Séance de mise en pratique
Utilisation d'un SAT-solver

MPSI Lycée Pierre de Fermat

Vous trouverez sur Cahier de Prépa une archive contenant les fichiers nécessaires à cette activité. Le but de cette séance est d'utiliser un SAT-solver afin de résoudre divers problèmes. L'archive contient un fichier de code "satsolver.ml". Plutôt que de l'importer dans utop pour l'utiliser en mode interactif, nous allons **compiler** le fichier, pour en faire un programme exécutable.
Compilez le code avec la commande suivante:

```
ocamlopt satsolver.ml -o sat -O3
```

Vous avez maintenant accès à un programme exécutable, que vous pouvez lancer en tapant `./satsolver nom_de_fichier.txt` dans le terminal. Ce programme lit dans le fichier que vous lui spécifiez, qui est supposé contenir une formule écrite selon un format précisé plus bas. Puis, il lance un algorithme de satisfiabilité booléenne, l'algorithme de **Quine**, afin de déterminer une éventuelle solution. Si la formule est satisfiable, il affiche la liste des variables à mettre à vrai pour satisfaire la formule (les autres restant à faux).

Format des formules Les formules devront suivre le format suivant:

- Le ET s'écrit &, le OU s'écrit "|", le NON s'écrit "$\sim$"
- L'implication s'écrit "$>$" et l'équivalence "$=$"
- La formule vraie s'écrit "T", la formule fausse s'écrit "F"
- Les parenthèses et les espaces sont autorisés, pas les retours à la ligne
- Tout le reste sera interprété comme une variable

Voici quelques exemples de formules sous ce format:

- La formule $A \vee (B \wedge \neg C)$ s'écrira "`a | (b & ~c)`"
- La priorité des opérateurs est comme suit: $\wedge$ puis $\vee$ puis $\leftrightarrow$ puis $\rightarrow$. Par exemple, si l'on écrit "`a & b | c`", cela représentera la formule $(A \wedge B) \vee C$. Dans le doute, utilisez des parenthèses pour supprimer les ambiguïtés.
- Les noms de variable peuvent avoir plusieurs lettres et même contenir des chiffres ou autres symboles (hors ceux des opérateurs), par exemple "`(X_1_2 | ~X_3_3) & (X_2_3 | ~X_1_3)`" est une formule valide.

Dans la suite de ce document, vous trouverez différents problèmes, de difficultés variables, que vous devez résoudre en utilisant ce SAT-solver. Pour chaque problème, vous devez donc procéder aux étapes suivantes:

1. Déterminer une transformation du problème en un problème SAT, i.e. trouver une formule encodant le problème.
2. Écrire un programme OCaml ou Python générant la formule en question sous le format décrit au dessus.
3. Stocker le résultat dans un fichier et lancer le SAT-solver dessus.
4. Déduire de la sortie du SAT-solver une solution au problème initial

On mettra toutes les formules sous FNC: le SAT-solver fourni est bien plus efficace sur les FNC que sur les formules quelconques. La séance est à faire en OCaml ou en Python, au choix. Sur cahier de prépa, vous trouverez une archive de code avec deux fichiers `utils.ml` et `utils.py` qui contiennent chacun une implémentation d'une fonction faisant la conjonction de formules, afin de vous aider à démarrer et à vous familiariser avec les objectifs.

Problème introductif

Q1. Déterminer si la formule suivante est satisfiable à l'aide du SAT-solver:

$$(X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee W) \wedge (\neg Y \vee Z \vee \neg W) \wedge (\neg X \vee \neg Z \vee W)$$

Q2. Écrire une fonction `au_moins_une: string list -> string)` qui prend en entrée une liste L non vide de noms de variables et qui fabrique une clause disjonctive exprimant qu'au moins une des variables de l est vraie. Par exemple:

```
let (vars: string list) = ["x"; "y"; "z"]
let (phi: string) = au_moins_une vars
// phi est la chaîne "x | y | z"
```

Q3. Écrire une fonction `au_plus_une: string list -> string)` qui prend en entrée une liste L de variables et qui fabrique une formule exprimant qu'au plus une des variables de L est vraie. Le résultat devra être sous FNC.

Nous allons utiliser ces fonctions pour résoudre le problème des N dames, qui consiste à considérer un damier d'échecs de N lignes et N colonnes sur lequel on veut placer N dames de telle sorte qu'aucune dame ne puisse en attaquer une autre. Selon les règles des échecs, il faut donc que chaque colonne, ligne, et diagonale, contienne au plus une dame.
On modélise la situation avec N^2 variables propositionnelles $(X_{i,j})_{i,j\in[\![0,N-1]\!]}$, de telle sorte que $X_{i,j}$ représente l'affirmation "la case (i,j) contient une dame". On notera ces variables "`X_i_j`" dans les formules générées. Par exemple, la formule "`~X_2_3 | ~X_2_7`" exprime qu'on ne peut pas avoir à la fois une dame en $(2,3)$ et en $(2,7)$. On veut construire une grande formule exprimant les contraintes du problème, à savoir:

- Sur chaque ligne, il y a exactement une reine;
- Sur chaque colonne, il y a au plus une reine;
- Sur chaque diagonale, il y a au plus une reine.

Q4. Écrire une fonction `variable_dame: int -> int -> string` telle que `variable_dame i j` renvoie la chaîne de caractères "`X_i_j`", i.e. la variable propositionnelle signifiant "une dame se situe en (i, j). Par exemple, `variable 7 23` renvoie la chaîne "`X_7_23`".

Q5. Écrire une fonction `contrainte_une_ligne: int -> int -> string` telle que `contrainte_une_ligne n i` renvoie une formule exprimant la contrainte sur la ligne i dans le problème des n dames.

Q6. Écrire une fonction `contrainte_toutes_lignes int -> string` telle que `contrainte_toutes_lignes n` renvoie une formule exprimant la contrainte sur toutes les lignes dans le problème des n dames.

Q7. Écrire des fonctions similaires exprimant les contraintes sur les colonnes et les diagonales.

Q8. Écrire une fonction `gen_formule_n_dames: int -> string` telle que `gen_formule_n_dames n` génère la formule modélisant le problème à n dames.

Q9. En utilisant la fonction `ecrire_fichier` de `utils.c`, générer des fichiers pour les problèmes des 3, 4, 8, 15 dames, et les faire résoudre par le SAT-solver. Dessiner les solutions lorsqu'elles existent.

Les sections suivantes décrivent chacune un problème à modéliser par des formules de logique propositionnelle.

A Coloriage de carte

On considère la carte des régions de France métropolitaine. On souhaite colorier les régions de cette carte, de telle sorte que deux régions adjacentes ne soient jamais de la même couleur. On veut utiliser le moins de couleurs possible. Est-il possible de procéder au coloriage en 4 couleurs seulement ? Et en 3 ?
Bonus: Qu'en est t-il de la carte des départements ? Du monde ?

Figure 1: Régions de France

B Problème des cinq maisons

Cinq maisons de couleurs différentes sont alignées le long d'une route. Dans chacune habite une personne de nationalité différente. Chaque personne boit une boisson différente, a un animal domestique différent, et pratique un sport différent:

1. L'Anglais vit dans une maison rouge.
2. Le Suédois a des chiens.
3. Le Danois boit du thé.
4. La maison verte est à gauche (mais pas forcément voisine) de la maison blanche.
5. Le propriétaire de la maison verte boit du café.
6. La personne qui fait du vélo a des oiseaux.
7. Le propriétaire de la maison jaune fait de la danse.
8. La personne qui vit dans la maison du centre boit du lait.
9. Le Norvégien habite la première maison.
10. La personne qui fait de l'escalade vit à côté de celle qui a des chats.
11. La personne qui a un cheval est voisine de celle qui fait de la danse.
12. La personne qui fait du basket boit du Yop.
13. L'Allemand fait du karaté.
14. Le Norvégien vit juste à côté de la maison bleue.
15. Le fan d'escalade a un voisin qui boit de l'eau.

La question est: **à qui appartient le poisson rouge ?**

C Emploi du temps

Trois professeurs de maths, info et physique veulent mettre en place un emploi du temps pour la dernière semaine de cours, afin d'organiser des séances de révisions en demi-groupes. Chaque jour est divisé en trois créneaux: 9h-12h, 12h-15h, 15h-18h. En pratique, les créneaux sont un peu plus court, ce qui permet aux élèves de manger même s'ils ont cours sur les trois créneaux. La classe dispose donc de 5 jours, chaque jour divisé en 3 créneaux, et chaque créneau peut accueillir deux cours simultanés, pour les deux demi-groupes.
En plus des contraintes structurelles évidentes (un prof ne peut pas participer à deux séances sur le même créneau, etc...), l'emploi du temps doit respecter plusieurs règles:

- Aucun élève ne doit finir deux jours d'affilée à 18h;
- Chaque élève a au plus un créneau de maths par jour;
- Le prof de maths doit aller chercher ses enfants à l'école, et doit finir ses cours avant 16h;
- La prof d'informatique ne veut pas commencer avant 10h;
- Il doit y avoir un créneau complètement libre dans la semaine pour le discours annuel du proviseur.

Au total, chaque demi-groupe doit avoir 3 créneaux de maths, 3 créneaux de physique, 3 créneaux d'informatique