UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

Departamento de Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

BOLETÍN DE EJERCICIOS
SISTEMAS INTELIGENTES

Bloque 1: Representación del Conocimiento

Septiembre 2019

CUESTIONES

1) Dado un SBR, con una única regla:

```
(defrule regla-1
   ?f <- (lista ?x $?y ?x $?y ?x)
=>
        (retract ?f)
        (assert (lista $?y)))
```

, y la BH inicial {(lista a b c b c b c b c b a b c b c b c b c b a)}, ¿cuál será el estado final de la Base de Hechos?

A. {(lista a)}
B. {(lista a) (lista)}
C. {(lista b a b)}
D. {(lista c b c)}

---

2) En un determinado SBR para resolver un problema, la declaración de una BH inicial representa:

A. Los datos estáticos del problema.
B. Los datos dinámicos del problema, en su estado inicial.
C. Los datos estáticos y dinámicos, en su estado inicial.
D. Ninguna de las anteriores es cierta. La BH representa los hechos y el conjunto de reglas del dominio.

---

3) Supongamos que tenemos en una mesa varias cajas de distintos tamaños que queremos apilar en una torre, de mayor a menor tamaño, mediante acciones de 'apilado'. La regla meta aclara el objetivo deseado. Definimos un SBR donde la BH inicial se describe de la siguiente forma:

```
(deffacts prueba   (mesa 2 5 1 6 8 7 4 torre ))
```

, y se definen las siguientes reglas:

```
(defrule mesa-a-torre
  (mesa  $?rest1  ?x  $?rest3  torre  $?rest1 ?y )
  (test (> ?y ?x))
 =>
   (assert (mesa $?rest1 $?rest3  torre  $?rest1 ?y ?x )))

(defrule mesa-a-torre-vacia
   (mesa  $?rest1 ?x $?rest3  torre )
 =>
   (assert (mesa  $?rest1 $?rest3  torre  ?x )))
```

```
(defrule meta
   (mesa   torre 1 2 4 5 6 7 8 )
 =>
   (halt))
```

¿Cuál de las siguientes afirmaciones es correcta?

A. El SBR definido funciona correctamente y obtendrá en la torre las cajas ordenadas por tamaño.
B. El SBR definido funciona correctamente, obtendrá en la torre las cajas ordenadas por tamaño, y la regla meta no es necesaria ya que el proceso acabará al quedar la mesa vacía.
C. El SBR solo funcionaría si las cajas en la mesa están ordenadas de menor a mayor, por ejemplo: (deffacts prueba (mesa 1 2 4 5 6 7 8 torre)).
D. Ninguna de las afirmaciones anteriores es cierta.

---

4) Sea el siguiente SBR para calcular el factorial de un número, ¿cuál de las siguientes afirmaciones es correcta?

```
(deffacts factorial (numero 5) (fact 1))

(defrule fact
  ?f1 <- (numero ?n1)
  ?f2 <- (fact ?n2)
=>
  (retract ?f1 ?f2)
  (assert (numero (- ?n1 1)))
  (assert (fact (* ?n2 ?n1 ))))
```

A. El SBR funciona correctamente.
B. Lo anterior no es cierto, pues se requiere una condición (test (> ?n1 1)) en la premisa de la regla para que funcione correctamente.
C. La modificación anterior no bastaría, siendo además necesario una regla de parada: (defrule parada (declare (salience 100)) (numero 1) => (halt)).
D. Ninguna de las anteriores es cierta.

---

5) Sea el hecho:

(escuela clase 1 niños 15 niñas 18 clase 2 niños 21 niñas 14 clase 3 niños 16 niñas 17)

, donde el número que aparece después del símbolo ‘clase’ indica el identificador de dicha clase, y los valores numéricos que aparecen después de los símbolos ‘niños’ o ‘niñas’ indican el número de

niños o niñas de la clase correspondiente. Indica el patrón adecuado para obtener únicamente el identificador de una clase cualquiera y el número de niñas de dicha clase:

A. (escuela $? clase ?c $? niñas ?na $?)
B. (escuela $? clase ?c niños ? niñas ?na $?)
C. (escuela $? clase ? niños $? niñas ?na $?)
D. (escuela clase ?c niños ? niñas ?na)

---

6) Sea un SBR formado por BHinicial={(lista 23 14 56 33)}, y las siguientes reglas:

```
(defrule R1
  (declare (salience 100))
   ?f <- (lista $?x ?z ?y $?w)
         (test (< ?z ?y))
 =>
  (assert (lista $?x ?z ?y $?w)))
```

```
(defrule R2
        (declare (salience 150))
         ?f <- (lista $?x ?z ?y $?w)
               (test (>= ?z ?y))
         =>
         (assert (lista $?x ?z ?y $?w)))
```

```
(defrule final
  (declare (salience 200))
    (lista $?list)
 =>
   (halt))
```

¿Cuál sería el contenido del Conjunto Conflicto (Agenda) tras el primer *pattern-matching*?

A. Una instancia de la regla R1, una instancia de la regla R2 y una instancia de la regla final
B. Dos instancias de la regla R2 y una instancia de la regla R1
C. Dos instancias de la regla R2, una instancia de la regla R1 y una instancia de la regla final
D. Una instancia de la regla final

---

7) Sea de nuevo el SBR de la pregunta 6. Asumiendo que la estrategia del Conjunto Conflicto (Agenda) es anchura, ¿cuál es la primera instancia de regla que selecciona el motor de inferencia de CLIPS para ser ejecutada? Indica cuál de las siguientes afirmaciones es CIERTA:

A. Selecciona una instancia de la regla R1.
B. Selecciona una instancia de la regla R2.
C. Selecciona una instancia de la regla final.
D. Ninguna de las anteriores.

---

8) Sea un SBR, con una única regla:

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

```
(retract ?f)
(assert (lista $?y ?x $?z))
(printout t "La lista se ha modificado " crlf))
```

, y la BH inicial {(lista a b a b a)}. Tras ejecutar el SBR, ¿cuántas veces se habrá mostrado en pantalla el mensaje "La lista se ha modificado "?

A. 1
B. 2
C. 3
D. 4

---

9) Tenemos un montacargas que recoge paquetes de la planta baja y los reparte entre diferentes plantas. Una instancia del problema tiene dos paquetes A y B, con destino 2º y 3ª planta, peso 2 kg. y 8 kg. respectivamente. El montacargas está en la planta baja y su peso máximo es 40 kg. ¿Cuál de las siguientes representaciones NO serviría para realizar una búsqueda en grafo en una representación basada en estados?

A. (montacargas planta 0 carga paquete A 2 2 paquete B 3 8 peso-maximo 40 nivel 0)
B. (montacargas planta 0 carga paquete A 2 2 paquete B 3 8) (peso-maximo 40)
C. (montacargas planta 0 carga paquete A 2 2 paquete B 3 8 nivel 0) (peso-maximo 40)
D. (montacargas planta 0) (carga paquete A 2 2 paquete B 3 8 nivel 0) (peso-maximo 40)

---

10) Dado el siguiente SBR, ¿cuántas reglas se insertarán en la agenda en el primer ciclo de inferencia?

```
(defrule R1
 (lista $?x1 ?y $?x2 ?y $?x3)
=>
 (assert (lista $?x1 ?y $?x3)))

(deffacts inicio
    (lista 2 3 1 2 3 2 1))
```

A. 4
B. 5
C. Ninguna
D. 3

---

11) Dado el siguiente SBR, indica cuál de las siguientes respuestas es CORRECTA:

```
(defrule R1
      (declare (salience 100))
```

```
        ?f <- (lista $?x ?y)
              (test (> ?y 5))
=>
        (retract ?f)
        (assert (lista $?x)))

(defrule R2
    (declare (salience 200))
        ?f <- (lista ?y $?x)
        (test (> ?y 5))
=>
        (retract ?f)
        (assert (lista $?x)))

(deffacts inicio
        (lista 3 7 1 5 9))
```

A. Sólo en el caso de que la estrategia de la agenda sea anchura, se ejecutará en primer lugar una instancia de R1
B. Sólo en el caso de que la estrategia de la agenda sea profundidad, se ejecutará en primer lugar una instancia de R2
C. Se ejecutará una instancia de R1 en primer lugar en cualquier caso
D. Se ejecutará una instancia de R2 en primer lugar en cualquier caso

---

12) Dado el siguiente hecho (pila A B A A B B A pilaA pilaB), que representa el estado inicial de un SBR, donde se tiene una pila inicial con bloques A y B y el objetivo es separar dichos bloques en dos pilas, una con bloques A y otra con bloques B. Indica cuál de las siguientes reglas NO toma un bloque A de la pila inicial y lo mueve a la pila de bloques A, de manera que se pueda resolver el problema:

A.
```
(defrule mover-a-pila-A
        (pila $?x A $?y pilaA $?z)
=>
        (assert (pila $?x $?y pilaA A $?z)))
```

B.
```
(defrule mover-a-pila-A
        (pila $?x ?b $?y pilaA $?z)
        (test (eq ?b A))
=>
        (assert (pila $?x $?y pilaA A $?z)))
```

C.
```
(defrule mover-a-pila-A
        (pila $?x ?b $?y pilaA $?z)
        (test (eq ?b A))
=>
        (assert (pila $?x ?b $?y pilaA ?b $?z)))
```

D. (defrule mover-a-pila-A
(pila $?x ?b $?y pilaA $?z)
(test (eq ?b A))
=>
(assert (pila $?x $?y pilaA ?b $?z)))

---

13) En un almacén se tienen dos zonas: una de carga y otra de descarga. En cada zona, puede haber varias pilas (identificadas con valores de 1 a 5) de pallets de tipo A, B o C para ser cargados o descargados en los camiones. Sea el siguiente estado inicial:

(almacen zona carga pila 1 A B C pila 2 B C B pila 3 A zona descarga pila 4 A B A pila 5 B A B B A)

Indica cuál de los siguientes patrones NO se podría utilizar para almacenar únicamente en la variable ?p el identificador de una pila de la zona de carga cuyo primer pallet sea de tipo A:

A. (almacen zona carga $?c pila ?p A $?r zona descarga $?d)
B. (almacen zona carga $?c pila ?p A $?r)
C. (almacen $?c pila ?p A $?r zona descarga $?d)
D. (almacen $?c pila ?p A $?r 4 $?d)

---

14) Dada la BH inicial= {(elemento e) (lista e a e b c d e f)}, y el siguiente conjunto de reglas:

```
(defrule R1
   ; (declare (salience  10))
        (elemento ?e)
        (lista $?a ?e $?b)
=>
       (assert (lista ?e $?a $?b)))

(defrule R2
    ; (declare (salience  -30))
        (lista   ?a  $?x ?a)
        (elemento   ?a )
=>
       (assert   (lista $?x)))
```

¿Cuál de las siguientes afirmaciones es CORRECTA? (NOTA: el ; delante de los comandos (declare (salience …)) indica que están comentados)

A. El estado final dependerá de la estrategia de control aplicada (anchura, profundidad, coste uniforme, etc.)
B. En el SBR definido no llega a lanzarse ninguna regla.
C. Si se hubieran declarado prioridades en las reglas (salience …), el estado final dependería de dichas prioridades

D. El estado final será el mismo, cualquiera que sea el tipo de control que se aplique

15) Dada la siguiente parte izquierda de una regla

```
(defrule r1
      (lista $? ?x $? ?y)
      (test (< ?x ?y))
=>
       …
```

, y el siguiente hecho: (lista 1 3 2 1 3 6), ¿Cuántas instancias de esta regla se incluirán en la agenda?:

A. 0
B. 1
C. 5
D. Más de 5

---

16) Dada la BH={(lista1 b a a a c c a c b b c)(lista2 a c)} y la siguiente regla

```
(defrule r1
  ?f <- (lista1  $?x ?a ?a $?y)
     (lista2  $? ?a $?)
   =>
 (retract ?f)
 (assert (lista1 $?x ?a $?y)))
```

Indicad cuál será la Base de Hechos final.

A. {(lista1 b b b c) (lista2 a c)}
B. {(lista1 b a c a c b b c) (lista2 a c)}
C. {(lista1 b b b c)}
D. {(lista1 b a c a c b c) (lista2 a c)}

---

17) Dada la siguiente parte izquierda de una regla:

```
(defrule r2
   ?f <- (lista  $? ?b $?x ?b $?x)
    =>
   ...
```

y el hecho (lista c c d c c d c c d). ¿Cuántas instancias de esta regla se insertarían en la agenda?

A. 1
B. 2
C. 3

D. 4

18) Dada la base de hechos inicial: BH={(lista 5 7 3 1 6 4) (maximo 0)} y la siguiente regla para calcular el máximo de una lista

```
(defrule r4
    ?f1 <- (lista $?a ?b $?c)
    ?f2 <- (maximo ?x)
    (test (>  ?b ?x))
=>
    (assert (lista $?a $?c))
    (assert (maximo ?b)))
```

Si nuestro objetivo es obtener una base de hechos final (tras la ejecución sucesiva de la regla) en la cual el hecho 'maximo' solo puede aparecer una vez (conteniendo el valor máximo de la lista). ¿Cuál de las siguientes afirmaciones es CIERTA para obtener nuestro objetivo?

A. La regla es correcta
B. Sería necesario añadir (retract ?f1)
C. Sería necesario añadir (retract ?f2)
D. La modificación de C no bastaría ya que sería también necesario añadir (retract ?f1)

---

19) Dado el siguiente hecho: (problema torre a b c nombre A torre a nombre B torre nombre C) ¿Cuál de los siguientes patrones serviría para obtener el nombre de una torre con un único elemento en ella?

A. (problema $?x torre ?a $?y nombre ? $?z)
B. (problema $?x torre ?a nombre ?z $?x)
C. (problema $?x torre ?a nombre ?z $?)
D. (problema $? torre ?a nombre ?)

---

20) Sea el siguiente SBR para calcular el número de Fibonacci de un número n >0, por ejemplo n=5 (el número de Fibonnaci se calcula en fib-1), ¿cuál de las siguientes afirmaciones es correcta?

```
(defrule fact
  ?f1 <- (numero ?n1)
  ?f2 <- (fib-1 ?n2)
  ?f3 <- (fib-2 ?n3)
=>
  (retract ?f1 ?f2 ?f3)
  (assert (numero (- ?n1 1)))
  (assert (fib-1 (+?n2 ?n3)))
  (assert (fib-2 ?n2 )))

(deffacts fibonacci (numero 5) (fib-1 1) (fib-2 0))
```

NOTA: El número de Fibonnaci se calcula como: f(n)=f(n-1)+f(n-2) siendo f(0)=0 y f(1)=1.

A. El SBR funciona correctamente.
B. Lo anterior no es cierto, pues se requiere una condición (test (> ?n1 1)) en la premisa de la regla para que funcione correctamente.
C. La modificación anterior no bastaría, siendo además necesario una regla de parada: (defrule parada (declare (salience 100)) (numero 1) => (halt)).
D. Ninguna de las anteriores es cierta.

---

21) Sea el hecho:

(Mercado Calle 1 Fruta 20 Pescado 0 Calle 2 Fruta 10 Pescado 10 Calle 3 Fruta 16 Pescado 4)

donde el número que aparece después del símbolo 'Calle' indica el identificador de dicha calle en el Mercado, los números que aparecen después de los símbolos 'Fruta' o 'Pescado' indican el número de puestos de fruta o puestos de pescado en la calle correspondiente. ¿Cuál será el patrón adecuado para obtener únicamente el identificador de una calle cualquiera y el número de puestos de pescado de dicha calle?

A. (Mercado $? Calle ? Fruta $? Pescado ?n $?)
B. (Mercado $? Calle ?c $? Pescado ?n $?)
C. (Mercado Calle ?c Fruta ? Pescado ?n)
D. (Mercado $? Calle ?c Fruta ? Pescado ?n $?)

---

22) Sea un SBR formado por BHinicial={(lista 2 1 6 2 3)}, y las siguientes reglas:

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

```
(defrule R2
    ?f <- (lista $?x ?z ?y $?w)
    (test (> ?z ?y))
=>
 (assert (lista $?x ?z ?y $?w)))
```

¿Cuál sería el contenido del Conjunto Conflicto (Agenda) tras el primer pattern-matching?

A. Una instancia de la regla R1 y dos de la R2
B. Dos instancias de la regla R1 y una instancia de la regla R2
C. Dos instancias de la regla R1 y dos instancias de la regla R2
D. Ninguna instancia

---

23) Dado un SBR compuesto de la siguiente regla:

```
(defrule regla-1
   ?f <- (lista ?y $?x ?y $?x ?y)
=>
      (retract ?f)
```

```
(assert (lista $?x)))
```

, y la BH inicial {(lista 1 2 3 2 3 2 3 2 3 2 1 2 3 2 3 2 3 2 3 2 1)}, ¿cuál será el estado final de la Base de Hechos?

A. {(lista 3 2 3)}
B. {(lista 1) (lista)}
C. {(lista 2 1 2)}
D. {(lista 1)}

---

24) Sea el formato de patrón (lista [nombre^{s} edad^{s}]^{m}) para representar el nombre y la edad de un conjunto de personas. Dada una lista determinada de personas, se quiere contar el número de ellas cuya edad está comprendida entre 18 y 65 años. Para ello se dispone del hecho que representa la lista de personas, un hecho inicial (contador 0) para contar el número de personas y la regla que se muestra a continuación. Indica la opción CORRECTA:

```
(defrule contar
   ?f1 <-(lista $?x1 ?num $?x2)
   ?f2 <- (contador ?cont)
   (test (numberp ?num))   ;; numberp devuelve TRUE si ?num es un número
   (test (and (>= ?num 18)(<= ?num 65)))
 =>
   (retract ?f2)
   (assert (contador (+ ?cont 1))))
```

A. El SBR funciona correctamente.
B. Para que el SBR funcione correctamente es necesario añadir solo la instrucción (assert (lista $?x1 $?x2)) en la RHS de la regla.
C. Para que el SBR funcione correctamente es necesario añadir la instrucción (retract ?f1) y la instrucción (assert (lista $?x1 $?x2)) en la RHS de la regla.
D. Ninguna de las anteriores

---

25) Dada la BH={(lista b a a a c a c b b c) (lista1 a c d e f g)} y la siguiente regla, indica cuál será la BH final.

```
(defrule R1
  ?f <- (lista  $?x ?a ?a $?y)
     (lista1  $? ?a $?)
   =>
 (retract ?f)
 (assert (lista $?x ?a $?y)))
```

A. {(lista b b b c) (lista1 a c d e f g)}

B. {(lista b b b c)}
C. {(lista b a c a c b b c) (lista1 a c d e f g)}
D. {(lista b a c b b c) (lista1 a c d e f g)}

---

26) Dada la base de hechos inicial: BH={(lista 5 7 3 1 6 4) (minimo 9999)} y la siguiente regla para calcular el mínimo de una lista

```
(defrule REGLA
    ?f1 <- (lista $?a ?b $?c)
    ?f2 <- (minimo ?x)
    (test (<  ?b ?x))
=>
    (assert (lista $?a $?c))
    (assert (minimo ?b)))
```

Si nuestro objetivo es obtener una base de hechos final (tras la ejecución sucesiva de la regla) en la cual el hecho (minimo …) solo puede aparecer una vez (conteniendo el valor mínimo de la lista). ¿Cuál de las siguientes afirmaciones es CIERTA para obtener nuestro objetivo?

A. La regla es correcta
B. Sería necesario añadir (retract ?f2)
C. Sería necesario añadir (retract ?f1)
D. Sería necesario añadir (retract ?f1) y (retract ?f2)

---

27) Indica cuál es el resultado final CORRECTO tras ejecutar el siguiente SBR con la BH inicial={(lista 34 77 34)}:

```
(defule R1
   (declare (salience 25))
     ?f <- (lista $?x1 ?num $?x2)
  =>
    (retract ?f)
    (printout t "Mensaje 1" crlf))
```

```
(defrule R2
     (declare (salience 10))
     ?f <- (lista ?num $?x ?num)
     =>
(retract ?f)
     (printout t "Mensaje 2" crlf))
```

```
(defrule R3
 =>
(printout t "Mensaje 3" crlf))
```

A. Mostrará tres veces el "Mensaje 1".
B. Mostrará una vez "Mensaje 1" y una vez "Mensaje 2".
C. Mostrará una vez "Mensaje 1" y una vez "Mensaje 3".
D. Mostrará una vez "Mensaje 1", una vez "Mensaje 2" y una vez "Mensaje 3".

---

28) El siguiente hecho representa un conjunto de pilas y los bloques que contiene cada una de ellas. El número que aparece después del símbolo 'pila' es el identificador de la pila y a continuación aparecen los bloques de la pila, siendo el primer bloque el tope de la pila. Indica cuál sería la LHS de una regla para que se instancie CORRECTAMENTE el bloque que se encuentra en la base de cualquier pila del hecho que se muestra a continuación:

(problema pila 1 A F G J K pila 2 B D pila 3 C H I L pila 4)

A. (problema $?x1 pila ?num $?x2 ?y pila $?x3) (test (not (member pila $?x1)))
B. (problema $?x1 pila ?num $?x2 ?y pila $?x3) (test (not (member pila $?x2)))
C. (problema $?x1 pila ?num $?x2 ?y $?x3)(test (not (member pila $?x3)))
D. Ninguna de las anteriores.

---

29) Sea un SBR formado por BHinicial={(lista 2 1 5 3)}, y las siguientes reglas:

```
(defrule R1
    (declare (salience 100))
 ?f <- (lista $?x ?z ?y $?w)
    (test (< ?z ?y))
=>
 (assert (lista $?x ?z ?y $?w)))
```

```
(defrule R2
        (declare (salience 150))
 ?f <- (lista $?x ?z ?y $?w)
        (test (>= ?z ?y))
=>
 (assert (lista $?x ?z ?y $?w)))
```

```
(defrule final
 (declare (salience 200))
   (lista $?list)
=>
 (halt))
```

tras el primer *pattern-matching,* ¿cómo quedarían ordenadas las instancias el Conjunto Conflicto (Agenda)?

A. Una instancia de la regla R1, una instancia de la regla R2 y una instancia de la regla final
B. Dos instancias de la regla R2, una instancia de la regla R1, una instancia de la regla final
C. Una instancia de la regla final, dos instancias de la regla R2, una instancia de la regla R1.
D. Una instancia de la regla final

---

30) Sea un SBR formado por BHinicial={(lista 2 1 6 2 3) (elemento 5)}, y la regla que se muestra a continuación. ¿Cuál sería el contenido final de la BH?

```
(defrule REGLA
  ?f <- (lista $?x ?z $?w)
        (elemento  ?y)
        (test (< ?z ?y))
 =>
   (assert (lista $?x  $?w))
   (retract ?f))
```

A. {(lista 6) (elemento 5)}
B. {(lista 2 1 2 3) (elemento 5)}
C. {(lista) (elemento 5)}
D. {(lista 2 2) (elemento 5)}

31) Sea la siguiente regla para calcular el Máximo Común Divisor (mcd) de dos números enteros positivos. Indica la respuesta **CORRECTA**:

```
(defrule mcd
    ?a <- (num ?n1)
    ?b <- (num ?n2)
    (test (> ?n1 ?n2))
=>
    (retract ?a)
    (assert (num (- ?n1 ?n2))))
```

A. Calcula correctamente el mcd quedando un hecho 'num' con dicho valor
B. Es necesario añadir una regla parada sin prioridad para que el sistema no entre en una ejecución sin fin
C. Es necesario añadir una regla parada con prioridad para que el sistema no entre en una ejecución sin fin
D. Ninguna de las anteriores es correcta

---

32) Dado un SBR cuya Base de Hechos inicial es (lista b a c c a b b a resto), y una única regla:

```
(defrule pasar
    ?a <- (lista $?x ?y ?y $?x $?z resto $?m)
=>
    (retract ?a)
    (assert (lista $?x $?x $?z resto $?m ?y)))
```

El contenido final de la Base de Hechos será:

A. (lista b a a b b a resto c)
B. (lista b a resto c a b)
C. (lista b a b a resto c b)
D. Ninguna de las anteriores

---

33) Dado el siguiente hecho: (Dueños coches a b c dueño P coches d dueño Q coches e f dueño R), donde se relaciona los coches y posteriormente su dueño ¿Cuál de los siguientes patrones serviría para obtener el nombre del dueño de un solo coche?

A. (Dueños $?x coches ?a dueño $?z)
B. (Dueños $? coches ? dueño ?z $?)
C. (Dueños $?x coches ?a dueño ?z $?x)
D. (Dueños $? coches ? dueño ?z)

---

34) Sea un SBR, con la BH inicial {(lista A B C A B C C B A C B A)}, y la siguiente regla:

```
(defrule regla1
    ?f1 <- (lista $?x1 ?y $?x2 ?y $?x3)
    (test (> (length $?x2) 0))
    (test (not (member ?y $?x2)))
=>
     (retract ?f1)
     (assert (lista $?x1 ?y ?y $?x3)))
```

Si se ejecuta este SBR, el resultado es:

A. Una lista que solo contiene letras A
B. Una lista que solo contiene letras B
C. Una lista que solo contiene letras C
D. Dependerá de la estrategia de control que se aplique, sea anchura o profundidad.

---

35) Dado el hecho (prueba 1 2 3 4 5 6 7 8 9 10) y la regla :

```
(defrule regla1
    ?f1 <-  (prueba $?a  $?c)
=>
    (retract ?f1)
    (assert (lista $?c)))
```

En el primer ciclo inferencial:

A. No se producirá ninguna instanciación
B. Se producirán 9 instanciaciones
C. Se producirán 10 instanciaciones
D. Se producirán 11 instanciaciones

---

36) Dada la base de hechos inicial: BH={(lista 3 5 2 5 3 4 2 9 8 8 9 6) (numero 5) (repeticiones 0)} y la siguiente regla para calcular el número de repeticiones de un elemento de una lista de números naturales

```
(defrule REGLA
    ?f1 <- (lista $?a ?b $?c)
    ?f2 <- (numero ?x)
    ?f3 <- (repeticiones ?z)
    (test (= ?b ?x))
=>
     (assert (lista $?a $?c))
     (assert (repeticiones (+ 1 ?z))))
```

Si nuestro objetivo es obtener una base de hechos final (tras la ejecución sucesiva de la regla) en la cual el hecho (repeticiones …) solo puede aparecer una vez (conteniendo el número de repeticiones del número indicado en el patrón (numero ..) en la lista). ¿Cuál de las siguientes afirmaciones es CIERTA para obtener nuestro objetivo?

A. La regla es correcta
B. Sería necesario añadir (retract ?f1)
C. Sería necesario añadir (retract ?f1) y (retract ?f3)
D. Sería necesario añadir (retract ?f3)

---

37) Sea un SBR formado por BHinicial={(lista 2 1 5 3)}, y las siguientes reglas:

```
(defrule R1
   (declare (salience 200))
      ?f <- (lista $?x ?z ?y $?w)
      (test (< ?z ?y))
 =>
      (assert (lista $?x ?z ?y $?w)))
```

```
(defrule R2
    (declare (salience 50))
        ?f <- (lista $?x ?z ?y $?w)
        (test (>= ?z ?y))
 =>
     (assert (lista $?x ?z ?y $?w)))
```

```
(defrule final
  (declare (salience 150))
      (lista $?list)
 =>
     (halt))
```

tras el primer *pattern-matching,* ¿como quedarían ordenadas las instancias el Conjunto Conflicto (Agenda)?

A. Una instancia de la regla R1, una instancia de la regla final, y dos instancias de la regla R2
B. Dos instancias de la regla R2, una instancia de la regla R1, una instancia de la regla final
C. Una instancia de la regla final, dos instancias de la regla R2, una instancia de la regla R1
D. Una instancia de la regla final

---

38) Dada la siguiente parte izquierda de una regla

```
(defrule r1
      (lista $?x $?w ?y ?z $?x)
=>
      ….
```

, y el siguiente hecho: (lista a b a b c c), ¿cuántas instancias de esta regla se incluirán en la agenda?:

A. 0
B. 1
C. 3
D. 5

PROBLEMAS

Ejercicio 1

Un chimpancé hambriento se encuentra dentro de una habitación. En el techo de la estancia, justamente en el punto central, hay colgado un racimo de plátanos. La distancia entre el extremo inferior del racimo y el suelo hace imposible que el chimpancé alcance los plátanos. Dentro de la habitación, distribuidas al azar, hay cinco cajas cada una de un tamaño. Sólo si se apilan todas las cajas de mayor a menor queda el mínimo espacio para que el chimpancé pueda coger los plátanos. El problema para el chimpancé es determinar la secuencia de acciones que le permiten satisfacer su apetito. Diseñar un SBR que ayude al chimpancé a determinar la secuencia de acciones para apilar las cajas.

1. Especifica la estructura de la Base de Hechos. Muestra un ejemplo de BH inicial y la BH final.
2. Escribe las reglas de producción necesarias para resolver el problema.

Solución:

Para representar el estado utilizaremos el siguiente patrón:

(HABITACION APILADAS x_1 x_2 … x_5 100 DESAPILADAS y_1 y_2 … y_5)

donde: $x_i \in \{1,2,3,4,5\}$
$y_i \in \{1,2,3,4,5\}$

y el patrón (FINAL x) para controlar la parada / x ∈ {SI,NO}

Los símbolos numéricos 1, 2, 3, 4 y 5 representan cada una de las cajas; concretamente indican el tamaño de cada caja (1: caja más pequeña, 5: caja más grande). El símbolo 100 que aparece después de la representación de las cajas apiladas sirve para denotar el 'fin' de las cajas apiladas, es decir, representa la base de la pila.

El estado inicial vendrá representado por los hechos:

(FINAL NO)
(HABITACION APILADAS 100 DESAPILADAS 1 5 3 2 4)

o cualquier otra secuencia de las cajas [1,…, 5] desapiladas.

El estado final vendrá representado por los hechos:

(FINAL SI)
(HABITACION APILADAS 1 2 3 4 5 100 DESAPILADAS)

La BH (o Working Memory –WM) estará constituida por el siguiente conjunto de reglas:

```
(defrule apilar
      (FINAL NO)
      ?f <- (HABITACION APILADAS ?x $?y DESAPILADAS $?z ?z1 $?z2)
      (test (< ?z1 ?x))
=>
      (assert (HABITACION APILADAS ?z1 ?x $?y DESAPILADAS $?z $?z2)))
```

```
(defrule desapilar
      (FINAL NO)
      ?f <- (HABITACION APILADAS ?x $?y DESAPILADAS $?z ?z1 $?z2)
      (test (> ?z1 ?x))
=>
      (assert (HABITACION APILADAS $?y DESAPILADAS $?z ?z1 $?z2 ?x)))


(defrule final
      ?f <- (FINAL NO)
      (HABITACION APILADAS 1 2 3 4 5 100 DESAPILADAS)
=>
     (retract ?f)
        (assert (FINAL SI))
        (printout t "Solucion encontrada " crlf))
```

En este caso no es necesario asociar una mayor prioridad a las reglas 'apilar' y 'desapilar' porque si éstas no se activan es que las cajas ya están apiladas en el orden correcto.

Ejercicio 2

Un robot se encuentra en una habitación donde hay un conjunto de cajas dispersas por el suelo que debe apilar en dos columnas distintas. Cada caja lleva una etiqueta que indica la columna donde debe ser apilada. Las restricciones del problema son:

- el robot puede apilar en cada movimiento hasta un máximo de 3 cajas si son de la misma columna o bien una caja en cada columna,
- la diferencia de cajas de ambas pilas nunca debe ser mayor que 2. Por ejemplo, si en PILA1 hay 3 cajas y en PILA2 hay 2 cajas entonces el robot no podrá llevar dos cajas a PILA1 (asumimos que el conjunto de cajas total existente cumple esta restricción).

Diseñar un SBR que permita resolver el problema. Los elementos necesarios son:

a) Estructura de la Base de Hechos; especificar un ejemplo de BH-inicial y BH-final.
b) Reglas de producción necesarias.

Solución:

Patrón:

(robot cajas 1 x^s 2 y^s pila 1 z^s 2 w^s) donde

$x^S \in$ INTEGER indica el número de cajas que hay en el suelo que se tienen que apilar en la pila 1

$y^S \in$ INTEGER indica el numero de cajas que hay en el suelo que se tienen que apilar en la pila 2

$z^S \in$ INTEGER es el numero de cajas de la pila1

$w^S \in$ INTEGER es el numero de cajas de la pila2

BHinicial= {(robot cajas 1 4 2 5 pila 1 0 2 0)} /* hay 4 cajas de la pila1 y 5 cajas de la pila 2*/
BHfinal= {(robot cajas 1 0 2 0 pila 1 4 2 5)}

De acuerdo a la representación escogida, lo más cómodo es plantear las siguientes reglas:

- mover-X-cajas-a-pila1
- mover-X-cajas-a-pila2
- mover-1-caja-a-cada-pila

aunque mediante variables multicampo podría emplearse una única regla para mover X cajas a cualquier columna. Sin embargo, la representación de la LHS de la regla sería bastante más complicada.

```
(defrule mover-X-cajas-a-pila1              ;;de pila1 a pila2
      (mover $?y ?x $?z)
      (robot cajas 1 ?c1 2 ?c2 pila 1 ?p1 2 ?p2)
      (test (and (> ?c1 0)(<= ?x ?c1)))
      (test (<= (- (+ ?p1 ?x) ?p2) 2))))
=>
       (assert (robot cajas 1 (- ?c1 ?x) 2 ?c2 pila 1 (+ ?p1 ?x) 2 ?p2)))


(defrule mover-1-caja-a-cada-pila
      (robot cajas 1 ?x cajas 2 ?y pila 1 ?px 2 ?py)
      (test (and (>= ?x 1) (>= ?y 1)))
=>
      (assert (robot cajas 1 (- ?x 1) 2 (- ?y 1) pila 1 (+ ?px 1) 2 (+ ?py 1)))


(defrule final
     (declare (salience 100))
       (robot cajas 1 0 2 0 $?)
 =>
      (halt)
      (printout t "Solution found " crlf))
```

Ejercicio 3

Dada una BH que represente una lista de naturales no ordenados, escribir una única (si es posible) regla de producción que obtenga, como BH-meta, la lista inicial con sus elementos ordenados. Por ejemplo:

BH inicial: (lista 4 5 3 46 12 10)
BH final: (lista 3 4 5 10 12 46)

Determinar además las instancias de dicha regla que resultan aplicables sólo para la BH-inicial del ejemplo.

Solución:

```
(defrule ordenar
      ?a <- (lista $?x ?y ?z $?w)
      (test (< ?z ?y))
=>
```

```
        (retract ?a)
        (assert (lista $?x ?z ?y $?w)))
```

Instancias aplicables con la BH-inicial:

1) ?y=5, ?z=3
2) ?y=46, ?z=12
3) ?y=12, ?z=10

Ejercicio 4

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 (asumir que existe una variable global inicializada a 0 que sirve para contar los aciertos). Ejemplo de BHinicial:

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

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

Solución:

```
(defglobal ?*num-aciertos* = 0)

(defrule cuenta-aciertos
  (boleto-ganador $?x ?y $?z)
  (combinacion $?a ?y $?b)
=>
 (bind ?*num-aciertos* (+ ?*num-aciertos* 1))
 (printout t "El numero de aciertos hasta el momento es " ?*num-aciertos* crlf))


(deffacts datos
   (boleto-ganador 2 4 8 12 22 34)
   (combinacion 42 13 22 30 2 5))
```

Sí, el SBR funcionaría de igual modo porque lo que hace la regla es detectar cualquier combinación de dos números iguales de los hechos (boleto-ganador ……) y (combinacion ……).

Ejercicio 5

Una familia que está a un lado de un puente está compuesta de 5 miembros. Cada uno de los miembros puede cruzar el puente, en uno u otro sentido, a una velocidad distinta. Debido a que es de noche, debe usarse una linterna para cruzar. Existe una única linterna, la cual tiene una duración límite de 30 segundos. En cada cruce del puente, la vida útil de la linterna se va reduciendo y no se puede recargar. La linterna solo se gasta al cruzar el puente. El puente solo resiste un máximo de 2 personas, y un par de personas cruza el puente a la velocidad del más lento. Las duraciones de cruce de cada miembro son: (Abuelo: 12 seg., Padre: 8 se., Madre: 6 seg, Hija: 3 seg, Hijo: 1 seg.). Por ejemplo:

Estado-Inicial: Todos a la derecha
Acción-1: Pasa el abuelo e Hijo a la izquierda, con la linterna (se gasta 12 seg).
Acción-2: Vuelve el Hijo a la derecha con la linterna (se gasta 1 seg)
Acción-3: Pasa el Padre y el Hijo con la linterna (se gasta 8 seg)

Diseñar un sistema basado en reglas, para una búsqueda en grafo, que obtenga la secuencia de acciones (cruces del puente), si existe, que permitiría a toda la familia pasar de un lado a otro del puente. La única acción permitida es la que se refiere a *cruzar* (en un sentido y/o en otro, una y/o dos personas, etc.).

a) Diseñar la base de hechos mediante patrones. Indicar la base de hechos inicial y base de hechos final.

b) Establecer las reglas correspondientes. En la parte derecha de las reglas solo pueden haber expresiones assert, retract y printout; y en la parte izquierda expresiones de pattern-matching y "test".

Solución:

1) Base de Hechos

Una posibilidad es:

(familia Abuelo derecha| izquierda 12 Padre derecha| izquierda 8 Madre derecha| izquierda 6
Hijo derecha| izquierda 1 Hija derecha| izquierda 3 linterna derecha| izquierda x),

donde x={0..30}

También podría haberse separado la información (estática) de los tiempos de trayecto en un patrón separado.

Base de hechos inicial: (familia Abuelo derecha 12 Padre derecha 8 Madre derecha 6 Hijo derecha 1 Hija derecha 3 linterna derecha 30)

Base de hechos meta: (familia Abuelo izquierda 12 Padre izquierda 8 Madre izquierda 6 Hijo izquierda 1 Hija izquierda 3 $?A)

2) Reglas (incorporando el nivel):

(deffacts problema-puente
(familia Abuelo derecha 12 Padre derecha 8 Madre derecha 6 Hijo derecha 1 Hija derecha 3 linterna derecha 30 nivel 0))

(defrule cruzar-I-2-PERSONAS
?f1 <- (familia $?A ?miembro1 derecha ?x $?B ?miembro2 derecha ?y $?C linterna derecha ?z nivel ?n)
(test (>= ?z (max ?x ?y)))
=>
(assert (familia $?A ?miembro1 izquierda ?x $?B ?miembro2 izquierda ?y $?C linterna izquierda (- ?z (max ?x ?y)) nivel (+ ?n 1))))

(defrule cruzar-D-1PERSONA
?f1 <- (familia $?A ?miembro izquierda ?x $?C linterna izquierda ?z nivel ?n)
    (test (>= ?z ?x))
=>
(assert (familia $?A ?miembro derecha ?x $?C linterna derecha (- ?z ?x) nivel (+ ?n 1))))

(defrule meta
(declare (salience 100))
(familia Abuelo izquierda ? Padre izquierda ? Madre izquierda ? Hijo izquierda ? Hija izquierda ? linterna ? ?z nivel ?n)
=>
(printout T "Meta Alcanzada en nivel " ?n " queda en linterna " ?z crlf)
(halt))

Nota: Este diseño, adecuado al objetivo deseado, es uno de los distintos diseños alternativos posibles. No sería adecuado plantear una regla para cada combinación posible de paso.

Entre otras, una posible solución en el nivel 7, con un consumo de 29 seg es: (Izq: Hijo, Hija), (der: Hija) (Izq: Abuelo Padre) (der: hijo) (Izq: hijo hija) (der: hijo) (Izq: madre hijo)

Ejercicio 6

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

```
(defrule R1                          (defrule R2
   ?f <- (lista ?x ?y $?z)              ?f <- (lista ?y $?x)
   (test (< ?x ?y))                        (test (and (<= (length $?x) 3) (>= (length $?x) 1)))
   =>                                 =>
   (retract ?f)                         (retract ?f)
   (assert (lista ?y ?z))                  (assert (lista ?x)))
```

asumiendo una estrategia de búsqueda en anchura (mayor prioridad para las instancias más antiguas, comenzando por R1), ¿cuál sería el estado final de la BH?. Realiza una traza que muestre el proceso inferencial.

Solución:

Base de Hechos (BH)	**Conjunto Conflicto (CC)**
H1: (lista 3 6 8 5 10) (1*)	R1: H1 (1*)
H2: (lista 6 8 5 10) (2*)	R1: H2 (2*) R2: H2 (se elimina al eliminar H2)
H3: (lista 8 5 10) (3*)	R2: H3 (3*)

H4: (lista 5 10) (4*)	R1: H4 (4*) R2: H4 (se elimina al eliminar H4)
H5: (lista 10)	No hay nuevas instancias

Nota: El número entre paréntesis indica el orden de disparo de la regla y la eliminación del hecho correspondiente como consecuencia del disparo de la regla. (*) indica que el hecho o la instancia de la regla se eliminan de la BH o CC respectivamente.

BHfinal= {(lista 10)}

Ejercicio 7

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):

a) ¿cuál sería el estado final de la BH?. Realiza una traza que muestre el proceso inferencial.
b) Si la BH inicial fuera BH={(lista 1 2 2 4)}, ¿cuántos hechos (lista-new ……) habría en la BH final ?. ¿Cuáles?.
c) 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 ……) ?
d) 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 ……) ?

Solución:

a)

Base Hechos (BH)	**Conjunto Conflicto (CC)**
H1: (lista 1 2 3 4)(1*)	R1,H1 (1*)
H2: (lista 2 3 4) (2*) H3: (elemento 1) (4*)	R1, H2 (2*)
H4: (lista 3 4) (3*) H5: (elemento 2) (6*)	R1, H4 (3*) R2, H3, H5 (4*)
H6: (lista 4) (5*)	R1, H6 (6*)

H7: (elemento 3) (7*) — R2, H3, H7 (5*, se elimina al eliminar H3)
R2, H5, H7 (7*)

==========

H8: (lista-new 1 2)

==========

H9: (lista) — R2, H5, H10 (8*, se elimina al eliminar H5)
H10: (elemento 4) — R2, H7, H10 (9*)

==========

H11: (lista-new 2 3)

==========

H12: (lista-new 3 4)

Nota: (*) indica que el hecho o la instancia de la regla se eliminan de la BH o CC respectivamente.

BHfinal= {(lista), (elemento 4), (lista-new 1 2), (lista-new 2 3), (lista-new 3 4)}

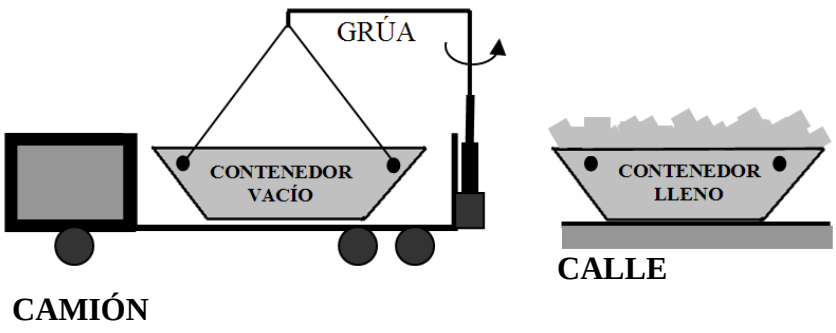
b) Asumiendo que no se permite la duplicidad de hechos, los hechos lista-new que habría son:
(lista-new 1 2), (lista-new 2 4)

c) Ninguno. Al no eliminar el hecho (lista …) que ha provocado la instanciación de R1 no se produce ningún cambio porque dicho hecho no volverá a instanciar de nuevo regla-1 (no se puede producir una instanciación de una misma regla, con un mismo hecho asignando los mismos valores a las variables del patrón).

d) Se producirían todas las combinaciones posibles con los hechos (lista-new …). Esto es,
(lista-new 1 2) **(lista-new 1 3) (lista-new 1 4)** (lista-new 2 3) **(lista-new 2 4)** (lista-new 3 4). (los hechos que están en negrita serían los nuevos hechos que aparecerían si se eliminara el comando (retract ?f) de R2.

Ejercicio 8

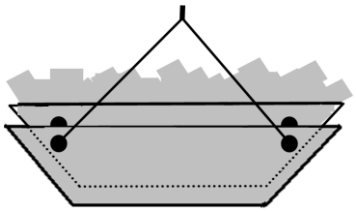
Un Camión-Grúa es utilizado para recoger y reemplazar los contenedores llenos de residuos de la construcción que están situados en la calle: se lleva los contenedores llenos y deja en su lugar uno vacío. Para ello, el camión lleva un contenedor vacío y una grúa. La grúa puede engancharse a un contenedor (el cual puede estar lleno de escombros, vacío, o llevar otro contenedor apilado dentro), cogerlo y colocarlo en el camión o en la calle. No hay sitio para dejar dos contenedores directamente (es decir, no apilados) sobre la calle, o sobre el camión.



Las acciones posibles de la grúa son:

Coger-Contenedor: La grúa coge un contenedor C, que puede estar vacío, lleno, o con otro contenedor dentro (ver figura). El contenedor puede estar en el camión o en la calle, y la grúa debe estar libre.

Dejar-Contenedor: La grúa deja el contendor C que tenga cogido (y que puede estar vacío, lleno, o con otro contenedor dentro) en un lugar (calle o camión). Tanto en la calle, como en el camión, ***solo hay sitio para un contenedor*** (salvo que estén apilados como se indica en la figura).



Apilar-Contenedor: La grúa apila el contenedor C1 (lleno o vacío) que tiene cogido, dentro del contenedor C2, que debe estar necesariamente vacío. La debe estar grúa libre y el contenedor C2 puede estar tanto en la calle, como en el camión.

Desapilar-Contenedor: La grúa desapila (coge) el contenedor C2 (lleno o vacío) que está apilado dentro del contenedor C1, el cual quedará vacío. La grúa debe estar libre y los contenedores apilados pueden estar tanto en la calle, como en el camión.

No se pueden establecer otro tipo de acciones que las indicadas. Las operaciones de 'Asir-Contenedor'/'Soltar-Contenedor', que se refieren a atar/desatar los cables de la grúa a un contenedor, no hace falta considerarlas, pues se asumen incluidas en las operación necesarias.

Asumiendo la situación inicial ***"El Camión-Grúa lleva un contenedor vacío (Contenedor-1) y existe un contenedor lleno (Contenedor-2) en la calle"***, especificar en CLIPS un sistema de producción para obtener la secuencia de acciones necesarias para obtener el objetivo: ***"El Camión-Grúa lleva el contenedor lleno y queda el contenedor vacío en la calle"***.

1. Especificar la Base de Hechos ***mediante patrones***.
2. Indicar la Base de Hechos inicial y la Base de Hechos objetivo.
3. Especificar las Reglas de Producción.

No utilizar más tipos de acciones que las indicadas, ni poner código procedural en la parte derecha de las reglas.
Las reglas deben ser ***suficientemente generalistas*** (tantas reglas como acciones posibles) y no deben ser particularizadas para actuar sobre un contenedor determinado o considerar casos concretos

Solución

Especificación de la Base de Hechos:

```
(grua libre|ocupado  camion libre|ocupado  calle libre|ocupado
        contenedor  Contenedor1  camion|grua|calle  vacio|C2
        contenedor  Contenedor2  camion|grua|calle|C1  lleno)
```

Base de Hechos inicial:

```
(grua vacia camion ocupado calle ocupado contenedor
Contenedor1 camion vacio contenedor Contenedor2 calle lleno)
```

Base de Hechos Objetivo:

(grua $?x contenedor ?C1 camion lleno $?y)
(grua $?z contenedor ?C2 calle vacio $?q)

La especificación en CLIPS de los hechos y reglas sería:

(deffacts inicio
(grua vacia camion ocupado calle ocupado contenedor Contenedor1 camion vacio contenedor Contenedor2 calle lleno))

(defrule coger-contenedor
?f1 <- (grua vacia $?x ?lugar ocupado $?y contenedor ?C1 ?lugar ?lleva $?z)
=>
(assert (grua llena $?x ?lugar libre $?y contenedor ?C1 grua ?lleva $?z)))

(defrule desapilar-contenedor
(grua vacia $?x contenedor ?C1 ?lugar ?C2 contenedor ?C2 ?C1 ?lleva)
=>
(assert (grua llena $?x contenedor ?C1 ?lugar vacio contenedor ?C2 grua ?lleva)))

(defrule dejar-contenedor
(grua llena $?x ?lugar libre $?y contenedor ?C1 grua ?lleva $?z)
=>
(assert (grua vacia $?x ?lugar ocupado $?y contenedor ?C1 ?lugar ?lleva $?z)))

(defrule apilar-contenedor
(grua llena $?x contenedor ?C1 ?lugar vacio contenedor ?C2 grua ?lleva)
=>
(assert (grua vacia $?x contenedor ?C1 ?lugar ?C2 contenedor ?C2 ?C1 ?lleva)))

(defrule objetivo
(declare (salience 100))
(grua $?x contenedor ?C1 camion lleno $?y)
(grua $?z contenedor ?C2 calle vacio $?q)
=>
(halt))

Ejercicio 9

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).

Solución:

```
(deffacts lista (lista 1 2 4 6 5 3 7 8 9))          ;Es un ejemplo de lista

(defrule uno
      ?f <- (lista $?x ?v $?y)
            (and  (test (<> ?v 0))          ;Debe controlarse, para que pueda acabar!!
                  (test (<> (length $?x) (- ?v 1))))

=>
      (retract ?f)
      (assert (lista $?x 0 $?y)))
```

Ejercicio 10

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)
BH Final: (lista 2 7 6 5 4 3 1 8)

Solución:

Esta es una posible regla. Hay otras alternativas.

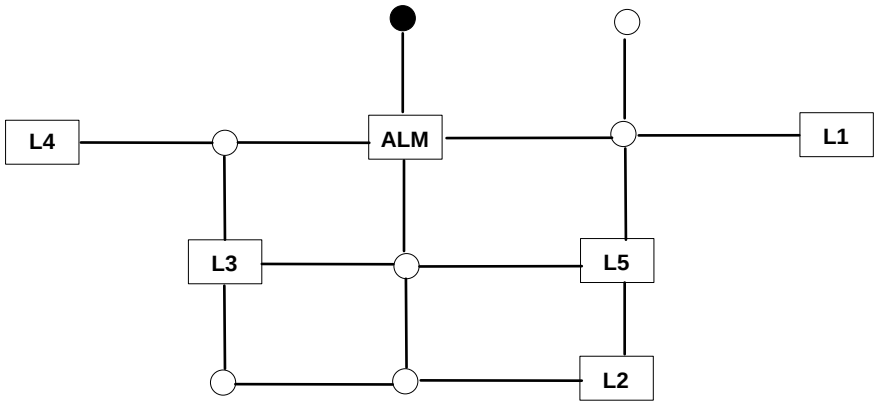
```
(defrule examen
   ?f1 <- (lista $?a ?x $?b ?x $?c)
   =>
   (retract ?f1)
   (assert (lista ?x ?a ?b ?c)))


(deffacts hechos
      (lista 1 3 4 5 6 7 4 6 4 4 3 4 5 6 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))
```

Ejercicio 11

Un robot tiene que suministrar bebidas a las peticiones que se hacen desde 5 puestos diferentes. Para ello el robot utiliza un carro que es parte inseparable de su anatomía y que tiene una capacidad para 3 botellas.

Existe una red de conexiones como la que se indica en la figura, donde L1,…,L5 son los puestos que solicitan botellas, ALM es el almacén donde se guardan las botellas que luego se repartirán y el resto son puntos intermedios donde el robot puede encontrase en un momento dado. El robot tiene que desplazarse físicamente hasta el puesto donde va a atender la solicitud; puede desplazarse entre cualquier par de localizaciones que se indican en la figura. En cada acción de movimiento se desplaza a una localización que esté conectada con su posición actual.



En el almacén (ALM) el robot puede llenar el carro de bebidas; para facilitar las cosas supondremos que siempre que el robot reponga existencias lo hará hasta completar su capacidad.

Un estado inicial del problema será: el robot se encuentra en una localización de la figura con el carro de bebidas vacío, y desde uno o varios puestos se solicitan un número de botellas. El número de botellas que solicita cada puesto no está limitado y el robot debe suministrar el total de botellas solicitado por un puesto en una sola entrega o hasta el final de existencias en el carro de bebidas. La situación final ha de ser que todas las peticiones de bebidas hayan quedado satisfechas, el robot debe estar de nuevo en el almacén y con el carro de bebidas lleno.

Ejemplo de situación inicial:

- El robot se encuentra en el punto sombreado que se indica en la figura
- El puesto L1 solicita dos botellas, L3 solicita una botella y L5 solicita cinco botellas

Diseña un Sistema de Producción que resuelva este problema para cualquier número de solicitudes y posición del robot:

- Especificar la estructura de los hechos que se van a emplear.
- Especificar el conjunto de reglas, en base a la representación elegida.

Solución

En el diseño del patrón se ha incluido la información de 'nivel' para poder mostrar el nivel del nodo solución, esto es, la longitud de la solución. También se incluye la variable ?*nod-gen* que almacena el número de nodos expandidos o reglas disparadas.

```
(defglobal ?*prof* = 20)
(defglobal ?*nod-gen* = 0)

(deffacts inicio
    (conexion Almacen p1)
    (conexion Almacen p4)
    (conexion Almacen p5)
    (conexion p2 p4)
    (conexion p4 L1)
    (conexion p3 Almacen)
```

```
    (conexion p3 L4)
    (conexion p5 L3)
    (conexion p3 L3)
    (conexion p5 L5)
    (conexion p4 L5)
    (conexion p8 L3)
     (conexion p8 p7)
     (conexion p5 p7)
     (conexion p7 L2)
     (conexion L5 L2)
    (robot situacion p1 botellas 0 solicitudes sol L1 2 sol L3 1 sol L4 5 nivel 0))

(defrule carga-robot
    (robot situacion Almacen botellas ?b $?x nivel ?n)
    (test (< ?b 3))
    (test (< ?n ?*prof*))
    =>
    (assert (robot situacion Almacen botellas 3 ?x nivel (+ 1 ?n)))
    (bind ?*nod-gen* (+ ?*nod-gen* 1)))



(defrule servir-peticion-completa
    (robot situacion ?s botellas ?b solicitudes $?x sol ?s ?bs $?y nivel ?n)
    (test (>= ?b ?bs))
    (test (< ?n ?*prof*))
    =>
    (assert (robot situacion ?s botellas (- ?b ?bs) solicitudes ?x ?y nivel (+ 1 ?n)))
    (bind ?*nod-gen* (+ ?*nod-gen* 1)))


(defrule servir-peticion-parcial
    (robot situacion ?s botellas ?b solicitudes $?x sol ?s ?bs $?y nivel ?n)
    (test (< ?b ?bs))
    (test (> ?b 0))
    (test (< ?n ?*prof*))
    =>
    (assert (robot situacion ?s botellas 0 solicitudes ?x sol ?s (- ?bs ?b) ?y nivel (+ 1 ?n)))
    (bind ?*nod-gen* (+ ?*nod-gen* 1)))


(defrule desplazarse
    (robot situacion ?sit $?x nivel ?n)
    (or (conexion ?sit ?dest)(conexion ?dest ?sit))
    (test (< ?n ?*prof*))
    =>
    (assert (robot situacion ?dest ?x nivel (+ 1 ?n)))
    (bind ?*nod-gen* (+ ?*nod-gen* 1)))


(defrule objetivo
    (declare (salience 100))
```

```
    (robot situacion Almacen botellas 3 solicitudes nivel ?n)
    =>
    (printout t "Solucion encontrada en el nivel: " ?n crlf)
    (printout t "NUMERO DE NODOS EXPANDIDOS O REGLAS DISPARADAS " ?*nod-gen* crlf)
    (halt))


(deffunction inicio ()
    (reset)
    (printout t "Profundidad Maxima:= " )
    (bind ?*prof* (read))
    (printout t "Tipo de Busqueda " crlf "    1.- Anchura" crlf "    2.- Profundidad" crlf )
    (bind ?a (read))
    (if (= ?a 1)
        then    (set-strategy breadth)
        else   (set-strategy depth))
    (printout t " Ejecuta run para poner en marcha el programa " crlf))
```

Ejercicio 12

Un cartón de bingo se compone de 5 líneas de 5 números cada una. La BH de un SBR 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.

Solución:

Patrón para el cartón: (carton linea $x1^m$ linea $x2^m$ linea $x3^m$ linea $x4^m$ linea $x5^m$)
x1, x2, x3, x4, x5 $\in$ INTEGER

```
(defrule bingo
   (linea-ganadora $?x)
   (carton $?a linea $?x $?b)
 =>
   (printout t "Linea ganadora " crlf))


(deffacts datos
    (linea-ganadora 12 21 34 56 77)
    (carton linea 1 2 3 4 5 linea 23 44 55 66 77 linea 12 21 34 56 77 linea 6 7 8 9 10 linea 18 28 38 48 58))
```

Ejercicio 13

Sea un SBR cuya BH inicial es BH={(elem e)(lista a b c d e f)}, y cuya Base de Reglas se compone de las reglas R1 y R2.

```
(defrule R1                          (defrule R2
   ?l1 <- (elem ?e)                     ?l1 <- (cambio)
   ?l2 <- (lista $?a ?e $?b)                 (lista $?l)
         =>                                    =>
   (assert (lista ?e ?a ?b))            (retract ?l1))
   (assert (cambio)))
```

asumiendo una estrategia en anchura (mayor prioridad para los hechos e instancias de reglas más antiguas, comenzando por R1), ¿cuál será el estado de la BH final?. Realiza una traza que muestre el proceso inferencial.

Solución

Base Hechos (BH)	Conjunto Conflicto (CC)
H1: (elem e) H2: (lista a b c d e f)	R1: H1, H2 (?e=e, $?a=a b c d, $?b=f)
H1, H2 H3: (lista e a b c d f) H4: (cambio)	R1: H1, H3 (?e=e, $?a={} $?b=a b c d f) R2: H4, H2 ($?l= a b c d e f) R2: H4, H3 ($?l= e a b c d f)
H1, H2, H3, H4 no se generan los hechos de los comandos assert porque son los mismos que H3 y H4	R2: H4, H2 ($?l= a b c d e f) R2: H4, H3 ($?l= e a b c d f) – se elimina al eliminar la instancia anterior el hecho H4 -
H1, H2, H3	

BH final= {H1: (elem e), H2: (lista a b c d e f), H3: (lista e a b c d f)}

Ejercicio 14

Las “matrioskas” son unas muñecas tradicionales rusas cuya originalidad consiste en que se encuentran huecas por dentro de modo que en su interior albergan una nueva muñeca de tamaño menor, y ésta a su vez otra de tamaño menor, hasta un número determinado de muñecas.

Tenemos un número N de muñecas. Cada una de las N muñecas tiene un distinto tamaño, de modo que una muñeca de tamaño **t** cabe en cualquier muñeca de tamaño **mayor que t.** Tenemos, además, una mesa con un número fijo M de huecos para albergar un ***máximo de M bloques de muñecas***. Un bloque de muñecas es cualquier composición de una o más muñecas, una dentro de otra (siempre cumpliendo la restricción de los tamaños).

Inicialmente, las N muñecas están distribuidas en bloques en los M huecos disponibles en la mesa, pudiendo ser el número de huecos ocupados menor que M. El objetivo es apilar las N muñecas rusas en un solo bloque en un hueco de la mesa, si esto es posible, teniendo en cuenta que el número máximo de

huecos o bloques con las que se puede trabajar es M. Para ello se dispone de un robot que puede realizar las siguientes operaciones:

1) El robot puede coger un bloque de muñecas de dos formas:
   a) Puede coger un bloque entero de muñecas que esté en uno de los huecos de la mesa, o bien,
   b) puede abrir la tapa de la muñeca más exterior de un bloque y coger todas las muñecas que hay en su interior, es decir las x-1 muñecas interiores de un bloque.

2) Una vez el robot tenga cogido un bloque de muñecas (bloque origen), el robot solo puede:
   a) introducir dicho bloque en otro bloque de muñecas (bloque destino) siempre y cuando el tamaño de la muñeca más exterior del bloque origen sea menor que el tamaño de la muñeca más interior del bloque destino, o bien,
   b) depositar el bloque origen de muñecas en otro hueco de la mesa siempre que dicho hueco esté vacío.

Por ejemplo, si en un hueco de la mesa hay un bloque de 6 muñecas como el que se muestra en la figura (las muñecas estarían contenidas unas en otras, de menor a mayor), el robot podría coger el bloque entero de 6 muñecas o bien podría abrir la muñeca más exterior (la más grande) y coger el bloque interior de 5 muñecas. Una vez el bloque de muñecas cogido, el robot puede introducirlo en otro bloque de muñecas de la mesa, si cabe todo el bloque cogido dentro de él, o bien formar un nuevo bloque de muñecas en un hueco disponible de la mesa.

Un ejemplo de situación inicial es: tenemos 11 muñecas rusas y una mesa con 4 huecos que puede albergar un máximo de 4 bloques de muñecas. Las 11 muñecas rusas están inicialmente distribuidas en tres bloques del siguiente modo: en un bloque están las muñecas de tamaño 1, 2, 5, 7, 8 y 11; otro bloque contiene las muñecas de tamaño 3, 4 y 9; un tercer bloque contiene las muñecas de tamaño 6 y 10.

Se pide diseñar un Sistema de Producción basado en reglas, siguiendo la sintaxis CLIPS, para determinar si dada una situación inicial de N muñecas y M huecos ***es posible resolver el problema o no***.

1) Especificar la base de hechos (para una situación inicial general) mediante patrones.
2) Especificar la BH inicial para la situación inicial indicada anteriormente, y especificar cómo sería la BH final en el SBR.
3) Especificar las reglas de producción. En la parte izquierda de la regla únicamente se pueden utilizar expresiones pattern-matching y "test". En la parte derecha de la regla sólo debe haber expresiones "assert", "retract","halt" y/o "printout". No se puede usar funciones.
4) El SBR debe devolver un mensaje diciendo si, para la situación inicial dada, es posible agrupar todas las muñecas en un solo bloque o no.

NOTA. Todas las reglas deben ser generalistas y ninguna de ellas debe estar condicionada al estado inicial o final de una instancia particular del problema.

Solución:

1) Especificación patrones

(maxbloques mb^s)

$mb^s \in$ INTEGER; indica el máximo número de bloques que se pueden formar en la mesa, o sea el máximo número de huecos que hay en la mesa. Este patrón representa un hecho estático ya que el máximo número de bloques o huecos de la mesa es fijo durante toda la ejecución de un problema.

(elementos x_1^s x_2^s … x_n^s)

$x_i^s \in$ INTEGER indica la i-muñeca, más concretamente el tamaño de la i-muñeca hasta un total de N muñecas. Este patrón representa también un hecho estático ya que el número de muñecas se mantiene fijo durante toda la ejecución de un problema.

(problema <bloque x^m finbloque>^{m} nbloques y^s robot z^m)

Este es el patrón principal y representa la información dinámica del problema. Se compone de:

a) un número variable de elementos **<bloque x^m finbloque>** que representa los bloques de muñecas que hay en la mesa; este número no podrá ser nunca inferior a 1 ni exceder el valor indicado en el hecho correspondiente al patrón (maxbloques mb^s). Si un robot coge el bloque entero de muñecas representado en **<bloque x^m finbloque>**, todos los elementos de **<bloque x^m finbloque>** se eliminarán del hecho principal.
b) $x^m \in 2^{[1\,..\,N]}$ indica las muñecas (concretamente tamaños de muñecas) que hay en ese bloque. Este valor puede ser cualquier combinación de números enteros de 1 hasta N (máximo número de muñecas), siempre respetando la restricción de los tamaños.
c) $y^s \in$ INTEGER representa el número de bloques o huecos de la mesa ocupados en dicho estado
d) $z^m \in \{EMPTY, 2^{[1\,..\,N]}\}$ representa si el robot está libre o no; si está libre, z^m tomará el valor EMPTY; en caso contrario el valor de z^m será el bloque de muñecas que el robot tiene agarrado.

2) BH inicial y BH final

BHinicial={
(maxbloques 4)
(elementos 1 2 3 4 5 6 7 8 9 10 11)
(problema bloque 1 2 5 7 8 11 finbloque bloque 3 4 9 finbloque bloque 6 10 finbloque nbloques 3 robot empty) }

BHfinal={
(maxbloques 4)
(elementos 1 2 3 4 5 6 7 8 9 10 11)
(problema bloque 1 2 3 4 5 6 7 8 9 10 11 finbloque nbloques 1 robot empty) }

3) 4) Reglas incluido mensajes que tiene que devolver el SBR

```
(deffacts datos
  (maxbloques 4)
  (elementos 1 2 3 4 5 6 7 8 9 10 11)
  (problema bloque 1 2 5 7 8 11 finbloque bloque 3 4 9 finbloque bloque 6 10 finbloque nbloques 3 robot
empty))


 (defrule sacar-bloque
   (problema $?x bloque $?b ?c finbloque $?y robot empty)
   (test (> (length $?b) 0))
    (test (not (member bloque $?b)))
 =>
   (assert (problema $?x bloque ?c finbloque $?y robot $?b)))


 (defrule coger-bloque-entero
      (problema $?x bloque $?b finbloque $?y nbloques ?np robot empty)
      (test (> (length $?b) 0))
      (test (not (member bloque $?b)))
 =>
     (assert (problema $?x $?y nbloques (- ?np 1) robot $?b)))


 (defrule meter-bloque
   (problema $?x bloque ?c1 $?b finbloque $?y robot $?r ?c2)
      (test (neq ?c2 empty))
      (test (not (member bloque $?b)))
      (test (< ?c2 ?c1))
 =>
     (assert (problema $?x bloque $?r ?c2 ?c1 $?b finbloque $?y robot empty)))


  (defrule hacer-bloque
    (problema $?x nbloques ?np robot $?r ?c2)
    (maxbloques ?maxp)
    (test (< ?np ?maxp))
    (test (neq ?c2 empty))
    =>
     (assert (problema $?x bloque $?r ?c2 finbloque nbloques (+ ?np 1) robot empty)))


 (defrule final1
  (declare (salience 100))
    (elementos $?elem)
    ?f1 <- (problema $?x bloque $?elem finbloque $?y)
 =>
     (printout t "Todas las muñecas estan apiladas  " crlf)
     (halt))
```

```
(defrule final2
  (declare (salience -5))
  =>
   (printout t "El problema no tiene solucion " crlf)
   (halt))
```

Ejercicio 15

Sea un SP cuya BH inicial es BH={(S 0 lista 2 3 1 20)} y cuya Base de Reglas se compone de la siguiente regla:

```
(defrule exam
  ?f <- (S ?s lista $?x ?y ?z $?w)
  (test (< ?z ?y))
=>
  (assert (S (+?s 1) lista ?x ?z ?y ?w)))
```

1) Asumiendo un encadenamiento hacia delante y una estrategia de búsqueda en anchura (mayor prioridad para los hechos e instancias de reglas más antiguas), ¿cuál será el estado final de la BH?. Realiza una traza que muestre el proceso inferencial.
2) Asume que se añade un comando (retract ¿f) en la RHS de la regla; muestra la nueva regla y el estado final de la BH.

Solución:

a)

Base de Hechos (BH)	**Conjunto Conflicto (CC)**
H1: (S 0 list 2 3 1 20)	exam: H1,?y=3, ?z=1, $?x=(2), $?w=(20) (1*)
H2: (S 1 list 2 1 3 20)	exam: H2, ?y=2, ?z=1, $?x=(), $?w=(3 20) (2*)
H3: (S 2 list 1 2 3 20)	No se producen más activaciones. De todas las posibles activaciones el patrón de la regla con el hecho H3, ninguna instanciación satisface el test.

La BH final contendrá tres hechos: BHfinal={(S 0 list 2 3 1 20) (S 1 list 2 1 3 20) (S 2 list 1 2 3 20)}

b) La traza será la misma excepto que se elimina de la BH el correspondiente hecho. De este modo, en cada ciclo del motor de inferencia, la BH contendrá únicamente un hecho. El contenido final de la BH será el hecho H3. BH final= BHfinal={(S 2 list 1 2 3 20)}.

Ejercicio 16

Considérese un problema caracterizado como sigue:

Existen tres torres (A1, A2 y A3) que contienen, cada una, un número indefinido de discos, todos ellos de diferente tamaño. Podemos asumir que el máximo tamaño es 99. Existe una cuarta torre (B) inicialmente vacía.

En las torres A1, A2 y A3, solo se puede colocar un disco sobre otro disco de tamaño mayor (o cuando la torre está vacía). En cambio, en la torre B, sólo se puede colocar un disco sobre otro disco de tamaño menor (o cuando la torre está vacía). Existe un brazo-robot que es capaz de mover un disco de una torre a otra. El brazo-robot sólo puede acceder al disco que está en la cima de las torres.

Diséñese un SBR que, a partir de una situación inicial cualquiera, obtenga una situación final en la que "todos los discos están en la torre B, ordenados de menor a mayor tamaño". Como situación inicial cualquiera entiéndase "un número indefinido de discos en las torres A1, A2 y A3; ninguno en la B". Por ejemplo, el estado inicial puede ser:

- Torre A1 contiene los discos 6 9 10
- Torre A2 contiene los discos 3 7 8
- Torre A3 contiene los discos 1 2 4
- Torre B está vacía

donde los números identifican el disco (su tamaño) correspondiente. En el estado final correspondiente, todos los discos están en la Torre B: {10 9 8 7 6 4 3 2 1}.

Notas: Se valorará un diseño de SVR que pueda alcanzar la solución con mínima búsqueda.

Solución:

(towers A1 dk_a1^{m} 100 A2 dk_a2^{m} 100 A3 dk_a3^{m} 100 B dk_b^{m}) , dk_a1, sk_a2, dk_a3, d_b $\in$ INTEGER

El entero 100 se utiliza para indicar la base de la torre, un disco 'suelo' de tamaño máximo.

```
(deffacts initialdata
    (towers A1 6 9 10 100 A2 3 7 8 100 A3 1 2 4 100 B))

(defrule move-from-A1-to-B
     (towers A1 ?da1 $?ra1 A2 ?da2 $?ra2 A3 ?da3 $?ra3 B $?rb)
     (test (<> ?da1 100))
     (test (and (< ?da1 ?da2)(< ?da1 ?da3)))
=>
     (assert (towers A1 $?ra1 A2 ?da2 $?ra2 A3 ?da3 $?ra3 B ?da1 $?rb)))


(defrule move-from-A2-to-B
     (towers A1 ?da1 $?ra1 A2 ?da2 $?ra2 A3 ?da3 $?ra3 B $?rb)
     (test (<> ?da2 100))
     (test (and (< ?da2 ?da1)(< ?da2 ?da3)))
=>
```

```
    (assert (towers A1 ?da1 $?ra1 A2 $?ra2 A3 ?da3 $?ra3 B ?da2 $?rb)))


(defrule move-from-A3-to-B
    (towers A1 ?da1 $?ra1 A2 ?da2 $?ra2 A3 ?da3 $?ra3 B $?rb)
    (test (<> ?da3 100))
    (test (and (< ?da3 ?da1)(< ?da3 ?da2)))
=>
    (assert (towers A1 ?da1 $?ra1 A2 ?da2 $?ra2 A3 $?ra3 B ?da3 $?rb)))


(defrule final
  (declare (salience 100))
    (towers A1 100 A2 100 A3 100 $?)
 =>
   (halt))
```

Con el diseño realizado, solo existirá una regla aplicable por cada hecho introducido en la BH, realizando así la mínima búsqueda posible.

La regla 'final' no sería necesaria ya que con el diseño realizado, cuando se alcanza el estado final ninguna de las reglas move-from-…. serían aplicables.

Ejercicio 17

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

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

Se pide:

Asumiendo un encadenamiento hacia delante, realiza y muestra una traza del SP aplicando sucesivamente el ciclo reconocimiento-acción de RETE hasta que el proceso se detenga. ¿Cuál es el resultado de la BH final?

Solución:

Base de Hechos	Agenda (instancias de reglas)
f-1:(lista a a b a) f-2:(par a 1) f-3:(par b 2)	R1:f-1,f-2 {?sym=a, $?x=(), $?y=(a b a), ?num=1, ?f=1} R1:f-1,f-2 {?sym=a, $?x=(a), $?y=(b a), ?num=1, ?f=1} R1:f-1,f-2 {?sym=a, $?x=(a a b), $?y=(), ?num=1, ?f=1} R1:f-1,f-3 {?sym=b, $?x=(a a), $?y=(a), ?num=2, ?f=1}
f-4:(lista 1 a b a)	R1:f-4,f-2 {?sym=a, $?x=(1), $?y=(b a), ?num=1, ?f=4} R1:f-4,f-3 {?sym=b, $?x=(1 a), $?y=(a), ?num=2, ?f=4} R1:f-4,f-2 {?sym=a, $?x=(1 a b), $?y=(), ?num=1, ?f=4}
f-5:(lista 1 1 b a)	R1:f-5,f-3 {?sym=b, $?x=(1 1), $?y=(a), ?num=2, ?f=5} R1:f-5,f-2 {?sym=a, $?x=(1 1 b), $?y=(), ?num=1, ?f=5}
f-6:(lista 1 1 2 a)	R1:f-6,f-2 {?sym=a, $?x=(1 1 2), $?y=(), ?num=1, ?f=6}
f-7: (lista 1 1 2 1)	

En la primera iteración se producen 4 instancias de R1, se dispara la primera de la agenda y se borran el resto por ser dependientes del hecho que se elimina f-1. En las siguientes iteraciones se produce sucesivamente una instancia menos ya que en cada iteración se reemplaza en el hecho (lista ...) un símbolo por un número. Independientemente de la estrategia de la agenda y el orden en el que se disparen las reglas, la BH final contendrá únicamente los hechos (lista 1 1 2 1)(par a 1)(par b2).

Ejercicio 18

Supongamos un problema del mundo de bloques. Disponemos de una superficie plana sobre la que pueden colocarse los bloques. Existen 5 bloques cúbicos etiquetados con las letras A, B, C, D y E. Todos los bloques son del mismo tamaño y pueden apilarse uno encima del otro de tal modo que sobre la superficie directa de un bloque puede existir a lo sumo otro bloque, formando así una pila con un máximo de N bloques. Hay un brazo de robot que puede manipular los bloques (sólo uno cada vez); las acciones que puede realizar este brazo son:

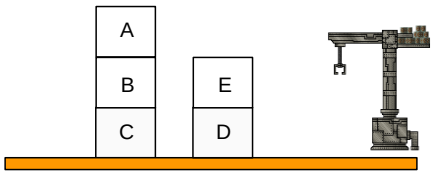
DESAPILAR(A,B): coger el bloque A en su posición actual encima del bloque B. El brazo debe estar vacío y el bloque A no debe tener ningún bloque encima de él. Cuando se desapila el bloque A del bloque B, el bloque B se queda libre sin ningún bloque encima.

APILAR(A,B): colocar el bloque A encima del bloque B. El brazo ya debe estar sosteniendo A, y la superficie de B debe estar despejada. Una vez se coloca el bloque A encima de B, el bloque B deja de estar libre.

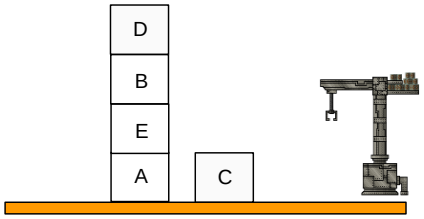
COGER(A): coger el bloque A de la mesa y sostenerlo. El brazo debe estar vacío y no debe haber nada encima del bloque A.

DEJAR(A): dejar el bloque A sobre la mesa. El brazo debe estar sosteniendo el bloque A.

Diseñar un SBR que determine la secuencia de acciones del brazo de robot para llegar a una configuración de bloques determinada partiendo de una configuración inicial. Por ejemplo:



initial state (s0)



goal state

Solución:

Patrón: (brazo br^s [pila bl^m]^m)

donde:

$br^s \in$ {libre, A, B, C, D, …}
$bl^m \in$ {A, B, C, D, …} con la excepción de que debe haber un bloque como mínimo. Si la pila está vacía entonces no existe tal pila y el símbolo 'pila' tampoco aparecerá en el hecho.
[pila bl^m]^m : puede haber tantas pilas como queramos; de nuevo, el multi-valuado aquí es una excepción porque tiene que haber como mínimo una pila

Generamos un patrón para el estado final que es igual que el patrón anterior pero con el símbolo 'meta' delante.

```
(deffacts inicial
  (brazo libre pila a b c pila e d)
  (meta brazo libre pila d b e a pila c))


(defrule Apilar
  ?f1 <- (brazo ?x $?r1 pila ?y $?r2)   ;el brazo sostiene a ?x (bloque) e ?y esta en cabeza
      (test (not (eq ?x libre)))
=>
  (assert (brazo libre $?r1 pila ?x ?y ?r2))
  (printout t "Apilar " ?x " en " ?y crlf))


(defrule Desapilar
  ?f1 <- (brazo libre $?r1 pila ?x ?y $?r2)   ;brazo libre, ?y (bloque) sobre ?x
      (test (not (eq ?y pila)))

=>
  (assert (brazo ?x $?r1 pila ?y $?r2))
  (printout t "Desapilar " ?x crlf))


(defrule Coger
  ?f1 <- (brazo libre $?r1 pila ?x $?r2)  ;brazo libre, ?x solo en mesa
      (test (or (= (length$ $?r2) 0) (eq pila (nth$ 1 $?r2))))
=>
```

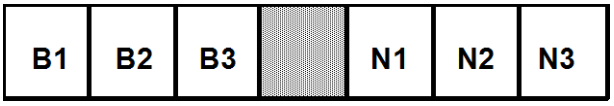
```
  (assert (brazo ?x $?r1 $?r2))
  (printout t "Coger " ?x crlf))


(defrule Dejar
  ?f1 <- (brazo ?x $?r1)    ;el brazo sostiene a ?x (bloque) y lo deja en mesa
        (test (not (eq ?x libre)))
=>
  (assert (brazo libre $?r1 pila ?x))
  (printout t "Dejar en mesa " ?x crlf))


(defrule final
  (declare (salience 100))
  (meta brazo libre $?v)
  (brazo libre $?v)
=>
  (printout t "Estado final alcanzado" crlf)
  (halt))
```

Ejercicio 19

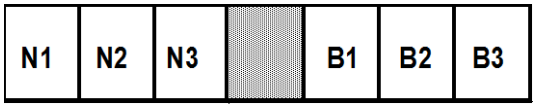
Un puzzle lineal consiste de tres fichas etiquetadas B1, B2 y B3, tres fichas etiquetadas N1, N2, y N3, y un espacio vacío. Supóngase que inicialmente se parte del puzzle de la figura:



Existen tres movimientos posibles:

a) poner una ficha en el espacio vacío. (Por ejemplo, B3 o N1 pueden pasar al espacio vacío).
b) saltar sobre una ficha al espacio vacío. (Por ejemplo, B2 o N2, saltarían al espacio vacío).
c) saltar sobre dos fichas al espacio vacío. (Por ejemplo, B1 o N3, saltarían al espacio vacío).

La meta es obtener el puzzle de la forma siguiente:



El estado inicial puede constar de cualquier distribución de las seis fichas, a izquierda o derecha del espacio vacío, y, en dicho estado inicial el espacio vacío puede estar en cualquier celda.

Diseña un Sistema de Producción que resuelva este problema para cualquier estado inicial:

- Especificar la estructura de los hechos que se van a emplear.
- Especificar el conjunto de reglas, en base a la representación elegida.

Solución:

En el diseño del patrón se ha incluido la información de 'nivel' para poder mostrar el nivel del nodo solución, esto es, la longitud de la solución. También se incluye la variable ?*nod-gen* que almacena el número de nodos expandidos o reglas disparadas.

```
(defglobal ?*prof* = 20)
(defglobal ?*nod-gen* = 0)

(deffacts inicio (puzzle B1 B2 B3 V N1 N2 N3 nivel 0))

(defrule Mover-1-Izq
      (puzzle $?y ?x V $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y V ?x $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))

(defrule Mover-2-Izq
      (puzzle $?y ?x1 ?x2 V $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y V ?x2 ?x1 $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))

(defrule Mover-3-Izq
      (puzzle $?y ?x1 ?x2 ?x3 V $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y V ?x2 ?x3 ?x1 $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))

(defrule Mover-1-Der
      (puzzle $?y V ?x $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y ?x V $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))

(defrule Mover-2-Der
      (puzzle $?y V ?x1 ?x2 $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y ?x2 ?x1 V $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))

(defrule Mover-3-Der
      (puzzle $?y V ?x1 ?x2 ?x3 $?z nivel ?n)
      (test (< ?n ?*prof*))
      =>
      (assert (puzzle $?y ?x3 ?x1 ?x2 V $?z nivel (+ 1 ?n)))
      (bind ?*nod-gen* (+ ?*nod-gen* 1)))
```

```
(defrule Objetivo
      (declare (salience 100))
      (puzzle N1 N2 N3 V B1 B2 B3 nivel ?n)
      =>
      (printout t "Solucion encontrada en el nivel: " ?n crlf)
      (printout t "NUMERO DE NODOS EXPANDIDOS O REGLAS DISPARADAS " ?*nod-gen* crlf)
      (halt)
)

(deffunction inicio ()
     (reset)
      (printout t "Profundidad Maxima:= " )
      (bind ?*prof* (read))
      (printout t "Tipo de Busqueda " crlf "    1.- Anchura" crlf "    2.- Profundidad" crlf )
      (bind ?a (read))
      (if (= ?a 1)
          then    (set-strategy breadth)
          else   (set-strategy depth))
     (printout t " Ejecuta run para poner en marcha el programa " crlf))
```

Ejercicio 20 (Examen 2013)

En una terminal de contenedores se dispone de un conjunto de N pilas donde se apilan contenedores, entre los cuales algunos van a ser cargados en el próximo barco. Así, se distinguen entre dos tipos de contenedores: de tipo A, si se van a cargar en el próximo barco y de tipo B, en caso contrario. Dada una situación como la de la figura 1, el objetivo del SBR sería redistribuir los contenedores, de forma que los de tipo A queden liberados, es decir, no tengan ningún contenedor de tipo B encima (ver figura 2).

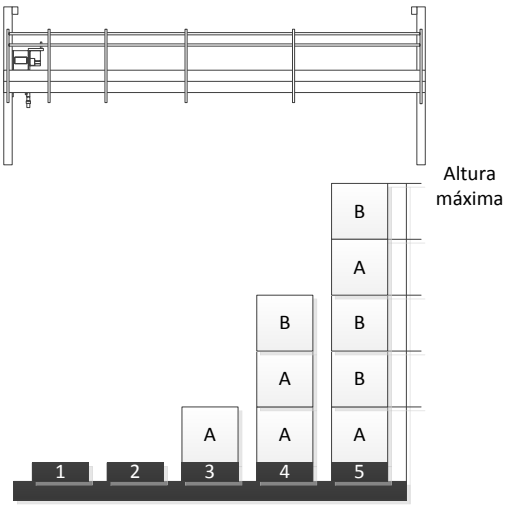


Figura 1. Situación inicial

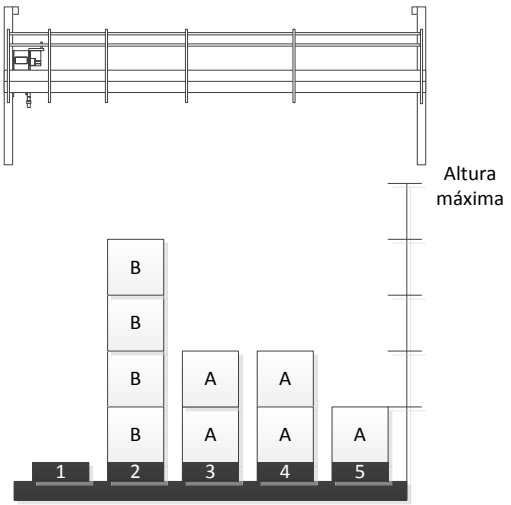


Figura 2. Una posible situación final

Los contenedores se pueden mover de una pila a otra por medio de una grúa que alcanza todas las pilas. Tanto el número de pilas como la altura de cada una de ellas están restringidos.

Asumamos que se utiliza el siguiente patrón para representar la información de un estado del problema:

(grua G^s [pila N^s contenedores C^s B^m]^m)

, donde:

- grua, pila y contenedores son símbolos constantes
- [pila N^s contenedores C^s B^m] se repite para cada una de las pilas
- G^s puede ser {libre, A, B}, es decir, se indica si la grúa está libre o qué tipo de contenedor sostiene
- N^s es el identificador de la pila, será un natural mayor que 0 que indicará la posición de la pila en la planta (1 pila del extremo izquierdo, n pila del extremo derecho, tal y como se indica en las figuras), las pilas estarán ordenadas de izquierda a derecha de menor a mayor identificador.
- C^s indica el número de contenedores en la pila
- B^m es la pila de contenedores, el primer contenedor de la lista es el que se encuentra en la parte superior de la pila

Además, se definen los siguientes hechos para indicar las restricciones sobre el número de pilas y sobre la altura máxima de las pilas.

(num-pilas X^s) (max-altura Y^s)

, donde X^s e Y^s son valores enteros. Se pide:

a) Escribir la base de hechos que corresponde a la situación inicial y final de las figuras 1 y 2. La representación de una pila vacía se deja a elección del alumno, teniendo en cuenta que el resto de reglas deben escribirse de acuerdo a esta representación.
b) Escribir una regla para desapilar un contenedor de tipo B de una pila si éste bloquea un contenedor de tipo A inferior.
c) Escribir una regla para apilar un contenedor de cualquier tipo en una pila distinta a la situada en el extremo derecho y que ya tenga al menos un contenedor. Se debe compensar la altura de las pilas, por lo que se podrá apilar un contenedor en una pila siempre y cuando la diferencia de altura entre dicha pila y la pila a su derecha sea inferior a 2 contenedores.
d) Se desea contabilizar el coste de todos los movimientos de contenedores realizados para alcanzar un determinado estado. El coste de desapilar cada contenedor de tipo A y de tipo B tiene un coste de 10 y 15, respectivamente. El coste de apilar cualquier tipo de contenedor es 5. Por ejemplo, la aplicación de estos costes daría un valor de 95 para el estado de la figura 2:
    a. Desapilar 4 contenedores B: 15*4 = 60
    b. Desapilar 1 contenedor A: 10*1 = 10
    c. Apilar los 5 contenedores: 5*5 = 25
    d. TOTAL = 95

Hacer las modificaciones necesarias en la Base de Hechos para poder representar esta información.

NOTAS:

- Estas reglas deben funcionar para trabajar con cualquier pila y con situaciones con cualquier número de pilas.
- Se pueden usar las siguientes funciones:

```
(member$ <expression> <multifield-expression>)
(length$ <multifield-expression>)
(abs <numeric-expression>)
```

Solución:

Apartado a)

```
(deffacts inicio
      (grua libre pila 1 contenedores 0 pila 2 contenedores 0 pila 3 contenedores 1 A pila 4 contenedores 3
B A A pila 5 contenedores 5 B A B B A)
      (max-altura 5)
      (num-pilas 5))
```

Situación final:

```
      (grua libre pila 1 contenedores 0 pila 2 contenedores 4 B B B B pila 3 contenedores 2 A A pila 4
contenedores 2 A A pila 5 contenedores 1 A)
```

Los hechos max-altura y num-pilas son estáticos y no varían.

Apartado b)

```
(defrule desapilar-B ;;se desapila un contenedor B si bloquea un contenedor A inferior
      (grua libre $?x pila ?p contenedores ?n B $?b $?y)
      (test (member A $?b))
      (test (= ?n (+ (length $?b) 1)))
=>
      (assert (grua B $?x pila ?p contenedores (- ?n 1) $?b $?y)))
```

Apartado c)

```
(defrule apilar
      (grua ?c $?x pila ?p contenedores ?n $?b pila ?p2 contenedores ?n2 $?b2 $?y)
      (max-altura ?max)
      (test (neq ?c libre))
      (test (> ?n 0))
      (test (= ?n (length $?b)))
      (test (= ?n2 (length $?b2)))
      (test (< ?n ?max))
      (test (< (abs (- ?n2 ?n)) 2))
=>
      (assert (grua libre $?x pila ?p contenedores (+ ?n 1) ?c $?b pila ?p2 contenedores ?n2 $?b2 $?y)))
```

Apartado d)

```
(deffacts inicio
      (grua libre pila 1 contenedores 0 pila 2 contenedores 0 pila 3 contenedores 1 A pila 4 contenedores 3
B A A pila 5 contenedores 5 B A B B A coste 0)
      (max-altura 5)
      (num-pilas 5)
         (coste desapilar A 10 B 15)
         (coste apilar A 5 B 5))
```

Ejercicio 21 (Examen 2013)

Supóngase el siguiente problema: "Un granjero quiere cruzar un río llevando consigo a una zorra, un ganso y un saco de trigo. Por desgracia, su bote es tan pequeño que solo puede transportar una de sus

pertenencias en cada viaje. Además, la zorra, si no se le vigila, se come al ganso, y el ganso, si no se le cuida, se come el trigo. Así, el granjero no debe dejar a la zorra sola con el ganso o al ganso solo con el trigo."

Se desea diseñar un SBR que, a partir de un estado inicial en el que están todos en el lado A del río, determine la secuencia de acciones para que pasen todos al lado B. Se pide:

a) Asumiendo la siguiente especificación de la Base de Hechos:

(problema Granjero x^S Zorra x^S Ganso x^S Trigo x^S) $x^S \in \{A, B\}$

NOTA: En la parte derecha de las reglas utilizad solo expresiones assert

a.1) Especificar en CLIPS la regla para pasar al granjero solo, del lado-A al lado-B.

```
(defrule GranjeroD  ; pasar hombre lado-B, no dejar ganso sólo con zorra o con trigo
  ?f <- (problema Granjero  A   Zorra  ?z   Ganso   ?g    Trigo ?t)
        (test   (or (eq ?g B)(and (eq ?t B)(eq ?z B))))
=>
        (assert (problema Granjero B Zorra ?z  Ganso ?g Trigo ?t)))
```

a.2) Especificar en CLIPS la regla para pasar a la zorra, del lado-A al lado-B.

```
(defrule ZorraD   ;pasar zorra a Lado-B, no dejar ganso sólo con el trigo
  ?f <- (problema Granjero A Zorra A Ganso ?g Trigo ?t)
        (test (or (eq ?g B)(eq ?t B)))     ; tambien, (not (and (eq ?g A )(eq ?t A)))
=>
        (assert (problema Granjero B Zorra B Ganso ?g Trigo ?t)))
```

b) Supongamos que el río es tan ancho que no puede cruzarse sin hacer paradas sucesivas en islotes {I1, I2, …, In} que hay al medio del río,

b.1) sin modificar el patrón ya indicado en (a), añade la información necesaria en la Base de Hechos, para cubrir esta nueva información, tal que las reglas puedan ser independientes del origen/destino del movimiento que representan.

La BH quedaría:

(problema Granjero x^S Zorra x^S Ganso x^S Trigo x^S) $x^S \in \{A, B, I1, I2, \ldots, In\}$

Y un conjunto de hechos indicando las conexiones que hayan entre las orillas del río e islotes:

(conectado A I1)	(conectado I1 A)
(conectado I1 I2)	(conectado I2 I1)
(conectado I2 I3)	(conectado I3 I2)
……	……
(conectado I_{n-1} I_n)	(conectado I_n I_{n-1})
(conectado I_n B)	(conectado B I_n)

b.2) con la modificación anterior, modificad la regla a.2 para pasar a la zorra, de una parte cualquiera a otra, sea lado del río o islote.

```
(defrule Zorra   ; pasar zorra de origen a destino, no dejar ganso sólo con el trigo"
  ?f <- (problema Granjero ?origen Zorra ?origen Ganso ?g Trigo ?t)
```

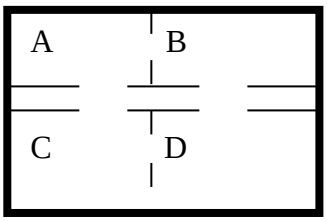
```
        (test  (not (and  (eq ?g ?origen )(eq ?t ?origen))))
        (conectado ?origen ?destino)
=>
        (assert (problema Granjero ?destino Zorra ?destino Ganso ?g Trigo ?t)))
```

Ejercicio 22 (Examen Enero 2015)

Se dispone de una zona distribuida en 4 habitaciones (A, B, C, D) y un pasillo que conecta con todas las habitaciones (ver figura). Existen 2 robots encargados de servir tazas de té o café a diferentes personas. Cada robot tiene asociada una zona por la que se puede mover y de la que no puede salir. La zona del primer robot son las habitaciones A y B y el pasillo. La zona del segundo robot son las habitaciones C y D y el pasillo.

En la habitación A se encuentra la máquina dispensadora de té y café. En la habitación C se encuentra el armario de las tazas. Las personas que solicitan tazas de té o café solo pueden estar en las habitaciones B o D. Cada robot solo puede llevar una taza en cada momento. Para pasar una taza (vacía o llena) de un robot X a un robot Y, el robot Y no debe llevar otra taza.

Las operaciones que se pueden hacer en el sistema son: (a) el robot coge una taza vacía, (b) los robots se pasan una taza entre sí, (c) el robot llena una taza vacía de té o café según petición, (d) un robot se mueve de una habitación a otra contigua o al pasillo y (e) el robot le da la taza a la persona que ha realizado la petición.



Dado el siguiente patrón para representar la información dinámica del problema,

(local robot 1 pos ?p1 lleva ?l1 robot 2 pos ?p2 lleva ?l2 peticion [persona ?p ?t]^m)

donde:

?p1 ∈ {A, B, PAS}, ?p2 ∈ {C, D, PAS}, ?p ∈ {B, D}, ?l1, ?l2 ∈ {nada, vacia, te, cafe}, ?t∈{te, cafe}

a) (0.4 puntos) Describe una base de hechos inicial donde el robot 1 está en la habitación A y el robot 2 en la habitación D. Hay dos personas en la habitación B que han pedido un café y un té, respectivamente.

(local robot 1 pos A lleva nada robot 2 pos D lleva nada peticion persona B cafe persona B te)
(conexion robot 1 A PAS) (conexion robot 1 PAS A)(conexion robot 1 A B) (conexion robot 1 B A)
(conexion robot 1 B PAS) (conexion robot 1 PAS B)
(conexion robot 2 C PAS) (conexion robot 2 PAS C)(conexion robot 2 C D) (conexion robot 2 D C)
(conexion robot 2 D PAS) (conexion robot 2 PAS D)

b) (0.2 puntos) Describe un posible estado final donde todas las peticiones han sido servidas.

```
(local robot 1 pos A lleva nada robot 2 pos D lleva nada peticion)
;; el resto de hechos son estáticos y se mantienen
```

c) (0.8 puntos) Escribe una única regla que sirva para mover cualquiera de los dos robots entre dos puntos de su zona permitida.

```
(defule mover
  (local $?x robot ?rob pos ?p1 $?y)
  (conexion robot ?rob ?p1 ?dest)
 =>
  (assert  (local $?x robot ?rob pos ?dest $?y)))
```

d) (0.8 puntos) Escribe una única regla que permita pasar una taza (vacía o llena) entre los dos robots.

```
(defule pasar_taza
 (local robot 1 pos ?p1 lleva ?l1 robot 2 pos ?p1 lleva ?l2 $?y)
 (test (or (and (eq ?l1 nada)(neq ?l2 nada)) (and (eq ?l2 nada)(neq ?l1 nada))))
 =>
 (assert  (local robot 1 pos ?p1 lleva ?l2 robot 2 pos ?p1 lleva ?l1 $?y)))
```

e) (0.8 puntos) Escribe una única regla para servir una petición a una persona.

```
(defule servir
 (local $?x robot ?rob pos ?pos lleva ?lle $?y persona ?pos ?lle $?z)
 =>
 (assert  (local $?x robot ?rob pos ?pos lleva nada $?y $?z)))
```

Ejercicio 23 (Examen Enero 2016)

En una población hay tres almacenes (A, B y C) cada uno de los cuales tiene guardados paquetes cuyo destino final es alguno de los otros dos almacenes. De este modo, el almacén A puede tener paquetes que son para B y/ó C, el almacén B tener paquetes que son para los almacenes A y/ó C, y el almacén C tener paquetes que son para los almacenes A y/ó B. El objetivo del problema es dejar todos los paquetes en su almacén destino.

Para transportar los paquetes, se dispone de un único camión que puede almacenar un máximo de 10 paquetes. El camión puede desplazarse entre cualquier par de almacenes. Cuando el camión está en un almacén X, se puede cargar en él paquetes que están en el almacén X y que tienen que ser transportados a otro almacén destino. Asimismo, cuando el camión se encuentra en un almacén X, se puede descargar únicamente paquetes del camión cuyo destino final sea dicho almacén X.

Ejemplo de situación inicial:

- En el almacén A hay 7 paquetes: 4 paquetes para B y 3 para C.
- En el almacén B hay 10 paquetes: 7 paquetes para A y 3 para C
- En el almacén C hay 6 paquetes, 3 paquetes para A y 3 paquetes para B.
- El camión está inicialmente en el almacén A y está vacío.

Dado el siguiente patrón para representar la información dinámica del problema

(transporte [almacen ?ciu [destino ?dest ?num]^m]^m camion ?loc [?dest_paq ?num_paq]^m total ?tot)

donde

?ciu, ?loc $\in$ {A,B,C}
?dest $\in$ {A,B,C} tal que ?dest $\neq$?ciu ;; destino
?num $\in$ INTEGER ;; número de paquetes a destino, inclusive cuando el número de paquetes es 0

?tot $\in$ INTEGER ;; número total de paquetes que lleva el camión

?dest_paq $\in$ {A,B,C} ;; destino, solo si existen paquetes para dicho destino
?num_paq $\in$ INTEGER tal que ?num_paq $\neq$ 0 ;; número de paquetes a destino siempre y cuando el número de paquetes sea distinto de 0

NOTA 1: Si el camión no lleva paquetes para un destino X entonces la etiqueta [X 0] no se almacena en el hecho.
NOTA 2: En cada almacén solo se representan el número de paquetes que tienen que ser transportados a otro almacén (se ignora los paquetes del propio almacén).

NOTA 3: Pueden añadirse hechos estáticos a la representación del problema si son necesarios para alguna de las reglas que se solicitan.

a) (0.5 puntos) Describe la BH inicial para reflejar la situación inicial descrita arriba.

(transporte almacen A destino B 4 destino C 3 almacen B destino A 7 destino C 3 almacen C destino A 3 destino B 3 camion A total 0)

b) (1 punto) Escribe una única regla que sirva para cargar en el camión todos los paquetes que hay en un almacén para un destino determinado y asumiendo que el camión no lleva previamente paquetes para dicho destino. Debe de cumplirse la restricción sobre el total de paquetes que puede llevar el camión.

```
(defrule cargar
   (transporte $?x1 almacen ?alm $?y1 destino ?dest ?num $?y2 camion ?alm $?z total ?total)
    (test (> ?num 0))
    (test (not (member almacen $?y1)))
    (test (<= (+ ?total ?num) 10))
  =>
   (assert (transporte $?x1 almacen ?alm $?y1 destino ?dest 0 $?y2 camion ?alm ?dest ?num
$?z total (+ ?total ?num))))
```

c) (0.8 puntos) Escribe una única regla que muestre un mensaje por pantalla por cada destino para el cual el camión NO lleva paquetes. Se debe mostrar un mensaje del tipo “El camión NO lleva paquetes para el destino XXXX “, para cada uno de los destinos que cumplan esta condición.

Generamos un hecho (destinos A B C)

```
(defrule mostrar
   (transporte $?x camion ?loc $?z total ?tot)
   (destinos $? ?d $?)
   (test (not (member ?d $?z)))
  =>
   (printout t "El camión no lleva paquetes para el destino "?d crlf))
```

d) (0.7 puntos) Escribe una única regla para descargar todos los paquetes que lleva el camión para un destino determinado. La regla deber servir para cualquier destino y en el hecho resultante no debe aparecer la etiqueta del destino ni número de paquetes.

```
(defrule descargar
   (transporte $?x almacen ?alm $?z camion ?loc $?y ?loc ?elem2 $?z total ?tot)
  =>
   (assert (transporte $?x camion ?loc $?y $?z total (- ?tot ?elem2))))
```

Ejercicio 24 (Examen Enero 2017)

Se desea diseñar un Sistema Basado en Reglas (SBR) que permita manejar la colección de cromos de varios niños. Cada cromo viene representado por un identificador alfanumérico (A1, B3, etc.). Cada niño puede tener varios cromos (incluidos repetidos) y el número de niños no está limitado en la aplicación. Un posible ejemplo es:

- El niño 1 tiene los cromos: A2 A4 A5 B1 A2 B3
- El niño 2 tiene los cromos: B3 A4 C2 C1 B3 C2
- El niño 3 tiene los cromos: C2 C4 B1 A2

La información dinámica del problema se representaría con el siguiente patrón:

(colecciones [niño ?n [?id-cromo]^m fniño ?n]^m)

donde:

?n ∈ INTEGER ;; Identificador del niño
?id-cromo ∈ {A1, A2, B1,…} ;;Identificador del cromo

Se pide:

a) (0.3 puntos) Escribe la Base de Hechos correspondiente al ejemplo que se muestra arriba.

```
(deffacts datos
 (colecciones  niño 1 A2 A4 A5 B1 A2 B3 fniño 1
               niño 2 B3 A4 C2 C1 B3 C2 fniño 2
               niño 3 C2 C4 B1 A2 fniño 3))
```

b) (1 punto) Escribe una única regla que permita a dos niños intercambiar un cromo. El intercambio solo es posible si el cromo que entrega cada niño es un cromo repetido en su colección, y el cromo que recibe cada niño es un cromo que no está en su colección.

```
(defrule intercambio
 (colecciones $?x niño ?n1 $?c1 ?c $?c2 ?c $?c3 fniño ?n1
                  $?y niño ?n2 $?p1 ?p $?p2 ?p $?p3 fniño ?n2 $?z)
 (test (neq ?c ?p))
 (test (and (not (member$ ?c $?p1))(not (member$ ?c $?p2))(not (member$ ?c $?p3))))
 (test (and (not (member$ ?p $?c1))(not (member$ ?p $?c2))(not (member$ ?p $?c3))))
 =>
 (assert (colecciones $?x niño ?n1 $?c1 ?c $?c2 ?p $?c3 fniño ?n1 $?y niño ?n2 $?p1 ?p
$?p2 ?c $?p3 fniño ?n2 $?z)))
```

Otra solución:

```
(defrule intercambio
```

```
  ?f1 <- (colecciones $?x niño ?n1 $?c1 ?c $?c2 ?c $?c3 fniño ?n1
                $?y niño ?n2 $?p1 ?p $?p2 ?p $?p3 fniño ?n2 $?z)
  ?f2 <- (colecciones $?x niño ?n1 $?todo1 fniño ?n1
              $?y niño ?n2 $?todo2 fniño ?n2 $?z)
  (test (eq ?f1 ?f2))
  (test (neq ?c ?p))
  (test (and (not (member$ ?p $?todo1))(not (member$ ?c $?todo2))))
 =>
 (assert (colecciones $?x niño ?n1 $?c1 ?c $?c2 ?p $?c3 fniño ?n1 $?y niño ?n2 $?p1 ?p
$?p2 ?c $?p3 fniño ?n2 $?z)))
```

c) (0.7 puntos) Escribe una única regla que muestre los niños que tienen algún cromo que aparece (exactamente) tres veces en la colección. Hay que mostrar un mensaje por pantalla por cada niño y cromo; ejemplo: "El niño " ?n " tiene el cromo " ?x " tres veces" .

```
(defrule acaparador
  (colecciones $? niño ?n $?x1 ?y1 $?x2 ?y1 $?x3 ?y1 $?x4 fniño ?n $?)
  (test (and (not (member ?y1 $?x1))(not (member?y1 $?x2))
             (not (member ?y1 $?x3))(not (member ?y1 $?x4))))
 =>
 (printout t "El niño " ?n " tiene el cromo " ?y1 " tres veces " crlf))
```

d) (1 punto) Supongamos que existen unos hechos del tipo (especial ?id-cromo) para indicar que el cromo identificado por ?id-cromo es un cromo especial. Escribe una única regla que calcule el número de niños que tienen al menos 2 cromos especiales y distintos entre sí. El resultado de la ejecución de la regla será un hecho con formato: (lista-especial $[?n]^m$) donde ?n es el identificador de un niño con al menos dos cromos especiales y diferentes. El identificador de cada niño solo debe aparecer una vez en la lista (aunque tenga varios cromos especiales). Asumir que en la BH existen varios hechos del tipo (especial ?id-cromo) y el hecho (lista-especial).

```
(defrule especiales
   (colecciones $? niño ?n1 $? ?a $? ?b $? fniño ?n1 $?y)
   (especial ?a)
   (especial ?b)
   (test (neq ?a ?b))
?r <- (lista-especial $?z)
   (test (not (member$ ?n1 $?z)))
 =>
   (retract ?r)
   (assert (lista-especial $?z ?n1)))
```

Ejercicio 25 (Enero 2018)

En un aeropuerto se disponen de varios trenes de equipaje para llevar las maletas desde la zona de facturación al avión asignado al vuelo de las maletas. Una maleta facturada lleva la etiqueta del vuelo correspondiente. Inicialmente los trenes no están asignados a ningún vuelo. El vuelo asignado a un tren será el vuelo de la primera maleta que se cargue en el tren. Un tren solo puede llevar maletas para un único vuelo y cada vuelo solo se puede asignar a un tren.

El patrón para representar la información dinámica de un estado de este problema es:

(aeropuerto [TREN num^s dest^s mal^m]^m) donde

num $\in$ INTEGER ;; es un número que identifica el tren
dest $\in$ {nada, F1, F2, F3,…} ;; es un símbolo que representa el vuelo asignado al tren (inicialmente cuando el vuelo es desconocido, el símbolo será nada)
mal $\in$ {M1, M2, M3,…};; es un símbolo que representa el identificador de la maleta (inicialmente este campo está vacío)

Una posible situación inicial del problema es la siguiente:

- Se tienen cinco maletas (M1, M2, M3, M4 y M5), las dos primeras están facturadas para el vuelo F14, la tercera para el vuelo F2 y las dos últimas para el vuelo F10
- Se dispone de tres trenes para recogida y reparto de equipaje y los trenes están vacíos

Se desea resolver este problema mediante un proceso de búsqueda en un espacio de estados con el diseño de un SBR en CLIPS. Se pide:

1) (0.7 puntos) Escribe la Base de Hechos correspondiente a la situación inicial que se muestra arriba. Incluye los patrones adicionales que necesites para representar la información estática del problema, así como los hechos asociados a dichos patrones.

```
(deffacts datos
  (destino M1 F14)
  (destino M2 F14)
  (destino M3 F2)
  (destino M4 F10)
  (destino M5 F10)
  (aeropuerto TREN 1 nada TREN 2 nada TREN 3 nada))
```

2) (1 punto) Escribe una regla para cargar la primera maleta en un tren y asignar el vuelo de la maleta cargada a dicho tren.

```
(defrule tren_vuelo
  (destino ?mal ?flight)
  (aeropuerto $?y TREN ?n ?dest $?z)
  (test (eq ?dest nada))
  (test (not (member ?flight $?y)))
```

```
  (test (not (member ?flight $?z)))
 =>
  (assert (aeropuerto $?y TREN ?n ?flight ?mal $?z)))
```

3) (0.8 puntos) Escribe una regla para cargar una maleta a un tren cuando el tren ya tiene asignado un vuelo. El vuelo de la maleta debe ser el mismo que el del tren y la maleta no debe estar ya cargada en el tren.

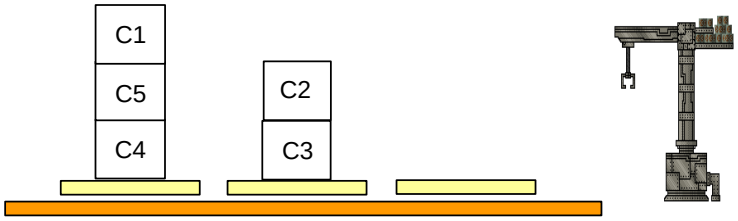
```
(defrule maleta_tren
   (destino ?mal ?flight)
   (aeropuerto $?x TREN ?n ?flight $?maletas)
   (test (not (member ?mal $?maletas)))
  =>
   (assert (aeropuerto $?x TREN ?n ?flight ?mal $?maletas)))
```

4) (0.5 puntos) Supongamos el patrón (vuelo vol^s) donde $vol^s \in$ {F1, F2, F3,…} es el identificador de un vuelo. Asumiendo un hecho que representa un vuelo determinado, escribe una regla que muestre por pantalla todas las maletas cargadas en el tren para dicho vuelo. Se deberá mostrar un único mensaje del tipo: "Las maletas X X X han sido cargadas en el tren Y".

```
(defrule listado_maletas
   (vuelo ?flight)
   (aeropuerto $? TREN ?n ?flight $?maletas $?resto)
   (test (not (member TREN $?maletas)))
   (test (or (= (length$ $?resto) 0) (eq (nth$ 1 $?resto) TREN)))
  =>
   (printout t "Las maletas " $?maletas " han sido cargadas en el tren "  ?n crlf))
```

Ejercicio 26 (Enero 2018)

En un puerto se dispone de un conjunto de contenedores apilados en varias pilas. Un ejemplo de estado inicial se puede ver en la figura donde hay un máximo de tres pilas, los contenedores C1 C5 y C4 están en la pila 1 y los contenedores C2 y C3 están en la pila 2 y la pila 3 está vacía. Los contenedores están apilados de tal modo que un contenedor *Ci* solo puede estar apilado encima de otro contenedor *Cj* si el peso de *Ci* es menor que el peso de *Cj*.



Se dispone también de una grúa que puede coger una torre de **n** contenedores de una pila, donde **n** <= 3, pudiendo ser **n** todos los contenedores de la pila. Asumimos que la torre de contenedores

que coge la grúa están en la misma posición que aparecen en la pila. Esto es, si la grúa coge la torre de contenedores [C1 C5], el contenedor C5 es la base de la torre y el contenedor C1 es el tope de la pila. Y si la grúa coge la torre de contenedores [C1 C5 C4], la base de la torre será C4 y el tope de la pila será C1. La grúa puede depositar una torre de contenedores en una pila vacía o bien sobre otro contenedor *Ci* siempre y cuando el peso del contenedor de la base de la torre sea menor que el peso de *Ci*. Por ejemplo, la grúa puede depositar la torre [C1 C5] encima de C2 siempre y cuando el peso de C5 sea menor que el peso de C2.

El patrón para representar la información dinámica de un estado de este problema es:

$$(\text{puerto } [\text{pila num}^{s} \text{ cont}^{m} \text{ finpila}]^{m} \text{ grua cogr}^{m}) \text{ donde}$$

num ∈ INTEGER ;; es un número que identifica la pila
cont ∈ {*C1*, *C2*, *C3*,…} ;; es un símbolo que representa un contenedor, el campo representa los contenedores de la pila
cogr ∈ {*C1*, *C2*, *C3*,…} ;; representa los contenedores que tiene la grúa

Se desea resolver este problema mediante un proceso de búsqueda en un espacio de estados con el diseño de un SBR en CLIPS. Se pide:

1) (0.7 puntos) Escribe la Base de Hechos correspondiente a la situación inicial que se muestra arriba. Incluye los patrones adicionales que necesites para representar la información estática del problema, así como los hechos asociados a dichos patrones.

```
(deffacts datos
   (puerto pila 1 C1 C5 C4 finpila pila 2 C2 C3 finpila pila 3 finpila grua)
   (peso C1 10)
   (peso C5 20)
   (peso C4 30)
   (peso C2 33)
   (peso C3 35))
```

2) (0.7 puntos) Escribe una regla que permita a la grúa coger todos los contenedores de una pila siempre y cuando se satisfaga la restricción de que el número de contenedores de la pila sea < = 3.

```
(defrule coger_n_bloques
   (puerto $?x pila ?num $?resto finpila $?y grua)
   (test (and (<= (length$ $?resto) 3) (>= (length$ $?resto) 1)))
   (test (not (member pila $?resto)))
   =>
   (assert (puerto $?x pila ?num finpila $?y grua ?top $?resto)))
```

3) (0.8 puntos) Asumiendo que la grúa está sujetando una torre de contenedores, escribir una regla para depositar esta torre completa de contenedores en una pila que contenga al menos un contenedor, siempre y cuando se satisfaga la restricción de pesos indicada en el enunciado.

```
(defrule dejar_bloques
   (puerto $?x pila ?num ?top $?resto finpila $?y grua $?restrob ?base)
   (peso ?top ?pestop)
   (peso ?base ?pesbas)
   (test (not (member pila $?resto)))
   (test (< ?pesbas ?pestop))
   =>
   (assert (puerto $?x pila ?num $?restrob ?base ?top $?resto finpila $?y grua)))
```

4) (0.8 puntos) Escribe una regla que muestre los contenedores de una pila. La regla deberá mostrar un mensaje de siguiente tipo para cada bloque contenido en una pila. "El contenedor Y está en la pila X".

```
(defrule mostrar
   (puerto $?x pila ?num $?p1 ?con $?p2 finpila $?y grua)
   (test (not (member pila $?p1)))
   (test (not (eq ?con pila)))
   (test (not (member pila $?p2)))
   =>
   (printout t "La pila " ?num " tiene el bloque " ?con crlf))
```