TP : Méthode de résolution

La méthode de vérification de satisfiabilité d'une formule propositionnelle vue en cours a un gros défaut : elle implique de générer les 2^n valuations possibles pour toutes les vérifier une par une. La complexité explose donc, et cet algorithme est inutilisable pour des formules ayant beaucoup de variables différentes.

Nous allons voir une autre méthode, appelée *méthode de résolution*.

On appellera *littéral* toute formule qui est soit une variable, soit la négation d'une variable. On appellera *clause* toute disjonction de littéraux, et la longueur d'une clause sera ce nombre de littéraux. Par convention, la clause de longueur 0 est unique, et ne peut s'évaluer qu'en `false` : on la note $\square$.

On admettra le résultat suivant :

Proposition 0.1
Toute formule logique propositionnelle est équivalente à une conjonction de clauses.

Chaque clause sera représentée comme la liste de ses littéraux, et chaque formule comme liste de ses clauses.

```
type litteral =
  |Pos of char
  |Neg of char
type clause = litteral list
type formule_cnf = clause list
```

On dispose de la fonction `translate` qui transforme toute formule en liste des clauses de sa forme normale conjonctive.

La méthode consiste à effectuer des résolutions sur les clauses :

Définition 0.2
Soient c_1 et c_2 deux clauses. On appelle résolvante *de c_1 et c_2 toute clause c telle qu'il existe un littéral u tel que*

- *u est un littéral de c_1*
- *$\neg u$ est un littéral de c_2*
- *les littéraux de c sont exactements ceux de c_1 et c_2, excepté u et $\neg u$.*

On appelle alors *preuve par résolution* toute suite de clause telle que chaque clause de la suite est soit une clause de départ, soit une résolvante de deux clauses antérieures de la suite.

Le nombre de littéraux étant fini au départ, et diminuant strictement à chaque étape, il est clair qu'une preuve par résolution termine toujours.

On note que toute clause qui apparaît dans une preuve par résolution correspond à une formule impliquée par la formule de départ.

L'idée de la preuve de résolution consiste donc à réfuter des formules, *i.e.* d'aboutir à $\square$ avec une preuve par résolution.

On va appliquer la méthode de Davis-Putnam pour simplifier au maximum nos clauses.

1. Si une clause contient un littéral et sa négation, alors elle est triviale, et on peut la supprimer.
   Écrire une fonction `elim_tautologie` qui enlève ce type de clauses de la liste.

2. Écrire une fonction `elim_doubles` qui supprime toutes les occurences sauf une d'une clause qui apparaît plusieurs fois.

3. Écrire une fonction, qui étant donné une liste de clauses et un littéral x, renvoie la liste de clauses où on a supprimé les clauses contenant x, et supprimé $\neg x$ des clauses le contenant.

4. En déduire une fonction `propagation_unitaire` qui, pour chaque clause ne contenant qu'un littéral de la liste, applique l'opération précédente à toute la liste : c'est la *propagation unitaire*.

5. Écrire une fonction qui donne la liste (sans répétition) des littéraux de la liste de clause.

6. Écrire une fonction qui élimine les paires $p, \neg p$ de cette liste : il ne restera plus que les littéraux purs, *i.e.* qui n'apparaissent que sous forme positive ou négative.

7. Si p est un littéral pur, on peut supprimer de la liste toute les clauses contenant $\neg p$. Écrire cette fonction `elim_litt_purs` : c'est l'*élimination des littéraux purs*.

Ce sont toutes les étapes de simplification qu'on peut faire. Regardons maintenant la méthode de résolution en elle-même.

8. Écrire une fonction qui, étant donnés deux clauses c_1 et c_2, renvoie $(c_1 - u, c_2 - \neg u)$ s'il peut y avoir une résolution, (c_1, c_2) sinon.

9. Écrire une fonction qui, étant données deux liste de clauses ℓ_1 et ℓ_2, renvoie la liste de toutes les résolutions possibles entre ℓ_1 et ℓ_2.

10. En déduire une fonction `resolution` qui affiche "Satisfiable" ou "Contradiction" : une formule est une contradiction si la clause vide apparaît après des résolutions, et satisfiable si on arrive à la formule vide.