Proiect PAC-MAC

Ungureanu Iulia Iarina

9 decembrie 2022

Rezumat

Cerință: Rezolvarea jocului "Lights out" în mace4.

1 Introducere

Lights Out este un joc electronic lansat de Tiger Electronics în 1995. Jocul constă dintr-o grilă de becuri de n pe n. Când începe jocul, se aprinde un număr aleatoriu sau un model stocat al acestor becuri. Apăsând pe oricare dintre becuri, acesta și becurile adiacente vor comuta. Scopul puzzle-ului este de a stinge toate luminile, de preferință în cât mai puține apăsări de butoane.

Consideram stările ca valori binare : 0 reprezinta un bec stins, iar 1 un bec aprins. Comutarea intre stări se face atunci când jucătorul apasă un bec (i,j). Acesta împreuna cu toți vecini sai (sus (i-1,j), jos (i+1,j), dreapta (i,j+1), stânga (i,j-1)), își schimba starea în funcție de starea actuala ("stins" - "aprins" , "aprins" - "stins").

La începutul jocului matricea are valori de 0 și 1 puse la întâmplare. După un set de apăsări de becuri, aceasta trebuie sa ajungă într-o stare anume setata în funcție de regulile jocului.

În aceasta lucrare o sa consideram ca în starea finala matricea are toate becurile stinse:

$$FinalMatrix = O_n$$

2 Implementare

2.1 Înțelegerea problemei

Pentru o matrice pătratica de dimensiune n=3 o sa consideram luminile inițiale ca o lista "START" de biți de 9 elemente cu valoarea 1 pentru aprins și 0 pentru stins.

Apăsarea unui bec de un număr impar de ori este echivalent cu o singura schimbare de stare (pentru becul respectiv și vecinii sai), iar apăsarea de un număr par de ori este echivalent cu nici o schimbare de stare. Deoarece noi dorim sa aflam soluția cu cele mai puține mișcări o sa consideram ca nu se efectuează apăsări repetate ale aceluiași bec, altfel ar exista o infinitate de soluția corecte. Soluția poate fi reprezentat ca un vector de biți "G" cu G(x)=1 pentru o apăsare pe becul "x" sau 0 altfel.

Efectul unei apăsări pe buton "x" poate fi scris și el ca o listă "f(x)" de 9 de biți:0 starea becului nu se schimbă și 1 dacă starea becului se schimbă. Astfel putem descrie rezultatul unei serii de apăsări de butoane ca fiind un vector egal cu suma celor doi vectori modulo 2. Aceasta înseamnă că rezultatul pentru un anumit bec este calculat prin adăugarea intrărilor corespunzătoare din cei doi vectori și că dacă avem doi de unu, atunci rezultatul 1+1 ar trebui să fie 0 (o lumină care este aprinsă și este schimbată de o apăsare de buton, se va stinge ulterior). De asemenea, puteți considera aceasta o operațiune „Sau exclusivă" pe un cuvânt de 9 biți.

Noi știmi din ipoteza faptul ca la final toate becurile trebuie sa fie stinse, deci putem defini un vector FINAL de 9 elemente cu valorile "0". Cunoaștem de asemenea și starea inițiala a matricei adică vectorul START. Trebuie sa aflam lista "G" cu apăsările care trebuie efectuate pe matrice știind ca:

$$\forall x final(x) = \neg(start(x) \iff f(x))$$

Însumarea este comutativă, cu alte cuvinte, nu contează în ce ordine adunăm lucrurile. Aceasta înseamnă că, de asemenea, nu contează în ce ordine sunt adunați vectorii și, prin urmare, efectul mai multor apăsări de butoane este pur și simplu suma efectelor lor individuale în orice ordine. O astfel de însumare a vectorilor poate fi scrisă ca o multiplicare matriceală.

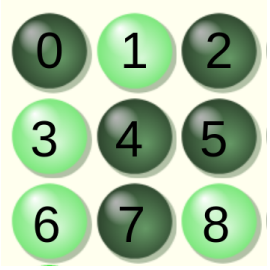

Figura 1: Matrice de becuri 3 x 3

- Fie F o matrice în care F(j,i) este 1 dacă starea becului i este schimbată de butonul j, sau 0 în caz contrar.
- Fie START vector de poziție de pornire, adică start(i) este 1 dacă becul i este aprins la început, 0 în caz contrar.
- Fie G lista de buton pe care le apăsăm, adică G(j) este 1 dacă apăsăm butonul j sau 0 dacă nu.
- Fie FINAL vectorul de poziție finală, adică FINAL(i) este 1 dacă becul i ar trebui să fie aprins la sfârșit, 0 în caz contrar.

Starea de becului i este dată de starea ei de la început, plus efectele tuturor apăsărilor de butoane pe care le au asupra ei. Din punct de vedere algebric, aceasta este:

$$final(i) = start(i) + \sum_{i=0}^{n} g(i)f(j,i)$$

2.2 Implementare în Mace4 pentru o matrice 3x3

O sa rezolvam matricea de becuri din Figura 1, care are becurile 1, 3, 6 și 8 aprinse, restul stinse. Indexarea se face de la 0 la 8 deci o sa adaugăm "assign(domainsize, 9)". O sa începem prin declarare vectorului "start" care conține "True" dacă matricea inițiala are un bec aprins pe poziția respectiva și "False" altfel. Apoi o sa declaram vectorul Final ca fiind "False" pe toate pozițiile conform ipotezei problemei:

```
-start(0).
start(1).    %1 aprins
-start(2).
start(3).     %3 aprins
-start(4).
-start(5).
start(6).   %6 aprins
-start(7).
start(8).   %8 aprins

 all x (-final(x)).
```

Pentru a genera matricea F putem hard-coda valorile astfel: Dacă apăsam pe butonul x (G(x)este True) asta implica ca în matricea f vom avea pe linia x și coloana y "True" dacă y este vecinul lui x, sau chiar x. Dacă butonul x nu este apăsat adică (G(x) este False) atunci toată linia x din F este Falsa.

$$\forall x, y(\neg g(x) \Rightarrow \neg f(x,y))$$

$$\forall x, y(g(x) \Rightarrow f(x, y_{vecin}) \wedge (\neg f(x, y_{nuestevecin})))$$

```
 -g(x) -> - f(x,y).

g(0) -> f(0,0) & f(0,1) & f(0,3) & (all y (y!=0 & y!=1 & y!=3 <-> -f(0,y) )).
g(1) -> f(1,0) & f(1,1) & f(1,2) & f(1,4) & (all y (y!=0 & y!=1 &y!=2 &y!=4 <-> -f(1,y))).
```

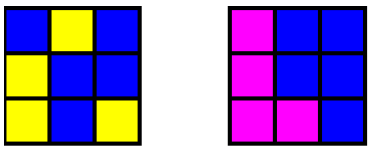

Figura 2: Exemplu din program

```
g(2) -> f(2,1) & f(2,2) & f(2,5) & (all y(y!=1 & y!=2 &y!=5 <-> -f(2,y))).
g(3) -> f(3,0) & f(3,3) & f(3,4) & f(3,6) & (all y(y!=0 & y!=3 &y!=4 &y!=6 <-> -f(3,y))).
g(4) -> f(4,1) & f(4,3) & f(4,4) & f(4,5) & f(4,7) & (all y(y!=1 & y!=3 &y!=4 &y!=5 &y!=7 <-> -f(4,y))).
g(5) -> f(5,2) & f(5,4) & f(5,5) & f(5,8) & (all y(y!=2 & y!=4 &y!=5 &y!=8 <-> -f(5,y))).
g(6) -> f(6,3) & f(6,6) & f(6,7) & (all y(y!=3 & y!=6 &y!=7 <-> -f(6,y))).
g(7) -> f(7,4) & f(7,6) & f(7,7) & f(7,8) & (all y(y!=4 & y!=6 &y!=7 &y!=8 <-> -f(7,y))).
g(8) -> f(8,5) & f(8,7) & f(8,8) & (all y(y!=5 & y!=7 &y!=8 <-> -f(8,y))).
```

Știm din calculele de matrice ca suma tuturor linilor din f cu startul modulo 2 trebuie sa fie egale cu finalul. Aceasta suma se poate scrie ca un sau exclusiv astfel:

$$\forall x final(x) = \neg(start(x) \iff f(x,0) \iff f(x,1)... \iff f(x,n))$$

```
all x(
     -(
       (((((((start(x) <-> f(0,x)) <-> f(1,x)) <-> f(2,x))<-> f(3,x) )
              <-> f(4,x))<-> f(5,x))<-> f(6,x))<->f(7,x))<-> f(8,x)
      ) -> final (x)).
```

Dacă introducem datele ca în figura avem rezultatul următor:

```
      relation(final(_), [ 0, 0, 0, 0, 0, 0, 0, 0, 0 ]),

      relation(g(_), [ 1, 0, 0, 1, 0, 0, 1, 1, 0 ]),

      relation(start(_), [ 0, 1, 0, 1, 0, 0, 1, 0, 1 ]),

      relation(f(_,_), [
                      1, 1, 0, 1, 0, 0, 0, 0, 0,
                      0, 0, 0, 0, 0, 0, 0, 0, 0,
                      0, 0, 0, 0, 0, 0, 0, 0, 0,
                      1, 0, 0, 1, 1, 0, 1, 0, 0,
                      0, 0, 0, 0, 0, 0, 0, 0, 0,
                      0, 0, 0, 0, 0, 0, 0, 0, 0,
                      0, 0, 0, 1, 0, 0, 1, 1, 0,
                      0, 0, 0, 0, 1, 0, 1, 1, 1,
                      0, 0, 0, 0, 0, 0, 0, 0, 0 ])
]).
```

Finalul și Startul sunt cum au fost introduse. G-ul este soluția optima a problemei (Figura 2): trebuie apăsate becurile 0, 3, 6 și 7. Matricea f arata toate schimbările care au fost efectuate. De exemplu, becul 3 a fost apăsat și se poate observa pe linia 3 din matrice f ca vecini sai (inclusiv el însuși) sunt: 0, 3, 4, și 6.

2.3 Îmbunătățire

Calcularea tuturor vecinilor unui bec devine din ce în ce mai complexa cu creșterea dimensiunilor matricei, de aceea o sa încercam evitarea hard-codării vecinilor. O sa avem nevoie de utilizarea "set(arithmetic)" și de definirea următoarelor variabile:

- n=3 (dimensiunea matricei)
- d=8 (domeniul maxim n*n-1)
- l1 = 2 ,l2 = 5 ,l3 = 8(ultimele valori de pe fiecare linie)

Utilizând valorile de pe primele rânduri putem defini predicatele : line1,line2,line3. Acestea conțin 1 dacă x este pe linia respectiva altfel 0. Partea aceasta a codului trebuie modificata pentru fiecare schimbarea a n-ului.

$$\forall x((x < l1 \vee x = l1) \iff line1(x))$$
$$\forall x(x > l1 \wedge (x < l2 \vee x = l2) \iff line2(x))$$
$$\forall x(x > l2 \wedge (x < l3 \vee x = l3) \iff line3(x))$$
$$...$$

Pentru calcularea vecinilor unui bec "x" procedam astfel : vecinul de sus se calculează ca fiind x-n (dacă x-n ¿= 0), vecinul de jos este x+n (dacă x+n ¡= d), vecinul din dreapta este x+1 dacă este pe aceeași linie ca x, iar vecinul din stânga este x-1 dacă este pe aceeași linie ca y:

```
all x all y (
 g(x) &
 (
    (y = x+1        -- vecinul din drepata
            & (y<d | y=d)    -- nu depaseste intervalul
            & ((line1(x) & line1(y)) | (line2(x) & line2(y)) | (line3(x) & line3(y)))
     )
     |(y = x-1      -- vecinul din stanga
            & (y>0 | y=0)
            & ((line1(x) & line1(y)) | (line2(x) & line2(y)) | (line3(x) & line3(y))))
    |(y = x+n       -- vecinul de sus
            & (y<d | y=d))
    |(y = x-n       -- vecinul de jos
            & (y>0 | y=0))
    |(y = x)        -- becul actual
   ) <-> f(x,y)
).
```

3 Testare

Testarea o vom realiza cu ajutorul interfeței date. Vom executa trei exemple de matrici generate de interfața la întâmplare, aceasta afișează datele de intrare și soluția generata de mace4.

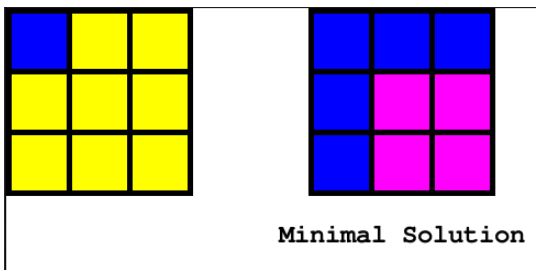

Figura 3: Test 1

```
Matricea initiala:
[[0 1 1]
 [1 1 1]
 [1 1 1]]
Out of Setup Mode
Rezultat:
{0: 0, 1: 0, 2: 0, 3: 0, 4: 1, 5: 1, 6: 0, 7: 1, 8: 1}
```

Figura 4: Test 1 output

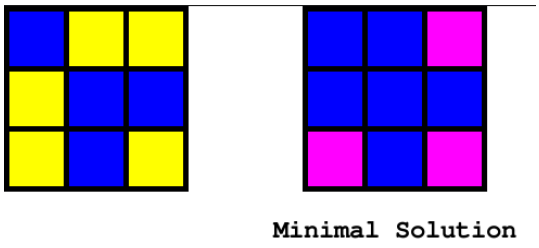

Figura 5: Test 2

```
Matricea initiala:
[[0 1 1]
 [1 0 0]
 [1 0 1]]
Out of Setup Mode
Rezultat:
{0: 0, 1: 0, 2: 1, 3: 0, 4: 0, 5: 0, 6: 1, 7: 0, 8: 1}
```

Figura 6: Test 2 output

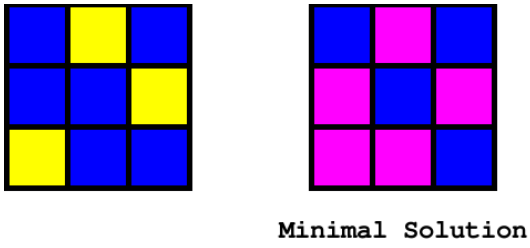

Figura 7: Test 3 output

```
Matricea initiala:
[[0 1 0]
 [0 0 1]
 [1 0 0]]
Out of Setup Mode
Rezultat:
{0: 0, 1: 1, 2: 0, 3: 1, 4: 0, 5: 1, 6: 1, 7: 1, 8: 0}
```

Figura 8: Test 3 output

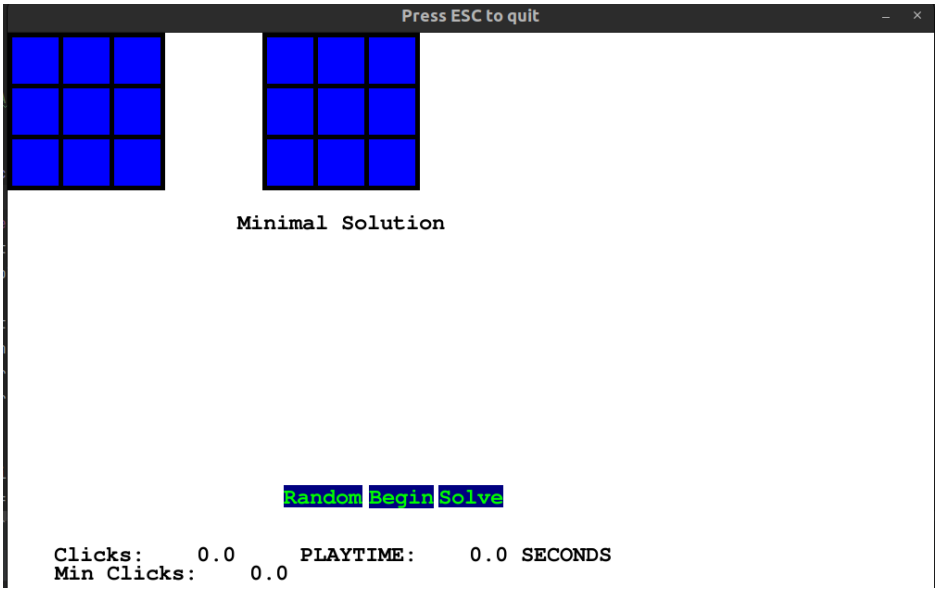

Figura 9: Pagina inițiala

4 Utilizarea aplicației

- Pentru a introduce o matrice de start se poate apasă pe becuri sau pe butonul "Random" (Figura 9).
- Pentru a începe jocul se apasă pe "Begin".
- Pentru a ase afișa soluția minima a jocului inițial se apasă butonul "Solve". Cele doua matrici vor conține: prima cu galben poziția inițiala, iar a doua cu roz butoanele care trebuie apăsate (Figura 10).
- Pentru verificarea soluției se pot apasă cu mouse-ul pătratele galben pentru a a ne întoarce la starea inițiala. Apoi se apasă "begin" și se pot apasă în orice ordine pătratele din soluție conform matricei 2.
- Pentru o alta introducere de date aplicația trebuie repornita, altfel rezultatul afișat nu va fi corect!

5 Bibliografie

- Informați despre Lights Out :https://www.jaapsch.net/puzzles/lomath.htm#linalg
- Interfața proiect: https://github.com/ProfDNash/Lights-Out

6 Codul în MACE4

```
assign(domain_size, 9).
set(arithmetic).

formulas(demodulators).
 n = 3.   % marime matrice nXn
 d = 8.   % donemiu
 l1 = 2.   % pentru a calcula linile:
 l2 = 5.
 l3 = 8.
end_of_list.

formulas(going_to_church).
```

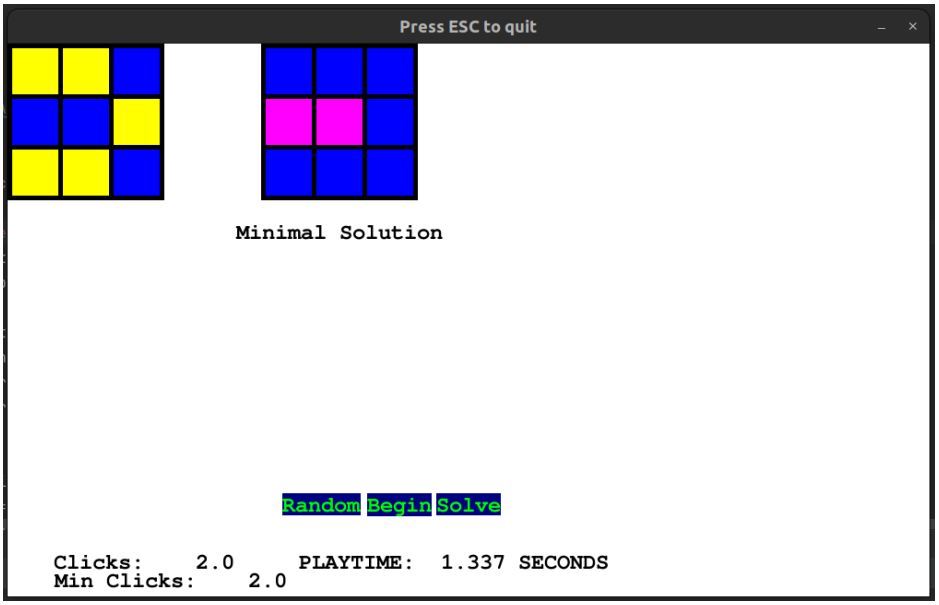

Figura 10: Pagina cu soluția

```
% predicatul start(x) este True daca becul x este aprins la inceputul rundei, altfel False:
  -start(0).
  start(1).
  -start(2).
  start(3).
  -start(4).
  -start(5).
  start(6).
  -start(7).
  start(8).

% predicatul final(x) este True daca becul x trbuie sa fie aprins la finalul rundei, altfel False (! noi v
 all x (-final(x)).


 -g(x) -> - f(x,y).

 (x<l1|x=l1) <-> line1(x).
 x>l1 & (x<l2|x=l2)<-> line2(x).
 x>l2 & (x<l3|x=l3)<-> line3(x).

 all x all y (
  g(x) &
  (
  (y = x+1 & (y<d | y=d) & ((line1(x) & line1(y)) | (line2(x) & line2(y)) | (line3(x) & line3(y))))
         |     (y = x+-1 & (y>0 | y=0) & ((line1(x) & line1(y)) | (line2(x) & line2(y)) | (line3(x) & line
          |     (y = x+n & (y<d | y=d))
          |     (y = x+-n & (y>0 | y=0))
          |     (y = x)
        ) <-> f(x,y)

).
```

```
all x(  -((((((((((start(x) <-> f(0,x)) <-> f(1,x)) <-> f(2,x))<-> f(3,x) ) <-> f(4,x))<-> f(5,x))<-> f(6,x

end_of_list.
```

Codul original:

```
assign(domain_size, 9).
set(arithmetic).

formulas(going_to_church).

% predicatul start(x) este True daca becul x este aprins la inceputul rundei, altfel False:
  -start(0).
  start(1).
  -start(2).
  start(3).
  -start(4).
  -start(5).
  start(6).
  -start(7).
  start(8).

% predicatul final(x) este True daca becul x trbuie sa fie aprins la finalul rundei, altfel False (! noi v

   all x (-final(x)).

  -g(x) -> - f(x,y).


  g(0) -> f(0,0) & f(0,1) & f(0,3) & (all y (y!=0 & y!=1 & y!=3 <-> -f(0,y) )).
  g(1) -> f(1,0) & f(1,1) & f(1,2) & f(1,4) & (all y (y!=0 & y!=1 &y!=2 &y!=4 <-> -f(1,y))).
  g(2) -> f(2,1) & f(2,2) & f(2,5) & (all y(y!=1 & y!=2 &y!=5 <-> -f(2,y))).
  g(3) -> f(3,0) & f(3,3) & f(3,4) & f(3,6) & (all y(y!=0 & y!=3 &y!=4 &y!=6 <-> -f(3,y))).
  g(4) -> f(4,1) & f(4,3) & f(4,4) & f(4,5) & f(4,7) & (all y(y!=1 & y!=3 &y!=4 &y!=5 &y!=7 <-> -f(4,y))).
  g(5) -> f(5,2) & f(5,4) & f(5,5) & f(5,8) & (all y(y!=2 & y!=4 &y!=5 &y!=8 <-> -f(5,y))).
  g(6) -> f(6,3) & f(6,6) & f(6,7) & (all y(y!=3 & y!=6 &y!=7 <-> -f(6,y))).
  g(7) -> f(7,4) & f(7,6) & f(7,7) & f(7,8) & (all y(y!=4 & y!=6 &y!=7 &y!=8 <-> -f(7,y))).
  g(8) -> f(8,5) & f(8,7) & f(8,8) & (all y(y!=5 & y!=7 &y!=8 <-> -f(8,y))).

all x (  -((((((((((start(x) <-> f(0,x)) <-> f(1,x)) <-> f(2,x))<-> f(3,x) ) <-> f(4,x))<-> f(5,x))<-> f(6,

end_of_list.
```