**Pré-rapport Projet Logique INF402**

*Sujet : Paint It Black II (nonogram)*

*Règles* : chaque ligne / colonne contient le nombre de case remplies consécutives indiqué en face de la ligne ou de la colonne, dans l’ordre, deux entiers séparés sont donc deux blocs nécessairement séparés (dernière ligne, fig. 1)

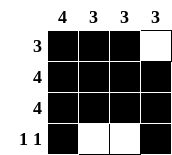
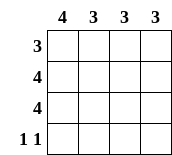
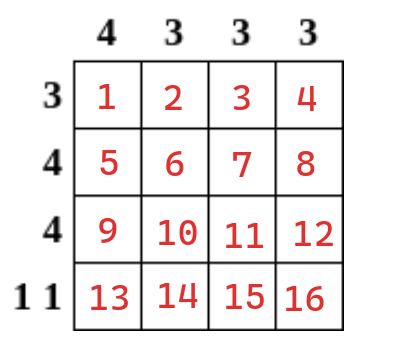




fig. 1

**Modélisation en Forme Normale Conjonctive :**

En prenant comme exemple la figure 1 on a :

nombre\_de\_variables = nombre\_lignes \* nombre\_colonnes (donc ici 16 = 4 \* 4)

Chaque littéral (entier ici) est soit positif ( ) si la case est possiblement remplie, soit négatif

( ) si la case est possiblement vide

Effectivement, chaque ligne / colonne a plusieurs possibilités de placements, prenons la dernière ligne de la fig. 1, avec les noms de variables définis plus haut (directement entiers pour le sat-solver, aussi plus simple pour vérifier la polarité):



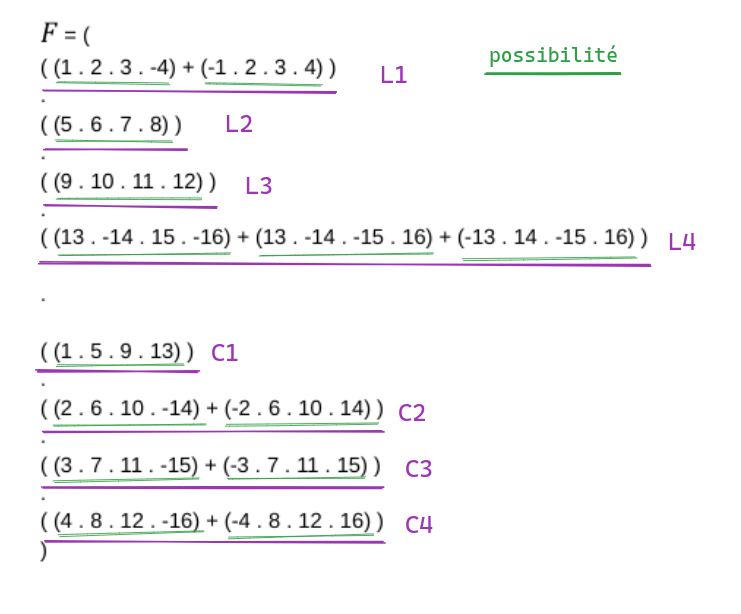
Une possibilité de la ligne :

Toutes les possibilités de la ligne :

On fait la même chose pour chaque ligne et chaque colonne, chaque formule de ligne / colonne est connectée au reste par un produit pour modéliser les contraintes qui s’appliquent en même temps pour toutes les lignes / colonnes, avec des sous - formules de différentes possibilités dans celles de lignes / colonnes

(une possibilité est un produit de littéraux (contraindre un seul arrangement de toutes les cases d’une ligne / colonne = une possibilité))

On obtient donc la formule modélisant le jeu, toujours pour la fig. 1 :



= (

( (1 . 2 . 3 . -4) + (-1 . 2 . 3 . 4) )

.

( (5 . 6 . 7 . 8) )

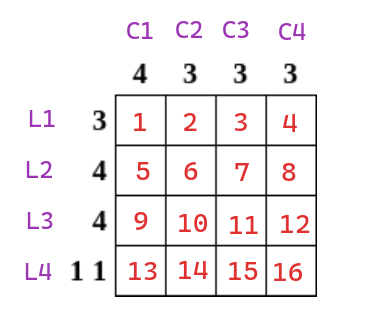
.

( (9 . 10 . 11 . 12) )

.

( (13 . -14 . 15 . -16) + (13 . -14 . -15 . 16) + (-13 . 14 . -15 . 16) )

.



( (1 . 5 . 9 . 13) )

.

( (2 . 6 . 10 . -14) + (-2 . 6 . 10 . 14) )

.

( (3 . 7 . 11 . -15) + (-3 . 7 . 11 . 15) )

.

( (4 . 8 . 12 . -16) + (-4 . 8 . 12 . 16) )

)

L’étape suivante consiste à distribuer l’intérieur des lignes et de colonnes pour obtenir des sous formules normales conjonctives (on peut enlever les parenthèses extérieures au final)

En distribuant la première ligne (L1), on obtient :

( (1 + -1) . (1 + 2) . (1 + 3) . (1 + 4) . (2 + -1) . (2 + 2) . (2 + 3) . (2 + 4) . (3 + -1) . (3 + 2) . (3 + 3) . (3 + 4) . (-4 + -1) . (-4 + 2) . (-4 + 3) . (-4 + 4) )

(donc une fnc)

En distribuant on obtient donc (raccourci car 140 clauses) (on pourra par la suite simplifier des clauses en retirant une même variable de même polarité qui apparaît plusieurs fois dans la même clause) :

( (1 + -1) . (1 + 2) . (1 + 3) . (1 + 4) . (2 + -1) . (2 + 2) . (2 + 3) . (2 + 4) . (3 + -1) . (3 + 2) . (3 + 3) . (3 + 4) . (-4 + -1) . (-4 + 2) . (-4 + 3) . (-4 + 4) . (5) . (6) . (7) . (8)

(9) . (10) . (11) . (12)

(13 + 13 + -13) . (13 + 13 + 14) . (13 + 13 + -15) . (13 + 13 + 16) . (13 + -14 + -13) . (13 + -14 + 14) . (13 + -14 + -15) . (13 + -14 + 16) . (13 + -15 + -13) . (13 + -15 + 14) . (13 + -15 + -15) . (13 + -15 + 16) . (13 + 16 + -13) . (13 + 16 + 14) . (13 + 16 + -15) . (13 + 16 + 16) . (-14 + 13 + -13) . (-14 + 13 + 14) . (-14 + 13 + -15) . (-14 + 13 + 16) . (-14 + -14 + -13) . (-14 + -14 + 14) . (-14 + -14 + -15) . (-14 + -14 + 16) . (-14 + -15 + -13) . (-14 + -15 + 14) . (-14 + -15 + -15) . (-14 + -15 + 16) . (-14 + 16 + -13) . (-14 + 16 + 14) . (-14 + 16 + -15) . (-14 + 16 + 16) . (15 + 13 + -13) . (15 + 13 + 14) . (15 + 13 + -15) . (15 + 13 + 16) . (15 + -14 + -13) . (15 + -14 + 14) . (15 + -14 + -15) . (15 + -14 + 16) . (15 + -15 + -13) . (15 + -15 + 14) . (15 + -15 + -15) . (15 + -15 + 16) . (15 + 16 + -13) . (15 + 16 + 14) . (15 + 16 + -15) . (15 + 16 + 16) . (-16 + 13 + -13) . (-16 + 13 + 14) . (-16 + 13 + -15) . (-16 + 13 + 16) . (-16 + -14 + -13) . (-16 + -14 + 14) . (-16 + -14 + -15) . (-16 + -14 + 16) . (-16 + -15 + -13) . (-16 + -15 + 14) . (-16 + -15 + -15) . (-16 + -15 + 16) . (-16 + 16 + -13) . (-16 + 16 + 14) . (-16 + 16 + -15) . (-16 + 16 + 16) .

*… raccourci …*

. (-16 + 16 + 16) . (1) . (5) . (9) . (13) . (2 + -2) . (2 + 6) . (2 + 10) . (2 + 14) . (6 + -2) . (6 + 6) . (6 + 10) . (6 + 14) . (10 + -2) . (10 + 6) . (10 + 10) . (10 + 14) . (-14 + -2) . (-14 + 6) . (-14 + 10) . (-14 + 14) . (3 + -3) . (3 + 7) . (3 + 11) . (3 + 15) . (7 + -3) . (7 + 7) . (7 + 11) . (7 + 15) . (11 + -3) . (11 + 7) . (11 + 11) . (11 + 15) . (-15 + -3) . (-15 + 7) . (-15 + 11) . (-15 + 15) . (4 + -4) . (4 + 8) . (4 + 12) . (4 + 16) . (8 + -4) . (8 + 8) . (8 + 12) . (8 + 16) . (12 + -4) . (12 + 8) . (12 + 12) . (12 + 16) . (-16 + -4) . (-16 + 8) . (-16 + 12) . (-16 + 16) )

**Vérification et programmation :**

On code en python et on récupère une instance du jeu (une configuration du jeu) dans un fichier JSON (car données structurées), librairie ‘json’ pour parser le fichier

On peut vérifier certains calculs avec [bddc](https://www-verimag.imag.fr/~raymond/home/tools/bddc/) (calculatrice logique)