

## TP n°3

### Mise en forme Normale Conjonctive

**DIMACS** DIMACS est un format standard de description de formule sous forme normale conjonctive. Voici un exemple de fichier au format DIMACS

```
p cnf 3 2
1 -3 0
2 3 -1 0
```

la première ligne indique, après les mots clef « p cnf » le nombre de variables intervenant dans la formule (ici 3), ainsi que le nombre de clauses disjonctives (ici 2). Chaque ligne suivante représente une clause disjonctive : les variables apparaissant dans la clause sont indiqués, avec un signe - si elles apparaissent sous forme de négation. Chaque ligne se termine par un 0. Le fichier ci-dessus correspond donc à la formule suivante :

$$(x_1 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee \neg x_1)$$

**Introduction** Dans ce TP on se propose de réaliser des méthodes qui permettront de mettre une formule en forme normale conjonctive. On voudra ensuite pouvoir engendrer un fichier en format DIMACS.

Nous allons procéder en plusieurs étapes. Il s'agira d'abord de construire des méthodes qui mettent les objets de type `Formule` en forme normale conjonctive. Cela fait, il s'agira de programmer une méthode qui prendra en argument des objets de type `Formule` et qui génère des fichiers texte en format DIMACS. Ces méthodes seront écrites dans la classe classe `Formule`.

#### Mise en forme normale d'un objet de type `Formule`

**Exercice 1** Dans la classe `Formule`, programmez une méthode boolean `isNnf()` qui dit si une formule est en forme normale de négation.

**Exercice 2** Programmez une méthode `Formule` `appliqueReglesNnf()` qui retourne une formule équivalente après lui avoir appliqué une ou plusieurs des règles de mise en forme normale de négation du cours.

**Exercice 3** Programmez une méthode `Formule` `miseEnNnf()` qui retourne une formule équivalente après lui avoir appliqué tant que possible les règles de mise en forme normale de négation.

**Exercice 4** Programmez une méthode boolean `isClauseDisjonctive()` qui dit si la formule est une clause disjonctive. Programmez une méthode boolean `isCnf()` qui dit si la formule est en forme normale de conjonction.

**Exercice 5** Programmez une méthode `Formule appliqueReglesCnf()` qui suppose que la formule est déjà en forme normale de négation, et qui retourne une formule équivalente après lui avoir appliqué une ou plusieurs des règles 4.6, 4.7, 4.15 et 4.16 de mise en forme normale de conjonction.

**Exercice 6** Programmez une méthode `Formule miseEnCnf()` qui suppose que la formule est déjà en forme normale de négation, et qui retourne une formule équivalente après lui avoir appliqué tant que possible les règles de mise en forme normale de conjonction.

### Fonctions annexes

**Exercice 7** Programmez `Formule[] unionTableau (Formule[] t, Formule[] s)` qui va prendre en argument deux tableaux de formules et retourne l'union de ces deux tableaux.

**Exercice 8** Programmez une méthode `Formule[] tableauClause()` qui, quand la formule est en forme normale conjonctive, retourne un tableaux des clauses disjonctive qui y apparaissent.

**Exercice 9** Programmez une méthode `Formule[] tableauLitteraux()` qui, quand la formule est une clause disjonctive, retourne le tableaux des littéraux qui apparaissent dans cette clause.

**Exercice 10** Programmez une méthode `String ligneCnf ()` qui, quand la formule est une clause disjonctive, retourne la ligne en format DIMACS qui contient les littéraux de la formule et se termine par 0.

**Exercice 11** Programmez une méthode `String toDIMACS ()` qui utilise l'ensemble des méthodes que vous avez déjà programmées. Il s'agira de construire une chaîne qui correspond aux format DIMACS pour une forme normale conjonctive de la formule.

Pour finalement écrire la chaîne dans un fichier, on utilisera la classe `Ecrivain` fournie. Pour écrire sur un fichier, il faut créer un objet de la classe `Ecrivain`, qu'on peut voir comme un magnétophone. Le fichier est au début vide (si le fichier existait déjà il est effacé). Pour des raisons d'efficacité, le magnétophone garde le texte à écrire dans un tampon. A un instant donné, la tête d'écriture se trouve après la dernière ligne du tampon. Un appel à la méthode `writeln(String s)` écrit sur le tampon la chaîne `s` et passe à la ligne suivante. Pour écrire le contenu du tampon dans le fichier, on appel la méthode `flush()`.

Les plus avertis pourront également utiliser la classe `FileWriter`

**Exercice 12** Programmez une méthode `main` qui lit une formule d'un fichier à l'aide de la méthode `parse` de la classe `Logic` et écrit un fichier en format DIMACS qui représente une forme normale de conjonction de la formule.