<u>MULTIPLE CHOICE QUESTIONS</u>

RULE BASED SYSTEMS

1) Given a RBS composed of a single rule:

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

, and WMinitial={(list a b c b c b c b c b a b c b c b c b c b a)}, how will the **final state** of the WM be?

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

---

2) Given a RBS that solves a particular problem, the Working Memory represents:

A. The static data of the problem.
B. The dynamic data of the initial state of the problem.
C. The static and dynamic data of the initial state of the problem.
D. None of the above is true. The WM contains the facts and the rules of the problem.

---

3) We have various boxes of different size on a table and we want to have them stacked on a pile from the largest box (at the bottom) to the smallest box (at the top) (see the 'final' rule to check the goal we are aimed to). We design a RBS with the initial WM:

```
(deffacts data (table 2 5 1 6 8 7 4 pile))
```

, and the following rules:

```
(defrule table-to-pile
  (table  $?rest1  ?x  $?rest3  pile  $?rest1 ?y)
  (test (> ?y ?x))
 =>
   (assert (table $?rest1 $?rest3  pile  $?rest1 ?y ?x)))

(defrule table-to-empty-pile
   (table $?rest1 ?x $?rest3  pile)
 =>
   (assert (table $?rest1 $?rest3 pile ?x)))
```

```
(defrule final
   (table pile 1 2 4 5 6 7 8)
 =>
   (halt))
```

Which of the following assertions is true?

A. The RBS works properly and it returns the correct solution that we want.
B. The RBS works properly, it returns the correct solution that we want and the 'final' rule is not necessary because the inference process will end when the table has no more boxes on it.
C. The RBS only works properly if the boxes on the table are ordered from the smallest to the largest; for instance: (deffacts data (table 1 2 4 5 6 7 8 pile))
D. None of the above assertions is true.

---

4) Given the following RBS that computes the factorial of a number, which of the following assertions is true?

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

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

A. The RBS works properly and it returns the correct solution.
B. Assertion A is not true because a test (test (> ?n1 1)) is required in the LHS of the rule to work properly.
C. The modification proposed in assertion B is not enough because a 'stop' rule is also necessary: (defrule stop (declare (salience 100)) (number 1) => (halt)).
D. None of the above assertions is true.

---

5) Given the fact:

(school class 1 boys 15 girls 18 class 2 boys 21 girls 14 class 3 boys 16 girls 17)

, where the numeric value after the symbol 'class' is the class identifier and the numeric values after the symbols 'boys' and 'girls' denote the number of boys and girls in the respective class.

Indicate the correct pattern to obtain **ONLY** the identifier of any of the classes and the number of girls in such class.

A. (school $? class ?c $? girls ?g $?)
B. (school $? class ?c boys ? girls ?g $?)
C. (school $? class ? boys $? girls ?g $?)
D. (school class ?c boys ? girls ?g)

---

6) Given a RBS with WMinitial={(list 23 14 56 33)}, and the rules:

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

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

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

, the contents of the Agenda (Conflict Set) after the first *pattern-matching* is:

A. One instance of R1, one instance of R2 and one instance of the rule 'final'
B. Two instances of R2 and one instance of R1
C. Two instances of R2, one instance of R1 and one instance of the rule 'final'
D. One instance of the rule 'final'

---

7) Given the RBS in question 6 and assuming the strategy of the Agenda is BREADTH-FIRST, which rule instance is first selected by the CLIPS inference engine to be executed? Mark the assertion that is TRUE:

A. It selects an instance of the rule R1.
B. It selects an instance of the rule R2.
C. It selects an instance of the rule 'final'.
D. None of the above.

---

8) Given a RBS composed of a single rule:

```
(defrule R1
   ?f <- (lista ?x $?y ?x $?z)
=>
    (retract ?f)
    (assert (lista $?y ?x $?z))
    (printout t "The list has changed " crlf))
```

, and WMinitial={(lista a b a b a)}, after executing the RBS, how many times the message "The list has changed " will be shown on the screen?

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

---

9) We have a fork lift to pick up objects from a ground floor and deliver them to the other floors of a building. In a particular problem instance there are two objects, A and B, whose destinations are the second and third floor, and weigh 2 Kg. and 8 Kg., respectively. The fork lift is at the ground floor and it cannot carry objects for more than 40 Kg. Which of the following representations **is NOT appropriate** to implement a graph search in a state-based representation?

A. (fork-lift floor 0 load object A 2 2 object B 3 8 max-weight 40 level 0)
B. (fork-lift floor 0 load object A 2 2 object B 3 8) (max-weight 40)
C. (fork-lift floor 0 load object A 2 2 object B 3 8 level 0) (max-weight 40)
D. (fork-lift floor 0) (load object A 2 2 object B 3 8 level 0) (max-weight 40)

---

10) Given the following RBS, how many rule instances will be inserted in the Agenda in the first inference cycle?

```
(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. None
D. 3

11) Given the following RBS, which of the following assertions is **<u>CORRECT?</u>**:

```
 (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. Only when the strategy of the Agenda is BREADTH, the first rule instance to be executed will be an instance of R1

B. Only when the strategy of the Agenda is DEPTH, the first rule instance to be executed will be an instance of R2

C. An instance of R1 will be always executed in the first place

D. An instance of R2 will be always executed in the first place

---

12) Let the fact (heap A B A A B B A heapA heapB) be the initial state of a RBS. The fact represents an initial heap that contains blocks of type A and B and the goal is to put each block in its corresponding heap; i.e., in heapA or in heapB. Which of the following rules **<u>DOES NOT</u>** take a block A and moves it to heapA so that the problem can be solved?

A.
```
(defrule move-to-heap-A
     (heap $?x A $?y heapA $?z)
  =>
     (assert (heap $?x $?y heapA A $?z)))
```

B.
```
(defrule move-to-heap-A
     (heap $?x ?b $?y heapA $?z)
     (test (eq ?b A))
  =>
    (assert (heap $?x $?y heapA A $?z)))
```

C. (defrule move-to-heap-A
```
        (heap $?x ?b $?y heapA $?z)
        (test (eq ?b A))
  =>
        (assert (heap $?x ?b $?y heapA ?b $?z)))
```

D. (defrule move-to-heap-A
```
        (heap $?x ?b $?y heapA $?z)
        (test (eq ?b A))
  =>
       (assert (heap $?x $?y heapA ?b $?z)))
```

---

13) A given warehouse has two distinctive areas: a load area and an unload area. In each area, we can find several heaps of blocks of type A, B or C. Heaps are identified with an integer number from 1 to 5. The goal of the problem is to put blocks from the load area in a truck and take them to the unload area. Let be the initial fact:

(warehouse area load heap 1 A B C heap 2 B C B heap 3 A area unload heap 4 A B A heap 5 B A B B A)

Assuming that we wish to instance in variable ?p only the identifier of a heap of the load area whose first block is of type A, which of the following patterns **IS NOT** valid for this purpose?

A. (warehouse area load $?c heap ?p A $?r area unload $?d)
B. (warehouse area load $?c heap ?p A $?r)
C. (warehouse $?c heap ?p A $?r area unload $?d)
D. (warehouse $?c heap ?p A $?r 4 $?d)

---

14) Given WMinitial= {(elemento e) (lista e a e b c d e f)} and the following rules:

```
(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)))
```

Which of the following assertions is **CORRECT**? (NOTE: the semicolon (;) before the (declare (salience …)) commands indicate the command is commented)

A. The final state will depend on the search strategy (breadth, depth, uniform cost, etc.)
B. No rule instance is ever triggered in this RBS
C. The final state would depend on the rules priority (salience …) if the (declare (salience …)) commands were not commented
D. The final state is the same regardless of the search strategy

---

15) Given the LHS of the rule:

```
    (defrule R1
            (list $? ?x $? ?y)
            (test (< ?x ?y))
    =>
        …
```

, and the fact: (list 1 3 2 1 3 6), how many instances of the rule will be inserted in the Agenda?

A. 0
B. 1
C. 5
D. More than 5

---

16) Given the WMinitial={(lista1 b a a a c c a c b b c)(lista2 a c)} and the following rule, indicate the final WM that will be reached among the given options.

```
(defrule  R1
  ?f <- (lista1  $?x ?a ?a $?y)
        (lista2  $? ?a $?)
  =>
    (retract ?f)
    (assert (lista1 $?x ?a $?y)))
```

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) Given the LHS of the rule:

```
(defrule R2
   ?f <- (lista  $? ?b $?x ?b $?x)
    =>
   ...
```

and the fact (lista c c d c c d c c d), how many rule instances will be inserted in the Agenda?

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

---

18) Let be the WMinitial={(lista 5 7 3 1 6 4) (maximo 0)} and the following rule to calculate the maximum value of a list of numbers

```
(defrule R4
    ?f1 <- (lista $?a ?b $?c)
    ?f2 <- (maximo ?x)
    (test (>  ?b ?x))
=>
     (assert (lista $?a $?c))
     (assert (maximo ?b)))
```

Assuming we wish to obtain a final WM (after successively executing the rule R4) in which a fact of type (maximo …) appears only once with the maximum value of the list of numbers, which of the following assertions is **TRUE**?

A. The rule is correct
B. It is necessary to add (retract ?f1)
C. It is necessary to add (retract ?f2)
D. The modification of answer C is not sufficient and it is also necessary to add (retract ?f1)

---

19) Given the fact (problem tower a b c name A tower a name B tower name C), which of the following patterns is suitable to retrieve the name of a tower that contains only one element?

A. (problem $?x tower ?a $?y name ? $?z)
B. (problem $?x tower ?a name ?z $?x)
C. (problem $?x tower ?a name ?z $?)
D. (problem $? tower ?a name ?)

---

20) Let be a RBS to calculate the Fibonacci number of an integer n>0, for example, n=5 (the Fibonacci number is stored in fib-1), which of the following assertions is **TRUE**?

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

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

NOTE: The Fibonnaci number is calculated as: $f(n)=f(n-1)+f(n-2)$ with $f(0)=0$ and $f(1)=1$.

A. The RBS works correctly
B. The above assertion is not true because a test condition (test (> ?n1 1)) is needed in the LHS of the rule to work properly
C. The above modification is not sufficient and it is also necessary to add a halt rule:
(defrule stop (declare (salience 100)) (number 1) => (halt)).
D. None of the above assertions is true

---

21) Given the fact:

```
(market street 1 fruit 20 fish 0 street 2 fruit 10 fish 10 street 3 fruit 16 fish 4)
```

, where the numeric value after the symbol 'street' is the street identifier and the numeric values after the symbols 'fish' and 'fruit' indicate the number of stalls of fish and fruit in the respective street. Indicate the correct pattern to obtain **ONLY** the identifier of any of the streets and the number of fish stalls in such street.

A. (market $? street ? fruit $? fish ?n $?)
B. (market $? street ?c $? fish ?n $?)
C. (market street ?c fruit ? fish ?n)
D. (market $? street ?c fruit ? fish ?n $?)

---

22) Let be a RBS composed of WMinitial={(lista 2 1 6 2 3)}, and the following rules:

```
(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)))
```

What are the contents of the Agenda after the first pattern-matching?

A. An instance of R1 and two instances of R2
B. Two instances of R1 and one instance of R2
C. Two instances of R1 and two instances of R2
D. No rule instance is generated

---

23) Given a RBS composed of the following rule:

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

, and the initial Working Memory {(lista 1 2 3 2 3 2 3 2 3 2 1 2 3 2 3 2 3 2 3 2 1)}, which is the final state of the WM?

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

---

24) The pattern format (list [name^s age^s]^m) represents the name and age of a number of people. Given a particular list of persons, we wish to count the number of them whose age is between 18 and 65 years old. In order to do this, we have the fact that represents the particular list of people, an initial fact (counter 0) that is used to count the number of people and the rule shown below. Mark the **CORRECT** answer.

```
(defrule count
   ?f1 <-(list  $?x1 ?num $?x2)
   ?f2 <- (counter ?cont)
   (test (numberp ?num))   ;; numberp returns TRUE if ?num is a number
   (test (and (>= ?num 18)(<= ?num 65)))
```

```
 =>
   (retract ?f2)
   (assert (counter(+ ?cont 1))))
```

A. The RBS works properly and returns the desired outcome.
B. It is only necessary to add the instruction (assert (lista $?x1 $?x2)) in the RHS of the rule for the RBS to work properly.
C. It is necessary to add the instructions (retract ?f1) and (assert (lista $?x1 $?x2)) in the LHS of the rule for the RBS to work properly.
D. None of the above.

---

25) Given the initial WM={(lista b a a a c a c b b c) (lista1 a c d e f g)} and the rule shown below, mark the answer that represents the final state of the WM.

```
(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) Let be an initial WM={(lista 5 7 3 1 6 4) (minimum 9999)} and the rule shown below. Assuming our goal is to obtain a final WM (after successively executing the rule REGLA) in which a fact of the type (minimum …) appears only once containing the minimum value of the list, which of the following assertions is **TRUE** to accomplish our goal?

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

A. The rule is correct
B. It is necessary to add (retract ?f2)
C. It is necessary to add (retract ?f1)
D. It is necessary to add (retract ?f1) and (retract ?f2)

27) Indicate the final **CORRECT** outcome after executing the following RBS with an initial WM={(lista 34 77 34)}:

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

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

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

A. It will show three times the message "Message 1".
B. It will show once "Message 1" and once "Message 2".
C. It will show once "Message 1" and once "Message 3".
D. It will show once "Message 1", once "Message 2" and once "Message 3".

---

28) The following fact represents a number of heaps along with the blocks contained in each heap. The number after the symbol 'heap' denotes the heap identifier; the blocks comprised by the heap appear after the identifier, being the first block the one on the top of the heap. Mark the **CORRECT** option that instantiates the block at the bottom of any heap for the fact shown below:

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

A. (problem $?x1 heap ?num $?x2 ?y heap $?x3) (test (not (member heap $?x1)))
B. (problem $?x1 heap ?num $?x2 ?y heap $?x3) (test (not (member heap $?x2)))
C. (problem $?x1 heap ?num $?x2 ?y $?x3)(test (not (member heap $?x3)))
D. None of the above.

---

29) Let be a RBS formed by WMinitial={(lista 2 1 5 3)}, and the following rules:

```
(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))
```

after the first *pattern-matching,* how would the activations be ordered in the Agenda?

A. One instance of rule R1, one instance of rule R2 and one instance of the final rule
B. Two instances of the rule R2, one instance of the rule R1, one instance of the final rule
C. One instance of the final rule, two instances of the rule R2, one instance of the rule R1.
D. One instance of the final rule

---

30) Let be a RBS formed by WMinitial={(lista 2 1 6 2 3) (elemento 5)}, and the rule shown below. Which is the content of the final WM?

```
(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) Let be the following rule that calculates the Greatest Common Divisor (GCD) of two positive integer numbers. Mark the <u>**CORRECT**</u> answer:

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

A. The rule correctly calculates the GCD and the final WM will contain a fact ‘num’ with the GCD value
B. A stop rule with no salience is needed to prevent the RBS from an endless execution
C. A stop rule with salience is needed to prevent the RBS from an endless execution
D. None of the above answers is correct

---

32) Let be a RBS whose initial WM is (list b a c c a b b a rest), and contains the rule:

```
(defrule R1
    ?a <- (list $?x ?y ?y $?x $?z rest $?m)
  =>
    (retract ?a)
    (assert (list $?x $?x $?z rest $?m ?y)))
```

The contents of the final WM is:

A. (list b a a b b a rest c)
B. (list b a rest c a b)
C. (list b a b a rest c b)
D. None of the above answers

---

33) Given the fact (owners cars a b c owner P cars d owner Q cars e f owner R), which describes the cars and then the owner of these cars, which of the following patterns would be used to obtain the name of an owner who has only one car?

A. (owners $?x cars ?a owner $?z)
B. (owners $? cars ? owner ?z $?)
C. (owners $?x cars ?a owner ?z $?x)
D. (owners $? cars ? owner ?z)

---

34) Let be a RBS whose initial WM is {(list A B C A B C C B A C B A)}, and contains the single rule:

```
(defrule R1
  ?f1 <- (list $?x1 ?y $?x2 ?y $?x3)
         (test (> (length $?x2) 0))
         (test (not (member ?y $?x2)))
  =>
       (retract ?f1)
       (assert (list $?x1 ?y ?y $?x3)))
```

The contents of the final WM is:

A. A list that only contains the letter ‘A’
B. A list that only contains the letter ‘B’
C. A list that only contains the letter ‘C’
D. It will depend on whether the used control strategy is breadth or depth

---

35) Given the fact (prueba 1 2 3 4 5 6 7 8 9 10) and the rule:

```
(defrule R1
    ?f1 <-  (prueba $?a  $?c)
=>
    (retract ?f1)
    (assert (lista $?c)))
```

After the first pattern-matching:

A. No rule instances will be produced
B. 9 rule instances will be produced
C. 10 rule instances will be produced
D. 11 rule instances will be produced

---

36) Given the initial WM, WM={(list 3 5 2 5 3 4 2 9 8 8 9 6) (num 5) (repetitions 0)}, and the following rule that calculates the number of times an element is repeated in a list of numbers.

```
(defrule R1
    ?f1 <- (list $?a ?b $?c)
    ?f2 <- (num ?x)
    ?f3 <- (repetitions ?z)
    (test (= ?b ?x))
=>
     (assert (list $?a $?c))
      (assert (repetitions (+ 1 ?z))))
```

Assuming we want to get a final WM (after the successive application of the rule), in which the fact (repetitions …) appears only once and the value of this fact is the number of times that the number ?x of the pattern (num ?x) appears in the list, mark the **CORRECT** answer to accomplish our objective.

A. The rule is correct
B. It is necessary to add (retract ?f1)
C. It is necessary to add (retract ?f1) and (retract ?f3)
D. It is necessary to add (retract ?f3)

---

37) Let be a RBS composed of WMinitial={list 2 1 5 3)}, and the following rules:

```
(defrule R1                                    (defrule R2
   (declare (salience 200))                        (declare (salience 50))
      ?f <- (list $?x ?z ?y $?w)                      ?f <- (list $?x ?z ?y $?w)
            (test (< ?z ?y))                                (test (>= ?z ?y))
 =>                                            =>
     (assert (list $?x ?z ?y $?w)))                   (assert (list $?x ?z ?y $?w)))

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

after the first pattern-matching, how rules would be ordered in the Agenda?

A. One instance of R1, one instance of the final rule, and two instances of R2
B. Two instances of R2, one instance of R1, and one instance of the final rule
C. Once instance of the final rule, two instances of R2, and one instance of R1
D. Once instance of the final rule

---

38) Given the following LHS of a rule:

```
(defrule r1
        (list $?x $?w ?y ?z $?x)
=>
```

And the fact: (list a b a b c c), how many instances of the rule will be included in the Agenda?:

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

---

39) Let be a RBS with initial Working Memory IniWM={(lista 3 4 5 6 6 6 8 9)}, and the following two rules:

```
(defrule R1                              (defrule R2
  ?f <- (lista $?x ?z ?y $?w)              ?f <- (lista $?x ?z ?y $?w)
        (test (= ?z ?y))                         (test (> ?z ?y))
 =>                                      =>
 (assert (lista $?x ?z ?y $?w)))           (assert (lista $?x ?z ?y $?w)))
```

What will be the contents of the Agenda (Conflict Set) after the first pattern-matching?

A. Two instances of rule R1 and two instances of rule R2.
B. Two instances of rule R1.
C. Two instances of rule R1 and one instance of rule R2.
D. No instance will be inserted in the Agenda.

---

40) Let be the following pattern for the Hanoi towers problem:

(hanoi [tower tw^s $d1^s$ $d2^s$ $d3^s$ $d4^s$]^m) ;; $tw^s \in$ {T1,T2,T3} $d_i^s \in [0\text{-}4]$

, for which two examples of facts are (hanoi tower T1 2 4 0 0 tower T2 0 0 0 0 tower T3 1 3 0 0) (hanoi tower T1 2 4 0 0 tower T2 1 3 0 0 tower T3 0 0 0 0).

Is the following rule correct for moving a disk from tower T3 to any of the other two towers T1, T2 on its left?

```
(defrule move-disk-from-T3-to-Tx
  (hanoi  $?rest1 tower ?Tx  ?d2  $?rest2 0  $?rest3 tower T3 ?d1 ?a ?b ?c)
  (test (or (= ?d2 0) (< ?d1 ?d2)))
 =>
 (assert (hanoi $?rest1 tower  ?Tx   ?d1 ?d2  $?rest2  $?rest3 tower T3 ?a ?b ?c 0)))
```

A. Yes, the rule is correct.
B. The rule would be correct if we check (not (member tower $?rest2)) in the LHS of the rule.
C. Besides the check indicated in response (B), it would be also necessary to check (<> ?d1 0) in the LHS of the rule.
D. Besides the two checks shown in (B) and (C), it would be also necessary to check (> (length $?rest2) 0) in the LHS of the rule.

---

41) Given the initial working memory WM={(lista e b a c d e a) (lista1 a c a d e e f g)} and the rule R1, show the state of the final WM.

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

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

---

42) Indicate the **CORRECT** final result when we execute the following RBS with initial WM={(lista 34 77 34)}:

```
(defrule R1
   (declare (salience 15))
     ?f <- (lista $?x1 ?num $?x2)
 =>
    (retract ?f)
    (printout t "Message 1" crlf))
```

```
(defrule R2
     (declare (salience 20))
       ?f <- (lista ?num $?x ?num)
 =>
        (retract ?f)
        (printout t "Message 2" crlf))
```

```
(defrule R3
  (declare (salience 30))
  =>
  (printout t "Message 3"))
```

A. "Message 3" is shown three times.
B. "Message 2" is shown once and then "Message 3" is shown once.
C. "Message 3" is shown once and then "Message 2" is shown once.
D. "Message 3" is shown once, then "Message 2" is shown once and then "Message 1" is shown once.

---

43) Let be an initial Working Memory WM={(list 6 3 5 1 4 7 2 6 3) (even_num 0)} and the following rule that counts how many even numbers are in a list:

```
(defrule count-even
  ?f1 <- (list $?a  ?b  $?c)
  ?f2 <- (even_num ?p)
  (test (= 0 (mod ?b 2)))
=>
  (assert (list $?a  $?c))
  (assert (even_num (+ 1 ?p))))
```

Assuming we want to get a final WM (after the successive application of the rule), in which the fact (even_num …) appears only once and the value of this fact is the number of even numbers in the list, mark the **CORRECT** answer to accomplish our objective.

A. The rule is correct
B. It is necessary to add (retract ?f1)
C. It is necessary to (retract ?f1) yand (retract ?f2)
D. It is necessary to (retract ?f2)

---

44) We want to write a CLIPS rule that matches the fact (lista a b a a b b c a b c). Which of the following patterns should we include in the LHS of the rule?

A. (lista $? $?x $? $?x ?)
B. (lista $?y ?x ?x $?y $?)
C. (?a $?c)
D. (lista $? $?y ?x ?x ?x $? $?y $? ?x)

---

45) Let be a RBS with initial working memory WMinitial={(lista 4 5 6 6 6 8 4 8)}, and the rules:

```
(defrule R1                                   (defrule R2
   ?f <- (lista    $?x    ?z    ?y    $?w)       ?f <- (lista   $?x  ?z   ?   ?y    $?w)
   (test (<>  ?z  ?y))                           (test (> ?z ?y))
=>                                            =>
   (assert (lista   $?x    ?z    ?y    $?w)))    (assert (lista   $?x     ?z     $?w)))
```

What are the contents of the Agenda after the first pattern matching?

A. Five instances of R1 and one instance of R2
B. Four instances of R1 and no instance of R2
C. Five instances of R1 and no instance of R2
D. Four instances of R1 and one instance of R2

EXERCISES
(open answer questions)

Exercise 1

A hungry chimpanzee is in a room where there is bunch of bananas hanging from the ceiling. In order to reach the bananas, the chimpanzee has to pile up 5 boxes that are spread all over the room. Each box has a different size and they must be stacked from the largest to the smallest size for the robot catch the bananas. The problem consists in finding the sequence of actions that allow the chimpanzee satisfy its appetite. Design a RBS that solves this problem by searching in a state space and helps the chimpanzee pile up the boxes

1. Specify the WM structure. Show an example of an initial WM and the final WM
2. Write the necessary production rules to solve the problem

Exercise 2

A robot is in a room where there are several boxes spread over the floor which must be stack in two different columns. Each box is labelled with a number (1 or 2) indicating the column where the box must be stack on. The problem restrictions are:

- In each move, the robot can stack a maximum of 3 boxes if they belong to the same column or stack one box in each column
- The difference in the number of boxes in both columns must never be greater than 2. For instance, if there are three boxes in COLUMN1 and two boxes in COLUMN2 the robot cannot take two boxes to COLUMN1 (we assume the initial number of boxes in the room satisfy this restriction).

By using the CLIPS language, outline a RBS that solves the above problem. The necessary elements are:

1. Specify the Fact Base (Working Memory) structure. Write down an example of an initial FB and the final FB
2. The necessary production rules to solve the problem

Exercise 3

Given a fact which represents a list of integer and non-ordered numbers, write a single rule (if possible) to get the initial list ordered. For example:

WMinitial: (lista 4 5 3 46 12 10)
WMfinal: (lista 3 4 5 10 12 46)

Show the rule instances that result from applying the rule to the initial WM.

Exercise 4

Write a simple RBS (only one rule, if possible) to find out how many correct numbers a board of the Loteria Primitiva has (this is a weekly lottery in Spain where players choose six numbers out of a total of 49). The initial WM contains two facts, one fact shows the winning number combination and the other shows the six numbers chosen by the player (you can use a global counter to store the number of correct numbers). Here is one example of initial FB:

WMinitial = {(winning-board 2 5 8 13 24 35) (chosen-numbers 3 5 15 24 26 37)}

Let's suppose the player does not introduce numbers in increasing order, would the RBS you've designed above work equally in this case? Why?

Exercise 5

Five members of a family are on one side of a bridge and need to cross to the other side. Each person crosses the bridge in one direction or another. Since they have to cross the bridge at night they must use a flashlight. There is only one flashlight available, which lasts a maximum of 30 seconds. Each family member crosses the bridge at a different speed and therefore it takes a different time (grandfather: 12 seconds, father: 8 sec., mother: 6 sec., daughter: 3 sec., son: 1 sec.). The flashlight is consumed each time a family member crosses the bridge and it cannot be recharged at any time. A **maximum of two people** can cross the bridge at the same time. Since each person moves at a different speed, if two of them cross at the same time they move at the speed of the slowest person. For instance:

Initial state: The five members are on the right side of the bridge
Action 1: The grandfather crosses the bridge to the left side with the son (the flashlight consumes 12 sec.)
Action 2: The son crosses to the right side (the flashlight consumes 1 sec.)
Action 3: The father and the son cross to the left side together (the flashlight consumes 8 sec.)

Design a RBS in CLIPS to find a solution for this problem, if a solution exists. The only permitted action is to cross the bridge (in one direction and/or another, one or two people, etc.)

1) Specify clearly the patterns used to define the fact base. Show the initial and the final fact base.
2) Write the production rules. In the RHS of the rules you can only use the commands "assert", "retract", "printout" and/or "halt". In the LHS you can only use pattern-matching templates and "test".

Exercise 6

Let a RBS be composed of WMinitial={(lista 3 6 8 5 10)} and whose RB is made up of the following rules:

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

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

```
    (retract ?f)                                   (retract ?f)
    (assert (lista ?y $?z))                        (assert (lista $?x)))
```

, assuming we apply a breadth-first strategy (more priority to older rule instances, starting by R1), which will be the final Working Memory?. Show the trace that follows from the inference process.

Exercise 7

Let a RBS composed of WMinitial= {(lista 1 2 3 4)} and whose RB is made up of the following rules:

```
(defrule R1                               (defrule R2
     ?f <- (lista ?x $?z)                   ?f <- (elemento ?x)
=>                                                 (elemento ?y)
   (retract ?f)                                    (test (< ?x ?y))
   (assert (lista $?z))                   =>
   (assert (elemento ?x)))                         (retract ?f)
                                                   (assert (lista-new ?x ?y)))
```

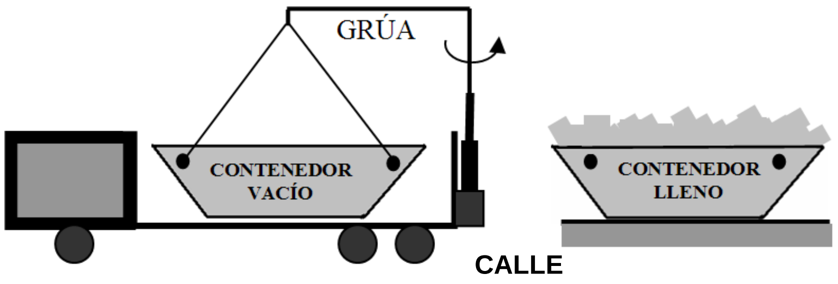
, assuming we apply a breadth-first strategy (more priority to older rule instances, starting by R1):

a) Which will be the final WM? Show the trace that follows from the inference process.
b) If the initial WM were WM= {(lista 1 2 2 4)}, how many facts (lista-new ....) would be in the final WM? Which ones?
c) Assuming again the initial WM WMinitial= {(lista 1 2 3 4)}, if we eliminate the command (retract ?f) from R1, which changes regarding the facts (lista-new ...) will be in the final WM?
d) Assuming again the initial WM WMinitial= {(lista 1 2 3 4)}, if we eliminate the command (retract ?f) from R2, which changes regarding the facts (lista-new ...) will be in the final WM?

Exercise 8

A crane-truck is used to collect and replace the containers full of rubble (constructions rests) on the street. The truck (CAMION) collects the full container (CONTENEDOR LLENO) and leaves an empty container (CONTENEDOR VACÍO) in the same place. The crane-truck transports the empty container and has a crane (GRUA) on it.

The crane can hook a container (either full of rubble, empty or having another container in it), pick up the container and place the container on the truck or on the street. There is not enough room to have directly two containers in the truck or on the street, unless one is into the other.

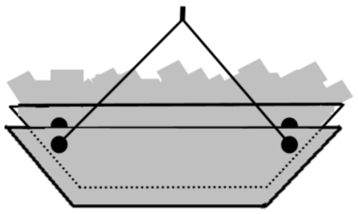


CAMIÓN

The possible actions for the crane are:

Pick-up-container: the crane picks up a container C which can be empty, full or have another container inside (see figure). The container can be situated in the truck or on the street and the crane must be free in order to pick up the container.

Drop-container: The crane drops the held container C (full, empty or with another container inside) in a place (street or truck). ***There is only room for one container in both the truck and the street unless one container is inside the other*** (see figure).



Stack-container: The crane stacks the held container C1 (full or empty) inside container C2, which must be necessarily empty. The crane must be free and container C2 can be in the truck or on the street.

Unstack-container: The crane unstacks container C2 (full or empty) which is inside container C1. After this operation, container C1 will remain empty. The crane must be free and the stacked containers can be on the street or in the truck.

Do not write any other actions than those described here. You don't have to consider the operations to grasp and release container, i.e those actions which refer to tie up and untie the crane wires to a container as we will assume these operations are included in the pick-up and drop container.

A possible initial situation is "***There is an empty container (Container 1) in the truck and a full container (Container 2) on the street***". Write out a RBS in CLIPS to obtain the necessary sequence of actions which achieves the following final situation "***The full container is in the truck and the empty one on the street***".

1. Specify clearly the **patterns** used to define the Fact Base.
2. Show the initial and the final fact base
3. Specify the production rules

Do not write any type of actions other than those specified in the problem description. Do not write procedural code in the rules RHS.

Rules must be general enough (as many rules as possible actions, approximately). Do not write specific rules for a particular container or for concrete situations.

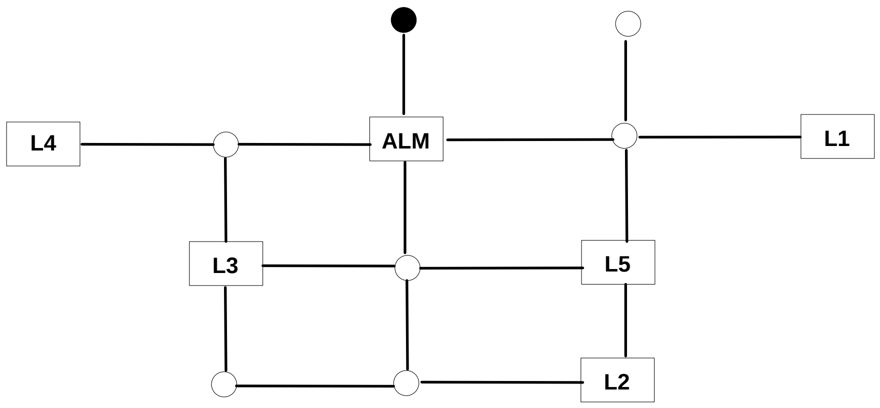
Exercise 9

Given a list of integer numbers, we want to obtain a new list where those numbers whose values do not coincide with their ordering positions are replaced by 0 (we assume the first number in the list takes up position number 1). For example, given the list (list 15 2 4 6 5 3 7 8 15) the new list would be (list 0 2 0 0 5 0 7 8 0). Write a single rule in CLIPS to perform this task.

Exercise 11

We have 5 different stalls that order soft drink bottles and a distributor robot that delivers the bottles to the stalls. The robot uses a trolley for the delivering, which has a maximum capacity of 3 bottles. The robot must physically reach the stall in order to deliver the order in such a stall.

There exists a network of connections as the one shown in the figure. L1,..., L5 stand for the 5 stalls. ALM is the store where bottles are stored. The rest of nodes are intermediate points where the robot can be located. The robot can move between any pair of locations shown in the figure. The robot moves to any location directly connected to its current position in one move action.



When the robot is at the store (ALM), it can refill the trolley with new soft drink bottles. We will assume the robot always refills the trolley to the maximum of its capacity.

An example of initial state of the problem is: the robot is at any of the locations in the figure and the trolley is empty; one or several stalls place an order of a number of bottles. Stalls can order any number of bottles and the robot must deliver the order in one delivering or until out of stock. The final situation is as follows: all of the orders must have been delivered, the robot is back at the store (ALM) and the trolley is full of bottles.

Example of initial situation:

- The robot is located at the point shaded in black
- Stall L1 orders two bottles, L3 orders one bottle and L5 orders five bottles

Design a RBS to solve this problem with any possible initial configuration

- Specify the structure of the facts (patterns) that you will use in your design
- Specify the set of rules according to the design of the patterns

Exercise 16

We have three towers (A1, A2 and A3) each containing a number of disks. Disks are all of different size. We will assume that the largest disk is of size 99. There exists a fourth tower (B) which is initially empty.

A disk d can be put in towers A1, A2 and A3 provided that the size of d is smaller than the below disk or that the tower is empty. However, a disk d' can only be put in tower B if the size of d' is greater than the below disk or tower B is empty.

We have a robot-arm to move one disk from one tower to another. The robot-arm can only access the top disk of any tower.

Design a RBS to have all disks properly ordered in tower B (from the largest to the smallest disk) for any possible initial situation; that is, for any number of disks in towers A1, A2 and A3 and tower B initially empty. An example of initial situation could be:

1) tower A1 contains disks 6 9 10
2) tower A2 contains disks 3 7 8
3) tower A3 contains disks 1 2 4
4) tower B is empty

, where numbers identify the disk size. In the final state, all disks should be ordered in tower B: {10 9 8 7 6 4 3 2 1}.

The design of a RBS that performs the minimal search possible will be valued.

Exercise 21 (exam 2013)

A farmer has to cross a river with his fox, goose and grain. Each trip, his boat can only carry himself and one of his possessions. Additionally, an unguarded fox eats the goose and an unguarded goose eats the grain. Consequently, the farmer cannot leave the fox alone with the goose nor the goose alone with the grain.

We want to design a RBS to cross the farmer, fox, goose and grain from one bank to the river to the other. Assume that all of them are in bank A of the river in the initial state and have to cross to bank B.

a) Assuming the following pattern to represent the information of one state in a state-based representation problem:

   (problem farmer x^S fox x^S goose x^S grain x^S) where $x^S \in \{A, B\}$

   NOTE: Use only command 'assert' in the RHS of the rules

   a.1) By using CLIPS language, write a rule to cross the farmer alone from bank A to bank B of the river.

   a.2) By using CLIPS language, write a rule to cross the fox from bank A to bank B of the river.

```
=>
    (assert (problem farmer B fox B goose ?g grain ?t)))
```

b) Suppose the river is too wide to cross it from one bank to the other without intermediate stops. Assume we have {I1, I2, …, In} islets in the middle of the river.

b.1) without modifying the pattern in (a), add the necessary facts to the Working Memory to represent this new information so that the move rules are independent of the origin/destination (bank of the river, islet).

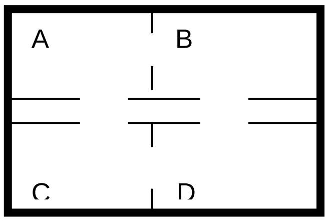
b.2) considering the new information in b.1), modify the rule in a.2) to cross the fox from any origin to any destination, either any bank of the river or any islet.

Exercise 21 (January 2015)

One floor of a building is distributed in 4 rooms (A, B, C and D) and a corridor which connects with the four rooms (see figure below). There are 2 robots which are responsible of delivering a cup of tea or a cup of coffee to several customers who are in the rooms. Robots are only allowed to move within a delimited area: robot 1 can only move across room A, room B and the corridor; robot 2 can only move across room C, room D and the corridor.

A vending machine of tea and coffee is found in room A. A cabinet with cups is found in room C. People who order a cup of tea or coffee can only be in rooms B or D. Robots can only hold a cup at a time. A robot X can pass on a cup (empty or full) to a robot Y if robot Y is not holding another cup.

The operators of this problem are: (a) the robot grabs an empty cup; (b) a robot pass on a cup (empty or full) to the other robot; (c) the robot fills the cup with tea or coffee according to the customer order; (d) a robot moves to an adjacent room or to the corridor; and (e) a robot delivers the cup of tea or cup of coffee ordered by the customer.



Given the following pattern to represent the dynamic information of this problem:

(problem robot 1 pos ?p1 carries ?l1 robot 2 pos ?p2 carries ?l2 order [customer ?p ?t]^m)

where, ?p1 ∈ {A, B, COR}, ?p2 ∈ {C, D, COR}, ?p ∈ {B, D}, ?l1, ?l2 ∈ {nothing, empty, tea, coffee} and ?t ∈ {tea, coffee}

a) (0.4 points) Write an initial working memory for this problem where robot 1 is in room A, robot 2 is in room D and there are two customers who have ordered a cup of coffee and a cup of tea, respectively.

b) (0.2 points) Write an example of final working memory in which all of the orders have been served.

c) (0.8 points) Write a single rule to move any robot between two points within its allowed area.

d) (0.8 points) Write a single rule to pass on a cup (empty or full) between the two robots.

e) (0.8 points) Write a single rule to deliver the cup of tea or cup of coffee ordered by a customer.

Exercise 22 (January 2016)

We have three storehouses (A, B and C) in a city location, each having a set of packages to be transported to any of the other two stores. Hence, store A may have packages for B and/or C; store B may have packages for store A and/or C; and store C may have packages whose destination is store A and/or B. The problem goal is to deliver all packages to their destination store.

For the package transportation, there is only one truck available which can keep 10 packages at most. The truck can move between any pair of storehouses. When the truck is at store X, packages located at store X and whose destination is a store other than X can be loaded into the truck. Likewise, when the truck is at store X, only the packages whose final destination is X can be unloaded from the truck.

Example of initial situation:

- There are 7 packages in store A: 4 of them go to B and 3 go to C.
- There are 10 packages in store B: 7 of them go to A and 3 go to C.
- There are 6 packages in store C: 3 for A and 3 for B.
- The truck is initially in store A and it is empty.

Given the following pattern to represent the dynamic information of the problem;

(transport [store ?cit [destin ?dest ?num]^m]^m truck ?loc [?dest_pack ?num_pack]^m total ?tot)

, where:

?cit, ?loc $\in$ {A,B,C}	
?dest $\in$ {A,B,C} such that ?dest $\neq$?cit	;; destination
?num $\in$ INTEGER	;; number of packages to destination, even when the number is 0
?tot $\in$ INTEGER	;; overall number of packages in the truck

```
?dest_pack ∈ {A,B,C}                                  ;; destination (only if there are packages for
                                                         such destination)
?num_pack ∈ INTEGER such that ?num_pack ≠ 0   ;; number of packages to destination
                                                         (only if it is not 0)
```

NOTE 1: If the truck is not carrying packages for store X then the label [X 0] does not appear in the fact
NOTE 2: Given a store X, we only represent the packages to be moved to another store (we do not represent the packages of X)
NOTE 3: If necessary, you can add a static fact in any of the following questions.

a) (0.5 points) Write the initial fact base that represents the information described above.

b) (1 point) Write a single rule that loads into the truck all the packages in store X that go to a destination Y. Assume the truck does not have a priori packages for Y. The restriction about the max capacity of the truck must be satisfied

c) (0.8 points) Write a single rule that displays a message for each destination for which the truck is not carrying packages. You must show a message like: "The truck has not packages for destination XXXX ", for each destination that satisfies this condition.

d) (0.7 points) Write a single rule that unloads from the truck in store X the packages whose destination is X. The rule must be valid for any destination and the new resulting fact must not contain the label of the destination nor the number of packages; that is, once the packages are unloaded at X, the label [X 0] of the truck must not appear in the new fact.

Exercise 23 (January 2017)

We want to design a RBS for handling the collection of stickers of a number of children. A sticker is represented by an alpha-numerical identifier (e.g., A1, B3, C2, etc.). A child has a number of stickers (including repeated stickers) and the number of children is not limited in the application. An example is:

- Kid 1 has the stickers: A2 A4 A5 B1 A2 B3
- Kid 2 has the stickers: B3 A4 C2 C1 B3 C2
- Kid 3 has the stickers: C2 C4 B1 A2

The dynamic information of the problem is represented with facts that follow this pattern:

(collections [kid ?n [?id-sticker]^m fkid ?n]^m)

where:

?n ∊ INTEGER ;; kid identifier
?id-sticker ∊ {A1, A2, B1,…} ;;sticker identifier

Answer the following questions:

a) (0.3 points) Write the facts of the Working Memory that represent the above example.
b) (1 point) Write a single rule for two kids to swap a sticker. Swapping a sticker is only allowed if kids hand over a repeated sticker of their collections and they get in return a sticker which is not in their collections.
c) (0.7 points) Write a single rule to display the children that have a sticker exactly three times in their respective collections. The rule must display a message per kid and sticker; example: "The kid " ?n " has the sticker " ?x " three times".
d) (1 point) Let's suppose that the WM contains facts that follow the pattern (special ?id-sticker) to indicate that the sticker identified with ?id-sticker is a special sticker. Write a single rule to calculate the number of children who have at least two special and different stickers. The result of executing the rule will be a fact that follows this format: (special-list [?n]^m), where ?n is the identifier of a kid who has at least two different special stickers. The kid identifier must appear only once in the list (even though the kid has several special stickers). Assume that the WM contains several facts with the format (special ?id-sticker) and the fact (special-list).

Exercise 24 (January 2019)

In an airport, several pickup luggage trains are available to take the suitcases (bags) from the check-in area to the plane assigned to the flight of the suitcases. A checked bag carries the label of its flight. Initially, the trains are not assigned to any flight. The flight assigned to a train will be the flight of the first bag loaded in the train. A train can only transport bags for one flight, and each flight is assigned to only one train.

The pattern to represent the dynamic information of a problem state is:

(airport [TRAIN num^s dest^s bag^m]^m) where

num ∈ INTEGER;; is the train identifier
dest ∈ {none, F1, F2, F3, …} ;; is a symbol that represents the flight assigned to a train (initially, the flight of the train is unknown and the value of this field will be none).
bag ∈ {B1, B2, B3,…};; is a symbol that represents the bag identifier (initially, this field is empty)

A possible initial situation for this problem is:

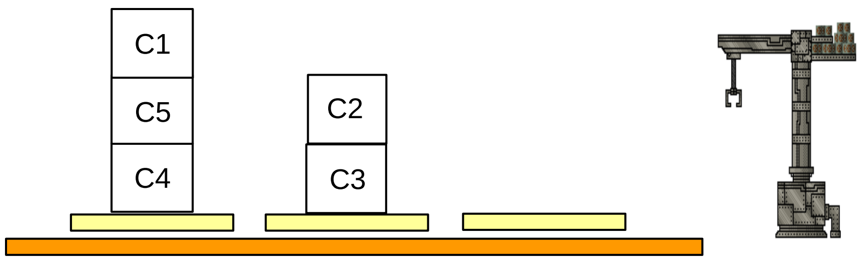
- There are five bags (B1, B2, B3, B4 and B5); the first two bags are cheked for flight F14; the third bag is checked for flight F2 and the last two bags go to flight F10
- There are three available luggage trains in the airport. Initially, the trains are empty.

We want to solve this problem through a search process in a state space by designing a RBS in CLIPS:

1) (0.7 points) Write the facts of the Working Memory (WM) that represents the situation given above. Specify also the patterns that you need to represent the static information of the problem and include in the WM the facts associated to these patterns.
2) (1 point) Write a single rule to load the first bag in a train and assign the flight of the loaded bag to the train.
3) (0.8 points) Write a single rule to load a bag in a train when the train is already assigned to a flight. You must check that the flight of the bag is the same as the flight of the train and that the bag is not already loaded in the train.
4) (0.5 points) Let be the pattern (flight f^s), where $f^s \in$ {F1, F2, F3, …} is a symbol that represents the flight identifier, and let's assume a fact that denotes a particular flight. Write a single rule that displays all the bags loaded in the train assigned to such a flight. The rule must display only one message of the type: "The bags X X X are loaded in the train Y".

Exercise 25 (January 2019)

In the harbor of a city there exists a series of containers stacked in several piles. An example of an initial situation is shown in the figure, where there is a maximum of three piles: containers C1, C5 and C4 are in pile 1, containers C2 and C3 are in pile 2 and the third pile is empty. Containers are arranged in a pile such that a container Ci is on top of a container Cj if the weight of Ci is smaller than the weight of Cj.



There is also a crane that can grab a tower of n containers, where n <= 3, and n can be all the containers of a pile. We will assume that the containers grabbed by the crane are in the same position as they appear in the pile. For example, if the crane grabs the tower of containers [C1 C5], container C5 will be the base of the tower and C1 will be the top of the tower. If the crane grabs the tower of containers [C1 C5 C4], the base of the tower will be C4 and the top container of the tower will be C1.

The tower of containers grabbed by the crane can be placed in an empty pile or over a container Ci whenever the weight of the base container is smaller than the weight of Ci. For example, let's suppose the crane grabs the tower [C1 C5]; then, the crane can place [C1 C5] on top of C2 if the weight of C5 is smaller than the weight of C2.

The pattern to represent the dynamic information of a problem state is:

(harbor [pile num^s cont^m endpile]^m crane cra^m) where

num $\in$ INTEGER ;; is a number that identifies the pile
cont $\in$ {C1, C2, C3,…} ;; is a symbol that represents the container; this field represents the containers of the pile
cra $\in$ {C1, C2, C3,…} ;; represents the containers grabbed by the crane

We want to solve this problem through a search process in a state space by designing a RBS in CLIPS:

1) (0.7 points) Write the facts of the Working Memory (WM) that represents the situation given above. Specify also the patterns that you need to represent the static information of the problem and include in the WM the facts associated to these patterns.
2) (0.7 points) Write a rule for a crane to grab all the containers of a pile. The rule must satisfy the restriction that the number of containers of the pile is <= 3.
3) (0.8 points) Assuming the crane is holding a tower of containers, write a rule to place the tower of containers in a non-empty pile that has at least one container. The rule must satisfy the constraint on the weights explained above in the wording of the problem.
4) (0.8 points) Write a rule that displays the containers of a pile. The rule must show a message like "The container Y is in pile X" for every container of a pile.

Exercise 26 (October 2019)

A factory manufactures three different ranges (categories) of a given product: mid range, high range and supreme range. Making a product of each range requires a different number of pieces of two types, A and B:

- A mid-range product is made with 3 pieces of type A and 1 piece of type B
- A high-range product is made with 2 pieces of type A and 2 pieces of type B
- A supreme-range product is made with 3 pieces of type B

The factory keeps pieces of type A and type B in stock. We also know the factory has a number of product orders of each range. The pattern that represents the dynamic information of this problem is:

(factory [piece type^s num_p^s]^m [product-order range^s num_o^s]^m)

type ∈ {A,B}
num_p ∈ INTEGER ;; number of pieces of the corresponding type
range ∈ {mid, high, supreme}
num_o ∈ INTEGER ;; number of product orders of the corresponding range

An example of initial situation is:

- The factory has 10 pieces of type A and 15 pieces of type B in stock
- The product orders are: 2 mid-range products, 1 high-range product, 1 supreme-range product

1) (0.4 points) Write the facts of the Working Memory (WM) that represents the situation given above. Specify also the patterns that you need to represent the static information of the problem and include in the WM the facts associated to these patterns. NOTE: we recommend keeping the same order of the two types of pieces (A and B) in the dynamic pattern and in the necessary static patterns.
2) (0.5 points) Write a rule to deliver one product of any range. The effects of delivering a product are (a) decrementing the number of available pieces in the factory, and (b) decrementing the number of product orders of the corresponding range.
3) (0.5 points) Write a rule to deliver all the product orders of one same range.
4) (0.6 points) Write a rule that determines, for two product orders of different range, if there are only pieces for delivering one of them. The rule must display a message of this type: "The factory has only pieces in stock for delivering one product of range XXX or of range YYY".

Exercise 27 (January 2020)

We want to form two groups of people: a group of people who speak Russian and another group who speak Chinese. Several persons attend the call and bring along certificates that confirm their proficiency in one or the two languages. The language proficiency is measured in five levels, from 1 to 5, being 1 the lowest level and 5 the highest level.

- P1 certificates Chinese level 3 and Russian level 1.
- P2 certificates Russian level 4.
- P3 certificates Russian level 1 and Chinese level 2.
- P4 certificates Chinese level 3.
- P5 certificates Russian level 3.
- P6 certificates Chinese level 2 and Russian level 5.
- P7 certificates Chinese level 4.
- P8 certificates Russian level 3 and Chinese level 2.

The pattern for group formation is:

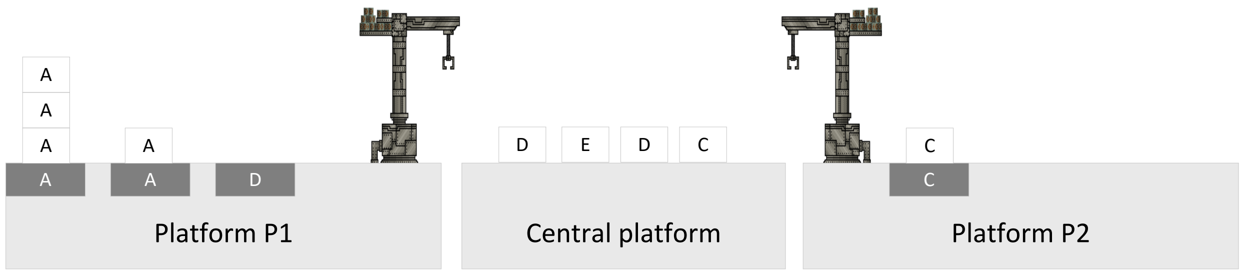
(groups Russian p^m Chinese q^m) where p,q ∈ {P1,P2,P3,P4,P5,P6,P7,P8}

1) (0.5 points) Write the facts of the Working Memory (WM) that represents the situation given above. Specify also the patterns that you need to represent the static information of the problem and include in the WM the facts associated to these patterns.
2) (0.8 points) Write a rule to add a person to the group of Russian or Chinese. The rule must check the person has a certificate that confirms the corresponding language proficiency at level 2 at least. The rule must also check the person did not sign up already for any group.
3) (0.7 points) Write a rule that displays a message indicating the number of people in each group when there is at least three people in each group.

Exercise 28 (November 2020)

The port of a city has a container handling system that works as follows:

1) There is a central platform, and two platforms named P1 and P2. Platforms P1 and P2 each has a crane (see the figure below).
2) A number of containers are in the central platform. All containers are accessible, that is, they are all on the floor. Containers belong to a class (A, B, C, D, …).
3) Every container is associated to a single type, either P1 or P2, which indicates its destination platform. The type Pi of each container is determined by the class of the container. That is, containers of a class X are meant to go to a particular platform Pi.
4) The crane of a platform Pi can only pick up one container at a time from the central platform whose destination is Pi.
5) The crane of a platform Pi drops a grabbed container of class X in an existing heap of the same class X in Pi. Heaps cannot hold more than 3 containers. In the example of the figure, two heaps of class A and one heap of class D stand in platform P1; one of the heaps of class A has already 3 containers, the other heap of class A has only one container and the heap of class D is empty.



The structure of the dynamic pattern is the following (you are not allowed to change the structure of this pattern):

(port b^m platform T1 [heap c^s n^s]^m crane T1 p^s platform T2 [heap c^s n^s]^m crane T2 p^s)

where:

b^m ∈ {A, B, C, D, …} each element in **b**^{m} denotes a container of the corresponding class in the central platform
c^s∈ {A, B, C, D, ...} class of the heap, it denotes the class of the containers the heap can stack
n^s ∈ [0 3] number of containers in the heap
p^s ∈ {empty, A, B, C, …} class of the container grabbed by the crane or empty otherwise

NOTE: For a crane to drop a grabbed container in a heap, a heap of the same class as the container must exist in the platform. The heap can be empty (0 containers). An ***empty heap*** and a ***non-existing heap*** is not the same. In the latter case, the heap does not appear in the pattern.

Initially, there are no heaps in any of the two platforms. An empty heap will be created with a CLIPS rule.

Answer the following questions using CLIPS language and assuming a GRAPH-SEARCH expansion:

a) (0.3 points) Write the **initial Working Memory** taking into account that the central platform contains the following containers: A E B B C D B B A E D. We know that containers of class A and D go to platform P1, and containers of class B, C and E go to platform P2. Neither of the two platforms contains a heap, not even an empty heap, and the cranes are empty. For the static facts, pick the pattern you consider to be the most appropriate.

b) (0.5 points) Write only one rule "initial_heap" that creates an empty heap in a platform Pi under certain conditions. Specifically, if there is a container of class X in the central platform, there is no heap of class X in its destination platform Pi and the crane is empty, the rule will create an empty heap in Pi. An empty heap is represented in the figure with a dark grey rectangle.

c) (0.7 points) Write only one rule "pick_up_container" that allows an empty crane of a platform Pi to pick up a container from the central platform whose destination is Pi. The rule must satisfy the condition that a heap of the same class as the container already exists in Pi and the heap is not full (i.e., it has less than three containers).

d) (0.5 points) Write a "stop" rule. This rule will trigger when all the containers are in their corresponding platforms. The rule will display the total number of created heaps.

Exercise 29 (November 2021)

We have a series of boxes scattered on the floor each of which is tagged with an integer numeric value. The same integer label can be shared by two or more boxes. For instance: 17, 5, 6, 22, 5, 4, 7, 12, 6, 1, 21, ….

We wish to stack boxes in different towers in decreasing order of their labels such that a box labeled as M can only be put on top of another box labeled as N provided that M <= N. The conditions on the formation of the towers are:

1) There is an indefinite number of towers.
2) There are two types of towers, those that store boxes tagged with an even number and those that store boxes labeled with an odd number.
3) There is an indefinite number of boxes in each tower.

The patter you must follow to represent a state of this problem is:

$$(\texttt{problem [tower bt}^{m}]^{m}\texttt{ floor bs}^{m})$$

where bt $\in$ INTEGER and b bs $\in$ INTEGER.

NOTE: The predicate `(evenp <arg>)` returns TRUE if `<arg>` is an even number.

By using a GRAPH-SEARCH strategy in CLIPS, answer the following questions:

1) Write an initial Working Memory that represents the following situation: two towers, one contains boxes labeled as 13 and 17 and the other one contains only a box labeled as 22. The boxes 17, 5, 6, 22, 5, 4, 7, 12, 6, 1, 21 are on the floor.

2) Assuming that various towers are already created in the problem, and that all of the towers contain at least one box, write a single rule that picks up a box from the floor and puts it on top of an existing tower that is allowed to store such a box (i.e., even box in even tower, odd box in odd tower).

3) Assuming that various towers are already created in the problem, and that all of the towers contain at least one box, we wish to design a rule to unstack a box with label X from a tower T (unstack means to grab the box which is on the top of the tower) provided that there are one or more towers different from T whose top box is also labeled as X. Write a single rule that attains this operation.

4) Assuming that various towers are already created in the problem, and that all of the towers contain at least one box, write a single rule that outputs a message if it exists AT LEAST TWO boxes on the floor labeled as X such that none of the existing towers has boxes with the same label X. The rule must show the following message: "There are at least two boxes on the floor labeled as X and none of the towers contains boxes with label X".