 18)
```

Instancias de reglas

Rule	# Match	Facts
R1	1	f-1
R2	1	f-1: {?num=2}
	2	f-2: {?num=5}
	3	f-3: {?num=17}
R3	1	f-1, f-1: {?n1=2, ?n2=2}
	2	f-1, f-2: {?n1=2, ?n2=5}
	3	f-1, f-3: {?n1=2, ?n2=17}
	4	f-2, f-1: {?n1=5, ?n2=2}
	5	f-2, f-2: {?n1=5, ?n2=5}
	6	f-2, f-3: {?n1=5, ?n2=17}
	7	f-3, f-1: {?n1=17, ?n2=2}
	8	f-3, f-2: {?n1=17, ?n2=5}
	9	f-3, f-3: {?n1=17, ?n2=17}
R4	1	f-4: {?n1=2, ?n2=17}
	2	f-4: {?n1=26,?n2=17}
R5	1	f-1,f-4: {?n1=2, $?list=(26 2 17 18)}
	2	f-3,f-4: {?n1=17, $?list=(26 2 17 18)}

6. Pattern-matching (unificación de reglas)

Base de Reglas

```
(defrule R1
     (list $? ?x $?)
     (test (evenp ?x))
=>
     (assert (number even ?x))
     (printout t ?x " is an even number of the list " crlf))


(defrule R2
     (list $? ?x $?)
     (test (oddp ?x))
=>
     (assert (number odd ?x))
     (printout t ?x " is an odd number of the list " crlf))
```

Base de Hechos

```
(deffacts data
     (list 26 2 17 18))
```

→

f-1: (list 26 2 17 18)

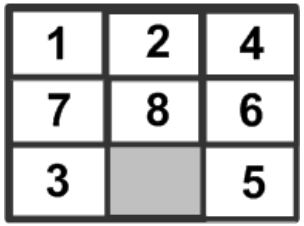
Instancias de reglas

Rule	# Match	Facts
R1	1 2 3	f-1: {?x=18} f-1: {?x=2} f-1: {?x=26}
R2	1	f-1: {?x=17}

Unificación de reglas

- Cuatro instancias de reglas, 3 instancias de la regla R1 y una de la regla R2
- El motor de inferencia seleccionará una de las instancias
- Una vez seleccionada una instancia, ésta se disparará y se ejecutará la parte RHS de la misma

6. Pattern-matching (unificación de reglas)



BH= { f-1: (puzzle 1 1 1) f-2: (puzzle 1 2 2) f-3: (puzzle 1 3 4) f-4: (puzzle 2 1 7) f-5: (puzzle 2 2 8) f-6: (puzzle 2 3 6) f-7: (puzzle 3 1 3) f-8: (puzzle 3 2 0) f-9: (puzzle 3 3 5)}

```
(defrule move_cell_to_left  ;; blank to right
      ?fa <- (puzzle  ?x0  ?y0  0)
      ?fb <- (puzzle  ?x1  ?y1  ?cell)
      (test (and (<  ?y0  3)(=  ?x0 ?x1)))
      (test (= ?y1 (+ ?y0 1)))
=>
      (retract ?fa ?fb)
      (assert (puzzle ?x0 ?y0 ?cell))
      (asssert (puzzle ?x1 ?y1 0)))
```

Instancias de reglas

Rule	# Match	Facts
move_cell_to_left	1	f-8, f-9: {?x0=3, ?y0=2, ?x1=3, ?y1=3, ?cell=5} ?fa=8, ?fb=9

BH= { f-1: (puzzle 1 1 1) f-2: (puzzle 1 2 2) f-3: (puzzle 1 3 4) f-4: (puzzle 2 1 7) f-5: (puzzle 2 2 8) f-6: (puzzle 2 3 6) f-7: (puzzle 3 1 3) f-10: (puzzle 3 2 5) f-11: (puzzle 3 3 0)}

7. Ejercicios

Ejercicio 1

Sea un SBR cuya BH inicial es BH={(lista 7 3 2 5)} y cuya BR se compone de la siguiente regla:

```
(defrule R1
    ?f <- (lista $?x ?y ?z $?w)
     (test (< ?z ?y))
   =>
   ....
```

Determina el conjunto de instancias de R1 que se generan tras la aplicación del proceso de pattern-matching.

Ejercicio 2

Sea un SBR cuya BH inicial es BH={(lista 3 6 8 5 10)} y cuya BR se compone de las siguientes reglas:

```
(defrule R1
   ?f <- (lista ?x ?y $?z)
         (test (< ?x ?y))
   =>
   ...
```

```
(defrule R2
      ?f <- (lista ?y $?x)
            (test (and (<= (length $?x) 3) (>= (length $?x) 1))
      =>
      ...
```

Determina el conjunto de instancias de R1 y R2 que se generan tras la aplicación del proceso de pattern-matching.

7. Ejercicios

Ejercicio 3

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

```
(defrule R1
   ?f <- (lista ?x $?z)
   =>
…
```

```
(defrule R2
   ?f <-  (elemento ?x)
          (elemento ?y)
          (test (< ?x ?y))
   =>
   ….
```

Determina el conjunto de instancias de R1 y R2 que se generan tras aplicar el proceso de pattern-matching.

Ejercicio 4

Sea un SP cuya BH inicial es BH={(lista a a b a) (par a 1) (par b 2)} y cuya BR se compone de la siguiente regla:

```
(defrule R1
   ?f <- (lista $?x ?sym $?y)
         (par ?sym ?num)
   =>
   …
```

Determina el conjunto de instancias de R1 que se generan tras la aplicación del proceso de pattern-matching.