TP Noté - Résolution de 2-SAT

Exercice 1 :

On dispose d'un ensemble $X = \{x_1, x_2, \dots, x_n\}$ contenant n variables.

On a un ensemble de symboles Y tel que $Y = \{x_1, \neg x_1, x_2, \neg x_2, \dots, x_n, \neg x_n\}$ contenant 2n variables (chaque variable de l'ensemble X et son contraire).

Chaque clause ne peut apparaître qu'une fois et chaque inversion de la position des variables pour une même clause ne compte que pour une clause, de plus la même variable ne peut pas apparaître deux fois dans une clause.

On cherche donc ici le nombre de permutations des éléments de Y dans un sous-ensemble contenant 2 éléments, soit parmi $2n$ (taille de l'ensemble Y)

Soit :

$$\binom{2n}{2} = \frac{2n!}{(2!\,(2n-2)!)} = \frac{2n(2n-1)(2n-2)!}{2(2n-2)!} = \frac{2n(2n-1)}{2} = n(2n-1)$$

Exercice 2 :

Afin de trouver une solution ne pouvant pas être évaluée à Vrai, nous recherchons une formule F pour lequel il n'existe aucun statut initial permettant à chaque clause C de retourner 1.

Nous utilisons donc ici une formule avec $n = 2$ pour lequel la table de vérité contient au moins une clause dont le retour est 0 pour chaque état.

Pour ceci, étant donné que pour $n = 2$ nous disposons de 4 états, nous allons utiliser 4 clauses, une étant fausse pour chaque état.

$$F = (x_1 \vee x_2) \wedge (x_1 \vee !x_2) \wedge (!x_1 \vee x_2) \wedge (!x_1 \vee !x_2)$$

$$c_1 \qquad c_2 \qquad c_3 \qquad c_4$$

x_1	x_2	c_1	c_2	c_3	c_4	F
0	0	0	1	1	1	0
0	1	1	0	1	1	0
1	0	1	1	0	1	0
1	1	1	1	1	0	0

Peu importe l'état initial, il y aura ici toujours une clause fausse, la formule F n'est donc ici pas satisfaisable.

Exercice 3 :

Pour évaluer une commande 2-SAT, on crée ici une fonction **check_formula** qui prend en paramètre une formule F et un état initial de chaque variable.

Dans un premier temps on recrée l'ensemble Y à partir de l'ensemble X dans un nouveau dictionnaire en ajoutant les opposés de chaque variable. Puis on vérifie si une des clauses retourne *False* (via un opérateur Non-Ou (Ici non-a et non-b)), dans quel cas on retourne *False*, sinon *True*.

Exercice 4 :

On construit ici le graphe d'implication de la formule $F2$, on convertit chaque clause (a ou b) en couple d'implications (non-a => b) et (non-b => a).

Les fonctions **get_node** et **get_opposite** permettent ici de récupérer le nom d'une variable en notation logique (ou son opposé) afin de l'insérer dans le graphe de manière lisible.

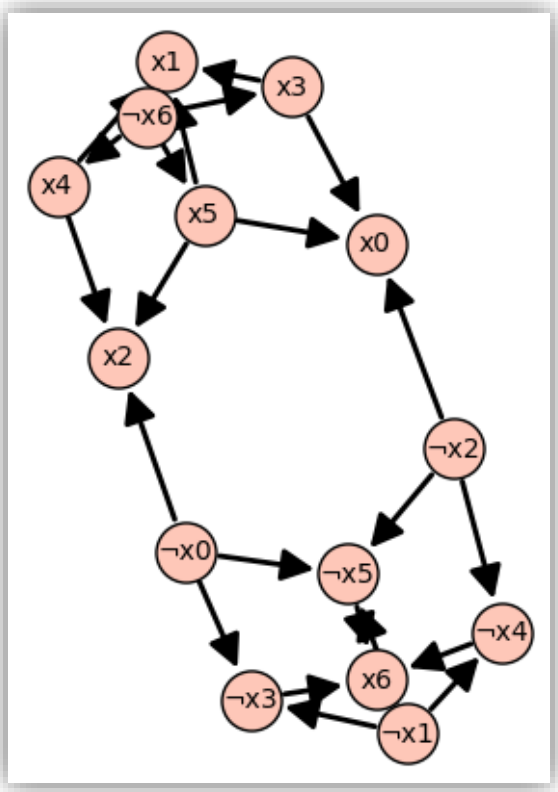

Figure 1 - Graphe de la Formule F2

Exercice 5 :

Pour cet exercice, on convertit tout d'abord le graphe en liste d'adjacence (à partir des neighbors out) afin d'y exécuter l'algorithme de Tarjan pour récupérer les composantes fortement connexes en temps linéaire $O(n)$, ensuite, on vérifie s'il y a pour une même composante fortement connexe, une variable et son opposé $(x_i \wedge \neg x_i)$, si c'est le cas, la formule est non satisfaisable étant donné qu'on aurait un cas pour lequel $x_i \Leftrightarrow \neg x_i$ (x_i si et seulement si $\neg x_i$, ce qui est impossible).

Si la formule est satisfaisable, étant donné que l'ordre retourné par l'algorithme de Tarjan des composantes est topologique, on en déduit directement une solution validant la formule 2-SAT.

La formule F1 testée dans le fichier sage fourni est celle testée pour l'Exercice 3.
La formule F2 représente une formule 2-SAT satisfaisable présente sur la page Wikipédia, pour lequel toutes les valeurs dans la solution sont à True.
La formule F3 est satisfaisable et est celle fournie pour l'Exercice 2.