MSE | MASTER OF SCIENCE IN ENGINEERING

Générateur de grilles de Sudoku en Promela

Mini-projet de David Fischer

5	6	4	3	1	7	8	2	9
8	9	7	6	4	2	5	3	1
3	1	2	8	9	5	4	7	6
6	4	5	9	7	3	1	8	2
2	8	1	5	6	4	7	9	3
7	3	9	1	2	8	6	4	5
4	5	6	2	8	9	3	1	7
9	7	3	4	5	1	2	6	8
1	2	8	7	3	6	9	5	4

24. janvier 2011
Module VERIF - M. Luigi Zaffalon

Table des matières

1. Cahier des charges

> Concevoir un algorithme en PROMELA [1] de génération de grilles complètes de sudoku ;
> Exécuter la solution proposée dans l'environnement de validation formelle SPIN [2] ;
> S'inspirer de l'algorithme non déterministe de résolution du problème du placement des 8 reines conçu par Robert W. Floyd en 1967 [3] ;

2. Introduction

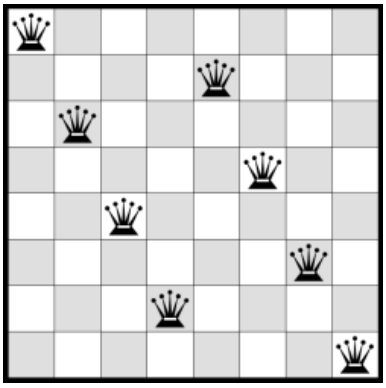
Le thème de ce mini-projet est la génération d'une grille complète de sudoku de taille quelconque, répondant aux règles du jeu. Pour rappel, voici la définition tirée de Wikipedia [4] :

> *« Le but du jeu est de remplir la grille avec une série de chiffres (ou de lettres ou de symboles) tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille. La plupart du temps, les symboles sont des chiffres allant de 1 à 9, les sous-grilles étant alors des carrés de 3 × 3. Quelques symboles sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet. »*

5	6	4	3	1	7	8	2	9
8	9	7	6	4	2	5	3	1
3	1	2	8	9	5	4	7	6
6	4	5	9	7	3	1	8	2
2	8	1	5	6	4	7	9	3
7	3	9	1	2	8	6	4	5
4	5	6	2	8	9	3	1	7
9	7	3	4	5	1	2	6	8
1	2	8	7	3	6	9	5	4

Et pour ce faire nous allons nous inspirer de l'algorithme de résolution du jeu des 8 reines, [5] :

> *« Le but du problème des huit reines est de placer huit reines d'un jeu d'échecs sur un échiquier de 8 × 8 cases sans que les reines ne puissent se menacer mutuellement, conformément aux règles du jeu d'échecs (la couleur des pièces étant ignorée). Par conséquent, deux reines ne devraient jamais partager la même rangée, colonne, ou diagonale. »*



3. Étude préliminaire

L'outil SPIN met à disposition plusieurs modes de fonctionnement, notamment un mode dit de Vérification. Dans ce mode, SPIN passe en revue l'ensemble des états que peut prendre l'algorithme et lève une exception si une des propriétés ou contrainte de la spécification serait violée.

L'algorithme des 8 reines de Robert W. Floyd se sert de cette propriété en proposant un algorithme non déterministe (dont le positionnement des reines dépend d'un aléa) basé sur le concept suivant :

méthode HuitReines :
boucler de la première colonne de l'échiquier à la dernière
> pour chaque colonne, proposer aléatoirement, la ligne où placer la reine
> « bloquer » si la ligne proposée contient déjà une reine
> « bloquer » si les diagonales de la case proposée contient déjà une reine
> placer la reine dans la case de position < colonne ; ligne >
> mettre à jour les variables nécessaires à l'algorithme
> « lever une exception » si toutes les (8) reines sont posées -> Solution trouvée !

Lors de l'exécution de l'algorithme suivant, l'outil SPIN passe en revue l'ensemble des exécutions possibles et s'arrêtera au moment où une exception serait levée, signe que les 8 reines ont été correctement placées !

L'utilisation de l'aléa, lors de la sélection de la ligne où placer la reine, permet ainsi de passer en revue l'ensemble des positionnements de reines possibles tandis que les expressions booléennes (bloquantes, le mot bloquer du pseudo-code) empêchent de proposer une solution erronée !

Il existe une analogie forte entre la résolution du jeu des 8 reines et celle de la génération d'une grille de sudoku ! En effet, ces deux jeux comportent des règles de placement pour les éléments du jeu.

Il est par conséquent possible de détourner l'implémentation PROMELA des 8 reines pour en faire un outils de génération de grille sudoku !

4. Solution proposée

Remarque : sauf indication contraire, les valeurs numériques données sont basées sur un sudoku standard, à savoir une grille 9x9 comportant 9x les nombres de 1 à 9.

L'idée maîtresse de mon algorithme tient en quelques lignes :

appel initial de la méthode Sudoku avec le premier nombre (1)

méthode Sudoku :
boucler de la première colonne de la grille du sudoku à la dernière
> pour chaque colonne, proposer aléatoirement, la ligne où poser le nombre actuel
> « bloquer » si la case proposée n'est pas vide
> « bloquer » si la ligne proposée contient déjà le nombre actuel
> « bloquer » si le bloc[1] lié à la case proposée contient déjà le nombre actuel
> placer le nombre actuel dans la case de position < colonne ; ligne >
incrémenter le nombre actuel
« lever une exception » si le nombre actuel est hors bornes (>9) -> Solution trouvée !
appel de la méthode Sudoku (récursivité)

Le placement des nombres dans la grille se faisant colonne par colonne, dans une case adéquate et lorsque toutes les occurrences du nombre actuel (9x le nombre 1, 9x le nombre 2, ..., 9x le nombre 9) sont posées, on fait appel récursivement à cette méthode pour placer le nombre « suivant » !

On peut se représenter les 9 occurrences d'un des nombres du Sudoku comme 9 reines à placer sur un échiquier (une grille) de 9x9 cases. Il est évident, qu'il faut respecter les règles du Sudoku et non celles du jeu des reines !

Et lors de chaque appel récursif de l'algorithme, il faut également prendre en compte le fait que certaines cases de la grille sont occupées par un nombre précédemment placé, des cases que nous devons éviter de proposer pour le placement d'un autre nombre !

1 Un bloc est, pour une grille de sudoku de 9x9, un carré de 3x3 cases et la grille en comporte 9 !

5. Réalisation

Voici un historique de mes différentes implémentations et analyses lors de ce Mini-projet :

- retranscription de l'algorithme des 8 reines en PROMELA
> Eight-Queens/eight-queens.pml

- analyse de « l'échiquier » résultant de l'exécution du code avec l'outil SPIN
> Résultats/échiquier-résultat.ods

- réflexion devant la feuille blanche, en vue de poser une solution analogue pour le Sudoku

- implémentation de l'algorithme né de ma réflexion, en PROMELA pour des grilles 9x9
> Sudoku/sudoku-grid-generator-9x9.pml

- lecture difficile des « grilles » résultants de l'exécution du code avec SPIN

- conception d'un script en bash pour la relecture facilitée des résultats par un être humain !
> sudokuViewer.sh

- implémentation PROMELA d'une version générique du générateur de grilles de Sudoku NxN
> Sudoku/sudoku-grid-generator-NxN.pml

- analyse des « grilles » résultant de l'exécution du code NxN avec l'outil SPIN
> Résultats/grid2x2_1_view.txt

- analyse des « grilles » résultant de l'exécution du code 9x9 avec l'outil SPIN
> Résultats/grid9x9_1_view.txt (du trail 1)
> Résultats/grid9x9_200_view.txt (du trail 200)
> Résultats/grid9x9_6000_view.txt (du trail 6000)

- implémentation PROMELA d'une version spécialisée pour une grille de 16x16
> Résultats/grid16x16_1_view.txt

6. Résultats

Voici quelques unes des grilles que j'ai cité au chapitre précédent ...

6.1 En 2x2 avec l'implémentation générique NxN

```
| 4 3 | 2 1 |
| 2 1 | 4 3 |

| 3 4 | 1 2 |
| 1 2 | 3 4 |
```

Spin est arrivé à en produire 288 différentes et ceci sans saturer la mémoire disponible !

6.2 En 9x9 avec la version spécialisé 9x9

```
| 1 2 3 | 4 5 6 | 7 8 9 |
| 7 8 9 | 1 2 3 | 4 5 6 |
| 4 5 6 | 7 8 9 | 1 2 3 |

| 2 1 4 | 3 6 5 | 9 7 8 |
| 9 7 8 | 2 1 4 | 3 6 5 |
| 3 6 5 | 9 7 8 | 2 1 4 |

| 5 3 1 | 6 4 2 | 8 9 7 |
| 8 9 7 | 5 3 1 | 6 4 2 |
| 6 4 2 | 8 9 7 | 5 3 1 |
```

Spin est arrivé à en produire 7'432 / 14'023 différentes en saturant 1 /2 Go de mémoire vive !

6.3 Et en 16x16 avec la version spécialisée 16x16

```
|  1  2  3  4 |  5  6  7  8 |  9 10 11 12 | 13 14 15 16 |
|  5  6  7  8 |  1  2  3  4 | 13 14 15 16 |  9 10 11 12 |
|  9 10 11 12 | 13 14 15 16 |  1  2  3  4 |  5  6  7  8 |
| 13 14 15 16 |  9 10 11 12 |  5  6  7  8 |  1  2  3  4 |

|  2  1  4  3 |  6  5  8  7 | 10  9 12 11 | 14 13 16 15 |
|  6  5  8  7 |  2  1  4  3 | 14 13 16 15 | 10  9 12 11 |
| 10  9 12 11 | 14 13 16 15 |  2  1  4  3 |  6  5  8  7 |
| 14 13 16 15 | 10  9 12 11 |  6  5  8  7 |  2  1  4  3 |

|  3  4  1  2 |  7  8  5  6 | 11 12  9 10 | 15 16 13 14 |
|  7  8  5  6 |  3  4  1  2 | 15 16 13 14 | 11 12  9 10 |
| 11 12  9 10 | 15 16 13 14 |  3  4  1  2 |  7  8  5  6 |
| 15 16 13 14 | 11 12  9 10 |  7  8  5  6 |  3  4  1  2 |

|  4  3  2  1 |  8  7  6  5 | 12 11 10  9 | 16 15 14 13 |
|  8  7  6  5 |  4  3  2  1 | 16 15 14 13 | 12 11 10  9 |
| 12 11 10  9 | 16 15 14 13 |  4  3  2  1 |  8  7  6  5 |
| 16 15 14 13 | 12 11 10  9 |  8  7  6  5 |  4  3  2  1 |
```

Spin est arrivé à en produire 2'993 différentes en saturant 2 Go de mémoire vive !

Il est pertinent d'affirmer que l'explosion de la taille et du nombre d'états nous empêche d'énumérer toutes les grilles possibles lorsque les dimensions de cette dernière atteint un certain « seuil ».

Cette limite dépend de plusieurs facteurs, notamment de la manière d'implémenter en PROMELA le générateur de grilles de Sudoku.

Prenons par exemple la génération du numéro de ligne de manière non déterministe, l'option la moins « générique » mais la plus performante selon mes analyses (elles-mêmes bornées au temps que je pouvais consacrer ...) semble être celle de la boucle do suivit de N gardes à true de ce gabarit :

```
do
:: true -> rowNb = 1;
:: true -> rowNb = 2;
:: true -> rowNb = 3;
.
.
.
:: true -> rowNb = N;
od ;
```

Tandis que le code générique est constitué d'une boucle réellement parcourue N fois !

```
rowNb     = 0;
randomCpt = 0;
do
:: randomCpt < N-1 -> randomCpt++; rowNb = rowNb+1;
:: randomCpt < N-1 -> randomCpt++;
:: else -> break;
od;
assert (rowNb >= 0 && rowNb < N);
```

*Avec l'ajout d'une instruction **atomic** { .. boucle ... } il m'a été possible de réduire le nombre d'états générés par la version générique, cependant cela n'a pas réellement réduit le temps d'exécution du code. C'est pourquoi j'ai développé 2 versions spécialisés, une pour le classique 9x9 et une autre pour une version plus impressionnante la 16x16 !*

Un autre élément entre en paramètre lors de l'énumération des états possibles : les options de l'outil SPIN. J'ai ainsi joué avec la quantité de mémoire disponible pour me convaincre que le nombre de grilles possibles en 9x9 et 16x16 n'est en définitive pas borné à ce qui est trouvé par l'exécution !

Je tiens à remarquer que même si les résultats présentés ci-dessus semblent valides, ce n'est, en définitive, pas prouvé de manière formelle !

En effet, nous avons détourné l'outil de vérification formelle pour atteindre notre objectif et celui-ci fut exprimé en terme d'expressions booléennes (bloquantes). Il suffit d'avoir mal traduit les règles du Sudoku en ces fameuses expressions booléennes (bloquantes) pour obtenir des grilles invalides selon les règles du jeu !

7. Guide de l'utilisateur

7.1 Comment puis-je tester l'implémentation des 8 reines ?

Démarrer l'environnement de vérification formelle iSpin, puis ouvrir le fichier eight-queens.pml, passer à l'onglet Verification et cliquer sur le bouton Run.

***Important** : Ne pas oublier de décocher l'option Invalid endstates (deadlocks), en effet, nous nous servons du principe d'état final invalide pour bloquer les solutions invalides !*

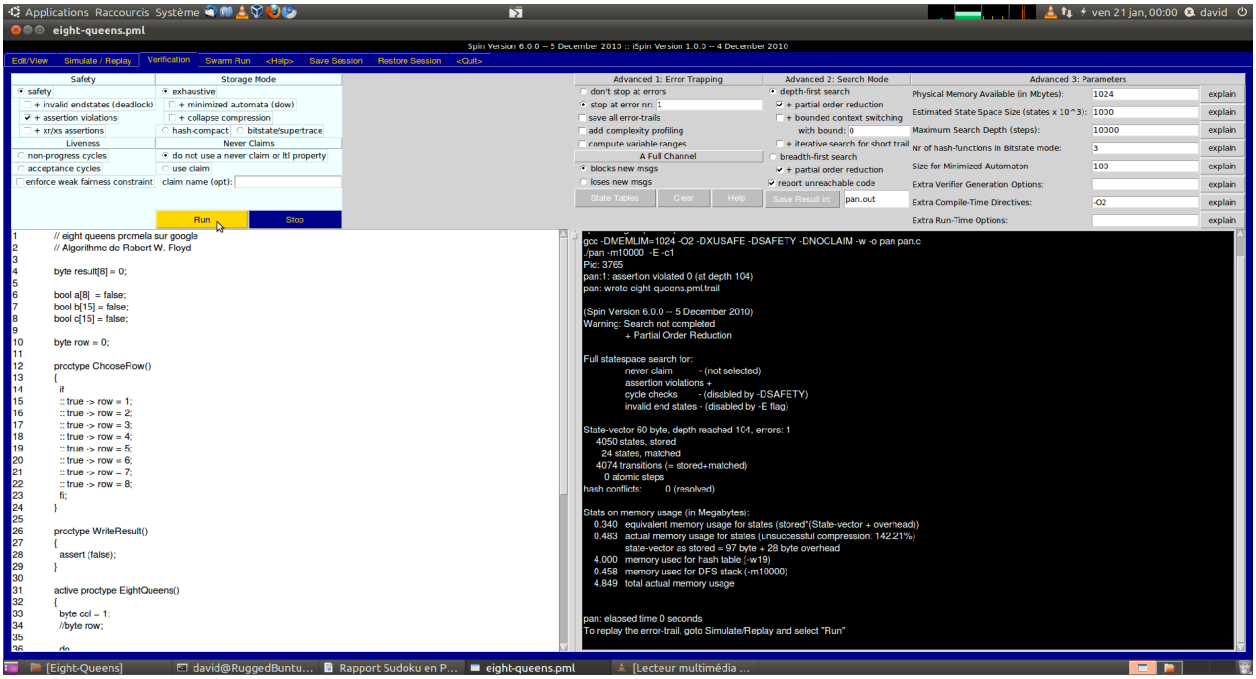
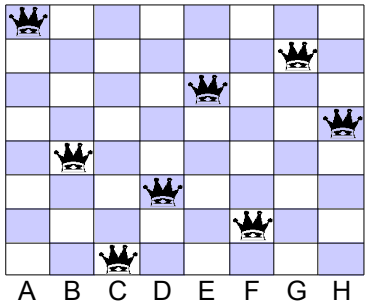


Illustration 1: Et une solution de trouvé !

Il suffit par la suite de passer à l'onglet Simulate/Replay et cliquer sur (Re)Run pour rejouer l'ensemble des transitions menant à l'état final invalide, état comportant l'ensemble des reines !

Illustration 2: Une solution



La solution se trouvant dans les valeurs du tableau result.
Par exemple,

result[0] = 1
result[1] = 5
result[2] = 8
result[3] = 6
result[4] = 3
result[5] = 7
result[6] = 2
result[7] = 4

Se décode ainsi :
Dans la colonne 1 (result[0]) la reine est positionnée en A (= 1) et ainsi de suite.

7.2 Comment puis-je tester l'implémentation du Sudoku ?

Démarrer l'environnement de vérification formelle iSpin, puis ouvrir le fichier sudoku-grid-generator-9x9, passer à l'onglet Verification et cliquer sur le bouton Run.

Important : Ne pas oublier de décocher l'option Invalid endstates (deadlocks), en effet, nous nous servons du principe d'état final invalide pour bloquer les solutions invalides !

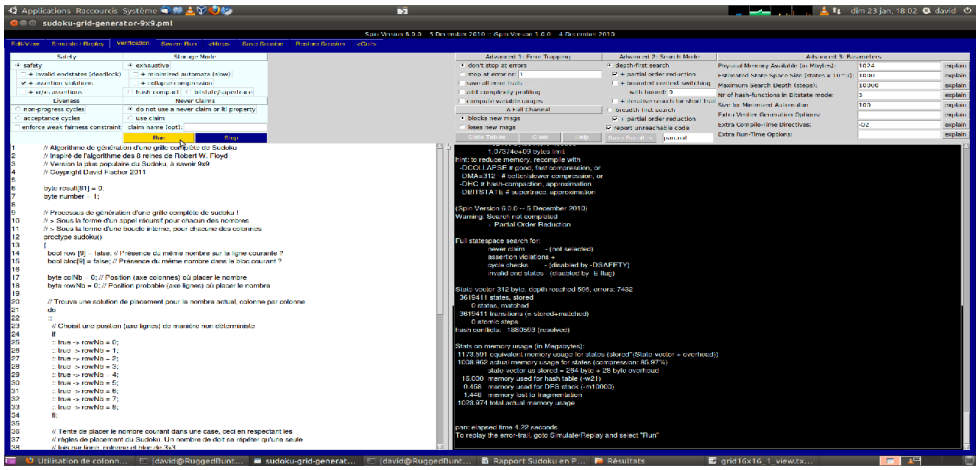


Illustration 3: Et 7432 solutions de trouvé !

Il suffit par la suite de passer à l'onglet Simulate/Replay et cliquer sur (Re)Run pour rejouer l'ensemble des transitions menant à l'état final invalide, état comportant l'ensemble des reines !

La solution se trouvant dans les valeurs du tableau result qu'il faut sélectionner et copier dans un fichier texte.

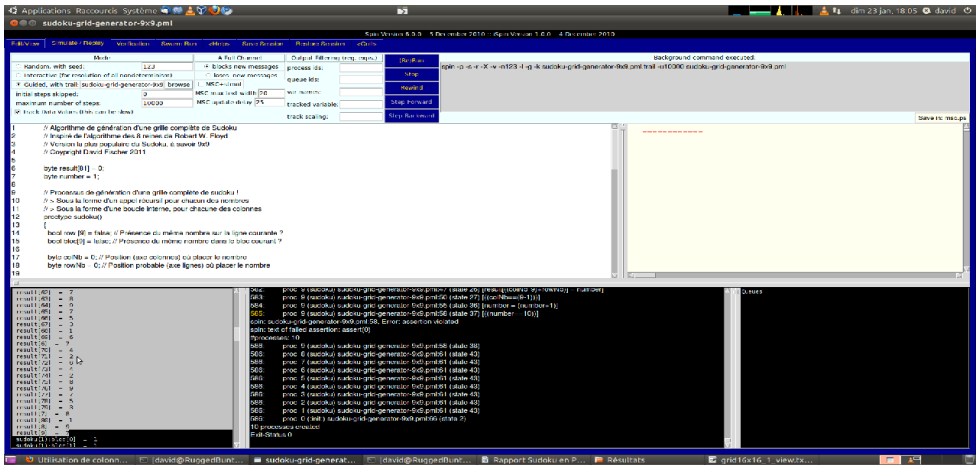


Illustration 4: Sélection des lignes result[] = ...

Puis, ouvrir un terminal pour exécuter le script sudokuViewer.sh avec le nom du fichier en paramètre :

```
$ ./sudokuViewer.sh in.txt > out.txt
```

Finalement, il suffit d'ouvrir le résultat du traitement dans un éditeur de texte pour observer la grille générée par l'exécution du générateur de grilles de sudoku !

8. Conclusion

Ce mini-projet m'a permit de mettre en œuvre de façon ludique les compétences que j'ai acquises en participant au cours dé vérification formelle [6] donné par Monsieur Luigi Zaffalon, professeur HES à l'hepia, ex EIG.

Il est intéressant de noter que l'implémentation PROMELA des deux algorithmes de résolution sont tous deux remarquablement concis et ceci grâce au comportement de l'outil SPIN !

La suite logique de ce projet pourrait-être l'étude et le développement d'un algorithme de vidage de la grille pour ainsi générer une énigme qu'un joueur de sudoku prendrait du plaisir à résoudre !

Je tiens à remercier mes collègues pour l'ambiance de travail sérieuse mais décontractée et mon professeur de vérification formelle, Monsieur L. Zaffalon, pour son effort et sa compétence à nous transmettre ce désir de formaliser un système pour ensuite pouvoir le valider !

En effet, cet aspect est à mon avis, un aspect trop facilement négligé par ceux qui œuvrent dans « l'informatique », au sens « développement de système d'information » du terme !

9. Références

[1] PROMELA selon Wikipedia : http://fr.wikipedia.org/wiki/PROMELA

[2] Site officiel de SPIN : http://spinroot.com/spin/whatispin.html

[3] Eight Queens Promela Algorithm : http://books.google.ch/books?id=eVTN8UanIGcC&lpg=PA168&ots=9CQonDmH3-&dq=eight%20queens%20promela&hl=fr&pg=PA168#v=onepage&q=eight%20queens%20promela&f=false

[4] Sudoku selon Wikipedia : http://fr.wikipedia.org/wiki/Sudoku

[5] 8 reines selon Wikipedia : http://fr.wikipedia.org/wiki/Probl%C3%A8me_des_huit_dames

[6] Site du cours VERIF du Master TIC HES-SO : http://eig.unige.ch/~iii/MSE_VERIF.htm