

L'objectif de la séance est d'implémenter le test de satisfiabilité d'une formule du calcul propositionnel sous forme 2SAT.

1. Rappeler la définition d'une formule sous forme 2SAT.

Pour représenter une telle formule, on utilisera le type suivant :

```
type littéral =
| P of int (* positive occurrence *)
| N of int (* negative occurrence *)

type clause = littéral * littéral
type twocnf = clause list
```

On supposera que les variables présentes dans une formule sont numérotées consécutivement à partir de zéro.

Soit  $\phi = \bigwedge_{i=1}^r F_i$  une formule en 2-CNF (chaque  $F_i$  est donc de la forme  $l \vee l'$ , où  $l$  et  $l'$  ne contiennent pas la même variable) et  $x_0, \dots, x_{n-1}$  les variables présentes dans  $\phi$ . On définit le graphe  $G_\phi = (V_\phi, E_\phi)$  comme suit :

- il y a un sommet pour chaque littéral possible (autrement dit,  $2n$  sommets  $x_0, \neg x_0, \dots, x_{n-1}, \neg x_{n-1}$ );
- pour chaque clause  $l \vee l'$ , il y a deux arcs  $\neg l \rightarrow l'$  et  $\neg l' \rightarrow l$ .

2. On considère la formule suivante :

$$\phi = (x_0 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (x_3 \vee \neg x_0)$$

Représenter son graphe  $G_\phi$ .

3. Écrire une fonction `max_var : twocnf -> int` qui renvoie l'indice maximal d'une variable apparaissant dans la formule passée en entrée.
4. Écrire une fonction `graph_of_cnf : twocnf -> graph` prenant en entrée une formule  $\phi$  en 2-CNF et renvoyant le graphe  $G_\phi$  associé. On pourra supposer qu'aucune clause ne contient deux fois le même littéral, ni un littéral et son complémenté. Pour le type graphe, on utilisera `type graph = int list array`. Tester votre fonction sur le graphe `ex` fourni. (on doit obtenir `[1 [6] ; [5] ; [4] ; [6] ; [6;0] ; [3] ; [] ; [1,5,2] []]`)
5. Rappeler la condition nécessaire et suffisante sur le graphe  $G_\phi$  vue en cours qui caractérise le fait qu'une formule  $\phi$  sous forme 2SAT est satisfiable.
6. Afin d'utiliser cette caractérisation, on va implémenter l'algorithme de Kosaraju.
  - (a) Écrire une fonction `transpose` qui prend en entrée un graphe  $G$  et renvoie son graphe transposé (ou miroir).
  - (b) Écrire une fonction `post_order` qui effectue un parcours en profondeur complet d'un graphe et renvoie la liste de ses sommets par instant de fin de traitement décroissant.
  - (c) Compléter le code fourni pour obtenir une fonction `kosaraju : graph -> int list list` qui renvoie une liste contenant chaque composante fortement connexe du graphe passé en entrée. Chaque CFC est elle même une liste.
  - (d) Tester votre fonction en vérifiant que le fichier `arxiv.txt` représente un graphe à 21608 composantes fortement connexes (une fonction de lecture du fichier vous est fournie).
7. Écrire une fonction `satisfiable` déterminant si une formule  $\phi$  en 2-CNF est satisfiable. On exige une complexité linéaire en la taille de la formule (nombre d'occurrences de variables).