SE – Laborator 8

; Algebra bool si descoperirea/demonstrarea de teoreme

;a+b=b+a a+(b+c)=(a+b)+c
;a*b=b*a a*(b*c)=(a*b)*c
;a*(b+c)=(a*b)+(a*c) a+(b*c)=(a+b)*(a+c)
;a*1=1
;a+0=a
;a*^a=0
;a+^a=1

```
(defmodule MAIN (export ?ALL)

(deftemplate bool (multislot expr))
(deftemplate eq (multislot A) (multislot B))

(deffacts base (bool (expr 0)) (bool (expr 1)))

(defrule init (declare (salience 200)) =. (set-strategy breadth))

(defrule simetria
        (declare (saleince 50))
        (eq (A $?x) (B $?y))
=>
        (assert (eq (A $?y) (B $?x))

(defrule tranzitivitate
        (declare (saleince 50))
        (eq (A $?x) (B $?y))
        (eq (A $?y) ( B $?z))
=>
        (assert (eq (A $?x) (B $?z))))

(defrule comut_or
        (bool (expr $?A))
        (bool (expr $?B))
=>
        (assert (bool (expr + $?A $?B)))
        (assert (bool (expr + $?B $?A)))
        (assert (eq  (A + $?A $?B)  (B + $?B $?A))))

(defrule comut_and
        (bool (expr $?A))
        (bool (expr $?B))
=>
        (assert (bool (expr * $?A $?B)))
        (assert (bool (expr * $?B $?A)))
        (assert (eq  (A * $?A $?B)  (B * $?B $?A))))

(defrule asoc_or
        (declare (salience –10))
        (bool (expr $?A))
        (bool (expr $?B))
        (bool (expr $?C))
=>
```

```
        (assert (bool (expr + $?A + $?B $?C)))
        (assert (bool (expr + + $?A + $?B $?C)))
        (assert (eq (A + $? A + $?B $?C) (B + + $?A $?B $?C))))

(defrule asoc_and
        (declare (salience –10))
        (bool (expr $?A))
        (bool (expr $?B))
        (bool (expr $?C))
=>
        (assert (bool (expr * $?A * $?B $?C)))
        (assert (bool (expr * * $?A * $?B $?C)))
        (assert (eq (A * $? A * $?B $?C) (B * * $?A $?B $?C))))

(defrule distributivitatea
        (declare (salience –10))
        (bool (expr $?A))
        (bool (expr $?B))
        (bool (expr $?C))
=>
        (assert (eq (A * $?A + $?B $?C) (B + * $?A $?B * $?A $?C)))
        (assert (eq (A + $?A * $?B $?C) (B * + $?A $?B + $?A $?C))))

(defrule zero
        (bool (expr $?A))
=>
        (assert (eq (A + $?A 0) (B $?A))))

(defrule unu
        (bool (expr $?A))
=>
        (assert (eq (A * $?A 1) (B $? A))))

(defrule non
        (bool (expr $?A))
=>
        (assert (bool (expr ^ $?S))
=>
        (assert (bool (expr ^ $?A)))
        (assert (eq (A * ^ $?A $?A) (B 0)))
        (assert (eq (A + ^ $?A $?A) (B 1))))

;--------------test:

(deffacts test
        (bool (expr a))
        (bool (expr b)))

(defrule test
        (declare (saleince 100))
        (eq (A + a b) (B + b a))
=>
(printout t “Asa e!”)
(halt))
```

Teme:

Termen de predare: doua saptamani (se va prezenta in cadrul laboratorului)- incepand din 22 Noiembrie 2002. Tema am prezentat-o in laboratoarele de Sisteme Expert din datele de 21 si 22 Noiembrie 2002.
Problema se va rezolva in CLIPS

Enuntul problemei: sa se rezolve problema minesweeper. Se dau ca fapte initiale dimensiunea tabelei si numar de bombe existente. Se vor scrie un set de reguli astfel incat calculatorul sa joace minesweeper si sa descopere bombele. Daca nu stie ce sa faca calculatorul va intreba utilizatorul ce este la o casuta/celula aleasa aleator (nedescoperita inca). O celula poate lua valori de la 0 la 8 (numarul de bombe care se afla in jurul celulei respective) sau valoarea de bomba. Cand se “calca” peste o bomba, programul se termina. Atunci cand calculatorul gaseste o bomba, el va tipari un mesaj de genul “gasit bomba la celula x y”. Raspunsurile pe care le va primi calculatorul de la utilizator trebuie sa fie corecte, ajutandu-l pe calculator sa rezolve problema minesweeper.