Mohamed Bilal NORAT
Léo PRZEWLOCKI

utc
Université de Technologie
Compiègne

- IA02 [P2018] -
Projet Sudoku

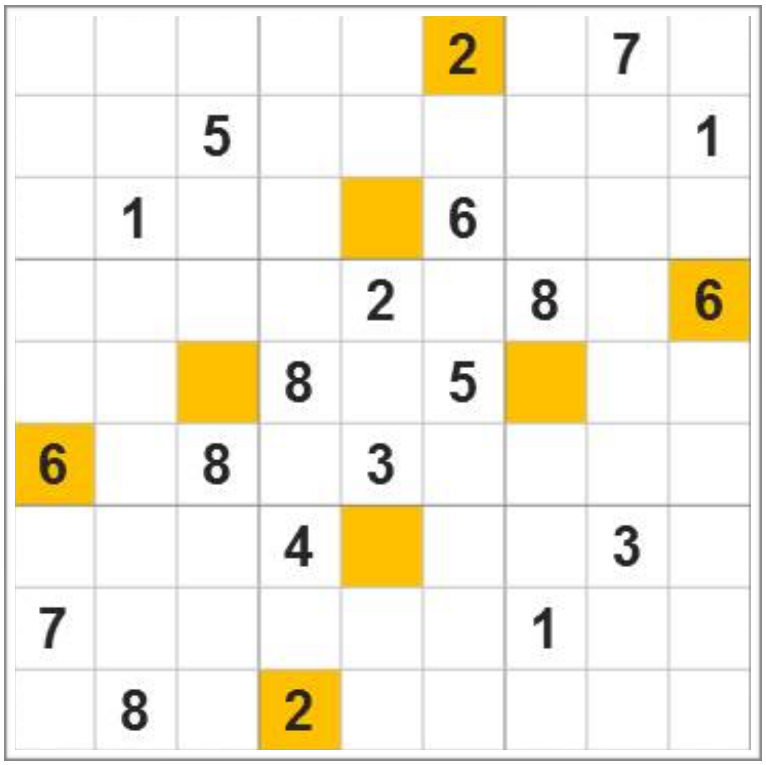

Table des matières

Objectifs du projet 2
Choix d'implémentation 2
Prédicats 3
Difficultés rencontrées 4
Notions apprises à travers le projet 4
Améliorations possibles 4

Chargé de TP : Zoubida AFOUTNI

OBJECTIFS DU PROJET

Le but de ce projet est de développer un Sudoku avec le langage Prolog, sans utiliser les prédicats de la partie programmation logique par contraintes de GNU-PROLOG. Nous avons donc dû adopter une logique récursive. Grâce aux cours, aux TDs et aux TPs suivis en IA02, nous avons pu apprendre à réfléchir d'une certaine manière pour résoudre des différents problèmes.
Cette application aura plusieurs fonctionnalité :

- Génération automatique de grille;
- Gestion d'un jeu de sudoku;
- Résolution automatique d'une grille donnée.

L'objectif de ce projet est donc d'appliquer les notions que nous avons apprises et passer de la théorie à la pratique.

CHOIX D'IMPLÉMENTATION

Pour pouvoir implémenter le jeu, nous avons commencé par définir nos besoins et nos objectifs de fonctionnement. Pour cela, nous avons mis en place de nombreux prédicats.

Représentation des connaissances :

- Liste de 9 listes correspondant aux lignes
- (X,Y) coordonnées : X ligne, Y colonne (ex : (3,2) pour le 7 ci-dessous, (1,8) pour le 3)
- x (minuscule) pour représenter une case vide
- Un calque est une grille avec des x et y, x correspond à n'importe quelle valeur possible entre 1 et 9 ou case vide, le y signifie qu'il s'agit d'une valeur de base de la grille : valeur inéchangeable lors d'une partie

```
[[x,x,x,x,x,x,3,x],
 [x,x,x,x,x,x,x,x,x],
 [x,7,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x],
 [x,x,x,x,x,x,x,x,x]]
```

PRÉDICATS

Voici une liste des prédicats les plus importants de notre jeu Sudoku :

- **imprime(S) :** imprime le sudoku
- **imprimeCalque(S,B) :** imprime le sudoku avec le format du calque (si valeur fixe, affiché de la manière suivante par exemple : 8~)
- **region(S,R) :** unifie les régions du sudoku S dans R sous forme de liste de 9 listes
- **get_region(X,Y,S,R) :** unifie la région de la cellule de coordonnées (X,Y)
- **transpose(S,T) :** unifie la transposé de S avec T sous forme de liste de 9 listes (on récupère les colonnes)
- **domainCheck(L) :** vérifie que tous les éléments de la liste L appartiennent au domaine des nombres compris entre 0 et 9.
- **element(N,L,X) :** Unifie le Nième élément de la liste L avec la variable X (prédicat générateur)
- **get_possibilite(X,Y,S,P) :** unifie avec P la liste des valeurs possible que peut prendre une case de coordonnées (X,Y) dans Sudoku S (on supprime de la liste de 1 à 9, toutes les valeurs dans ligne, colonne, et région sans la case actuelle, sauf si doublon)
- **case_valide(X,Y,S) :** s'efface si une case est valide (liste des possibilités non vide, valeur dans la liste ou case vide)
- **valide(S) :** vérifie si la grille S est valide (pas de case non valides)
- **solved(S) :** s'efface si S valide est sans cases vides

Paradigme Generate-and-test :

- **genere_sudoku(S,NS) :** génère un Sudoku NS issu de S après un seul ajout dans S (on utilise element et get_possibilite)
- **solve(S,Solution) :** unifie la solution de S avec Solution
- On **génère tous les sudokus possibles** tant qu'ils sont valides, et on s'arrête dès qu'ils sont complets
- **Génération de Sudoku aléatoire :** on donne un calque et on y place des valeurs possibles aléatoirement parmi la liste des possibles.
- **genere_random_coord(X,Y) :** unifie X et Y avec une coordonné aléatoire
- **genere_random_grid_y(N,S,RS) :** unifie RS avec au plus N coordonnées remplacés par y (création d'un calque aléatoire)
- **generated(S) :** s'efface si aucunes de valeurs 'y' dans S
- **genere_random_value (X,Y,S,V) :** unifie V avec une valeur aléatoire parmi les possibles pour (X,Y) dans S
- **genere_random_sudoku (S,Solution) :** unifie Solution avec un Sudoku aléatoire mais pas forcément résolvable
- **gen_r_repeat(S,NS) :** répète la génération jusqu'à en trouver un avec solution

DIFFICULTÉS RENCONTRÉES

Une des difficultés rencontrées a été le déboggage avec le langage Prolog. En effet, ce n'est pas toujours facile de trouver notre erreur avec les prédicats et nous avons quelques fois perdu beaucoup de temps sur de petites erreurs qui une fois découvertes, paraissaient très simples.
La fonction qui permet de résoudre un sudoku nous a posé quelques problèmes. En effet, elle supposait beaucoup de récession et il était très facile de se perdre dans les différentes imbrications.

NOTIONS APPRISES À TRAVERS LE PROJET

Grâce à ce projet, nous avons tout d'abord bien compris l'utilisation du Prolog avec les ensembles de prédicats qui utilisent d'autres prédicats pour répondre à des problèmes plus complexes. Nous avons donc bien acquis la logique récursive à utiliser ainsi que le découpage en étapes pour résoudre ce genre de problème qui est une cette façon un petit peu différente de réfléchir.

Nous avons aussi appris à gérer la masse de travail et bien organiser la réalisation d'un projet. En effet, nous avions tous les deux une fin de semestre chargée, nous avons donc du bien planifier ce projet pour optimiser notre temps et notre efficacité. Nous avons donc décider de toujours avancé le plus possible la veille du TP afin de pouvoir poser le plus de questions possibles à nos chargé de TP et ne pas rester bloqué trop longtemps sur des problèmes. Nous avons aussi bien réparti les taches afin de ne pas travailler sur les mêmes problèmes en même temps, tout en gardant une communication permanente afin de bien comprendre le travail de chacun.

AMÉLIORATIONS POSSIBLES

Les améliorations que nous pourrions envisager seraient premièrement de peut-être améliorer l'expérience utilisateur. Nous avons essayé de réaliser un menu et des choix de jeu qui permettraient de balayer un maximum de fonctionnalisés. Peut-être que nous aurions pu faire quelque chose de plus simple.
La deuxième chose que nous pourrions améliorer est la complexité de certaines fonctions. En effet, nous avons remarqué que certaines fonctions mettaient un peu de temps à être exécutées comme la fonction qui génère une grille aléatoirement.