




Project INF432 2020
Light Up (Akari)

I/ Rules

A grid of any size (in general a square but it is not mandatory) is composed of :

 : free case  /  : Walls (With or without Number N going from 0 to 4)

- Main goal : Dispose light to enlighten all the free cases of the grid.
 - A light enlighten all his column and line until the light touch a wall or the border of the grid
 - A light cannot enlighten an other light
 - It is impossible to put a light on a wall or to put 2 light in a free case
 - A wall can have a number from 0 to 4. This number is indicating how many light must be placed adjacent to it's sides.
Ex: In there is N written on the wall there should be N light directly touching his side
 - A wall without numbering is not constraint by this kind of rules (rule n° 4)

II/Modeling

We will present how we think to model the game and the Conjunctive Normal Form (CNF) associated to each rules.

First, we have one variable by case denoted $l_{x,y}$: "There is a light in the case of coordinate (x, y)"


We one variable by case we will know in each case we could put a light or not.

The adjacent case are though represented by:

$l_{x,y-1}$ (top one); $l_{x-1,y}$ (left one); $l_{x+1,y}$ (right one); $l_{x,y+1}$ (bottom one)

To simplify formulas we will simply call them respectively 1,2,3 and 4.

Rule n°1, n°2 and Main goal:

	1	2	3	4	5	6
---	---	---	---	---	---	---

A line beginning by a wall and ending on a border

Imagine that is a line or transposed raw, to represent the fact that a light could enlighten another but has the grid must be fill by light there should be one but not more.

So the formula that I will give represent "there should be one and only one light":

$$(\neg 1V \neg 2) \wedge (\neg 1V \neg 3) \wedge (\neg 1V \neg 4) \wedge (\neg 1V \neg 5) \wedge (\neg 1V \neg 6) \wedge (\neg 2V \neg 3) \wedge (\neg 2V \neg 4) \wedge (\neg 2V \neg 5) \wedge (\neg 2V \neg 6) \\ \wedge (\neg 3V \neg 4) \wedge (\neg 3V \neg 5) \wedge (\neg 3V \neg 6) \wedge (\neg 4V \neg 5) \wedge (\neg 4V \neg 6) \wedge (\neg 5V \neg 6) \wedge (1V 2V 3V 4V 5V 6)$$

Modeling this example we distinguish two parts, the blue that prevent to have two light on this line and the red one that assure that there will be one light in the line.

If we have a line with more free cases non interrupt by a wall or the border of the grid, we denote each cases by 1, ..., n and denote the set composed by those cases/numbers C. To have the general formula (in Boolean algebra) we have to exclude two by two each pair and to have big clause with all the elements of C.

$\prod_{(i,j) \in C} (\bar{i} + \bar{j}) * \sum_{i \in C} i$ And we have our conjunctive normal form for the rule n°1, rule n°2 and our main goal. In addition, we see that if it is alone this only add a clause with the variable because there is no couple in C..

Walls' rule:

- The most obvious is the rule n°3 to represent, If there is a wall empty or not on the case $l_{x,y}$ we had the clause $\neg l_{x,y}$ to our final CNF. (or $l_{(x,y)}$)
- Now, we last to model the walls with number and using some exclusions formulas.

Small remember: The adjacent case are though represented by:

$l_{x,y-1}$ (top one); $l_{x-1,y}$ (left one); $l_{x+1,y}$ (right one); $l_{x,y+1}$ (bottom one)

To simplify formulas we will simply call them respectively 1,2,3 and 4.

For example to exclude all possibilities to have 3 true variables and 1 false variable among the set $A=\{1,2,3,4\}$.

We have to use this kind of formula: that is representing a conjunction of all subset of three elements neglected completed with the one missing at positive terms at symmetrically for 3 false and 1 true and look similar to have 2 true and 2 false excluded.

	1	
--	---	--

2	N	3
	4	

In this context we have 5 cases (but it's like we have only 3) and their formulas.

- For $N=0$, $\neg 1 \wedge \neg 2 \wedge \neg 3 \wedge \neg 4$ which is a CNF with each clause containing one variable.
Symmetrically, for $N=4$ $1 \wedge 2 \wedge 3 \wedge 4$

- For $N=1$,

$$(1 \vee 2 \vee 3 \vee 4) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee \neg 4) \wedge \\ (1 \vee \neg 2 \vee \neg 3 \vee \neg 4) \wedge (\neg 1 \vee 2 \vee \neg 3 \vee \neg 4) \wedge (\neg 1 \vee \neg 2 \vee 3 \vee \neg 4) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee 4) \\ (\neg 1 \vee \neg 2 \vee 3 \vee 4) \wedge (\neg 1 \vee 2 \vee \neg 3 \vee 4) \wedge (\neg 1 \vee 2 \vee 3 \vee \neg 4) \wedge (1 \vee \neg 2 \vee \neg 3 \vee 4) \wedge (1 \vee \neg 2 \vee 3 \vee \neg 4) \wedge (1 \vee 2 \vee \neg 3 \vee \neg 4)$$

On the one hand the black assure at least one is true. On the other hand, the blue parts assure that there is not 4 light adjacent to the case. Green is killing the model with three light and red the one with 2 light.

- For $N=3$, only the green part is inverted and this give:

$$(1 \vee 2 \vee 3 \vee 4) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee \neg 4) \wedge \\ (\neg 1 \vee 2 \vee 3 \vee 4) \wedge (1 \vee \neg 2 \vee 3 \vee 4) \wedge (1 \vee 2 \vee \neg 3 \vee 4) \wedge (1 \vee 2 \vee 3 \vee \neg 4) \\ (\neg 1 \vee \neg 2 \vee 3 \vee 4) \wedge (\neg 1 \vee 2 \vee \neg 3 \vee 4) \wedge (\neg 1 \vee 2 \vee 3 \vee \neg 4) \wedge (1 \vee \neg 2 \vee \neg 3 \vee 4) \wedge (1 \vee \neg 2 \vee 3 \vee \neg 4) \wedge (1 \vee 2 \vee \neg 3 \vee \neg 4)$$

- Finally, for $N=2$, it re-use the same patterns:

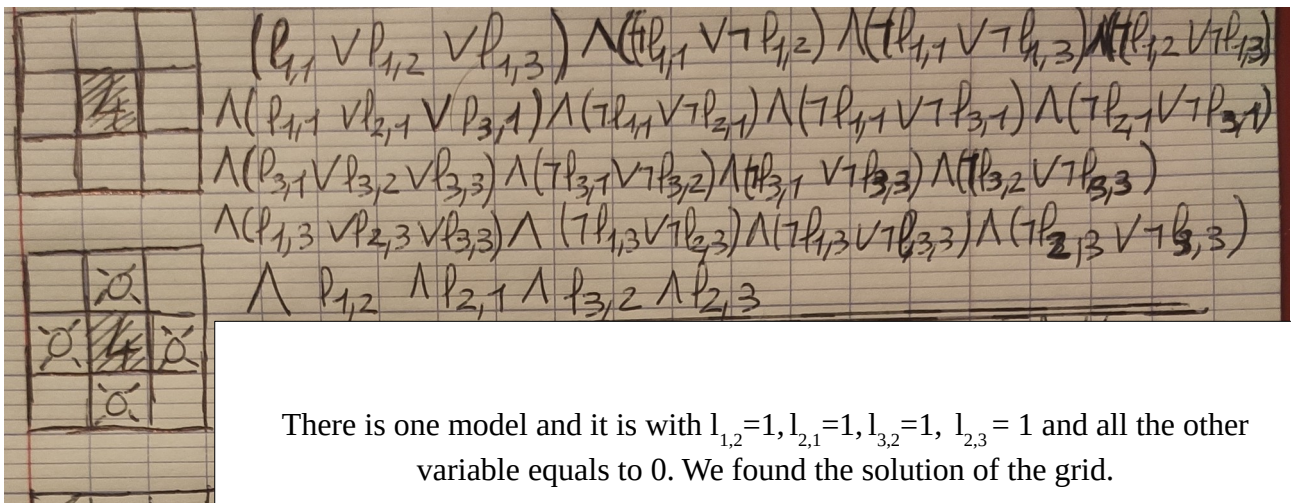
$$(1 \vee 2 \vee 3 \vee 4) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee \neg 4) \wedge \\ (\neg 1 \vee 2 \vee 3 \vee 4) \wedge (1 \vee \neg 2 \vee 3 \vee 4) \wedge (1 \vee 2 \vee \neg 3 \vee 4) \wedge (1 \vee 2 \vee 3 \vee \neg 4) \\ (1 \vee \neg 2 \vee \neg 3 \vee \neg 4) \wedge (\neg 1 \vee 2 \vee \neg 3 \vee \neg 4) \wedge (\neg 1 \vee \neg 2 \vee 3 \vee \neg 4) \wedge (\neg 1 \vee \neg 2 \vee \neg 3 \vee 4)$$

We have to be careful during programming to not add a case outside the grid. We just have to ignore it and to exclude with the same kind of technique shown before () but in environment where the size of A is 2 or 3 (there can be only 2 out for a size of grid superior to 2).

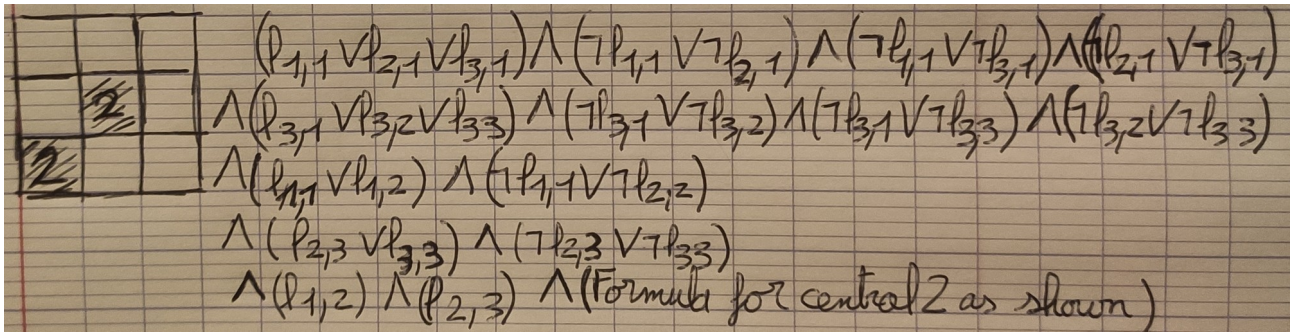
III/In context

We experiment our model on two example, one easy and the other a bit more complicated.

Using, at the beginning, the model for line and row and at the model for walls.



There is one model and it is with $l_{1,2}=1, l_{2,1}=1, l_{3,2}=1, l_{2,3}=1$ and all the other variable equals to 0. We found the solution of the grid.



The only model is again the only solution of the grid with $l_{1,2} = 1$, $l_{2,3} = 1$, $l_{3,1} = 1$ and the rest equal to 0.

We have to see if it's working for more complicated.

It is remarkable the adaptation of the formula for a line of two case at third and fourth line, as well as the adaptation for the wall with a 2 at last line. As the 2-combination of set of size two is one we only have one "positive" variable in our clause.

IV/ What do we have already achieve ? (in addition to modeling)

Now we have a generator of grid coded in C that is working nearly good the two problem is that if we asked a high density of a wall it will give an unsolvable grid. In fact, the proportion of solvable grid depend on the density asked.

Moreover, sometimes the number on the wall is not the good one but we have solved the problem about wall with number. In our generator, it will never have obligation to put light beside a wall that is enlighten an other light of a numbered wall.

Finally, the grid is not completely solved at the end of the generator only the part with wall it is and just have to remove the light from the grid to make playable.

Here is some grid at different size generated:

#:Empty wall 0,1,2,3,4:Numbered walls _:Free case .: case that should be empty

Density: 0,2

```
#.____
_.0.____
_____.
_#_#_#
_____.
_.0.____
_._#01
```

Density: 0,3

```
_____.#.#1
_____.#1.0
_._0.0#_
_____.
##.0.##
10_.##_
.##_#
```

Density: 0,4

```
_._030_
.0.0.0.
#._.0.
020##.
#.10#
##.0.
_#_#_
```

Density: 0,3

```
#._.0#_____.#_
0.0.2._____.0.0.0.#
2...0#0.0...01.
0.#1._____.0.0.0.
.0.0#_____.
#._.0._____.0.
10._____.#_____.#
_.0.#_____.0._____.#
_.#_____.0._____.
.0._____.0._____.#
...0._____.0.0.0.
20_.#_____.#_____.#
0_____.#_____.0._____.
#_.0.#0.0.10.
_.1.#10.#_____.020_#
_0_.#_____.#_____.02
#_.0._____.0.0.0.
#.0.1.#_____.#_____.0.
_.0#0.0.0.0._____.#
_____030#.#_____.##01
```