# Complexité TP2 : Réductions

Réalisé par : Raphael Charpy, Akim Saadi

## Mini-projet 1. Vérificateur déterministe pour SAT

Pour le mini projet 1 nous avons décidé de le réalisé en Python. Nous avons utilisé 7 fonctions pour créer le vérificateur. Tout d'abord une fonction qui lit les fichiers affectation et une fonction qui lis le fichier de clause et les stocks dans des variables. Ensuite pour avoir une sortie propre nous avons implémenté 2 fonctions print qui nous affiche le nombre de littéraux et le nombre de clause. Bien évidement il s'en suit une fonction qui affiche la formule en entière. Et enfin la fonction qui détecte si une clause est vrai ou fausse le tout regroupé dans la fonction principale is_sat qui vérifie le problème SAT.

```
def is_sat(self) :
    for literal in self.affectation :
        for clause in self.liste_clause :
            changer_close=False
            if (clause != True) :
                for literal_clause in clause :
                    if(changer_close==False) :
                        if literal_clause == literal :
                            self.liste_clause[self.liste_clause.index(clause)]=True
                            changer_close = True
                        elif (-1 * literal_clause == literal) :
                            clause[clause.index(literal_clause)]=False
                            if (SAT.clause_is_false(clause)) :
                                print("La probléme SAT n'est pas satisfiable.")
                                return False
    print(self.nombre_clause)
    for i in range (self.nombre_clause) :
        if self.liste_clause[i] != True:
            print("La probléme SAT n'est pas satisfiable.")
            return False
    print("La probléme SAT est satisfiable.")
    return True
```

# Mini-projet 2. Réduction de Zone Vide à SAT

Pour le mini projet 2, nous avons décidé de le coder en Python, pour cela, nous avons repris nos méthodes du TP1 pour calculer la zone vide maximal qui est de la plus grande taille possible, ce qui implique que si le cardinal de la zone vide maximal est inférieur à k, alors il n'existe pas de zone vide supérieure ou égale à k. Pour le passage en SAT, nous avons fait en sorte que si deux variables représentent deux sommets qui ont une arête en commun dans G, alors elles ne peuvent pas être vrai en même temps.

```
def Zone_Vide_vers_SAT (graph,zone_vide) :
    fichier = open("clausezonevide.txt","a")
    fichier.write("p cnf 5 20\n")
    #Ici pour k=3, on oblige le SAT à avoir au moins 3 variables positifs
    fichier.write ("1 2 3 0\n1 2 4 0\n1 2 5 0\n1 3 4 0\n1 3 5 0\n1 4 5 0\n")
    fichier.write ("2 3 4 0\n2 3 5 0\n2 4 5 0\n3 4 5 0\n1 2 3 4 0\n1 2 3 5 0\n")
    fichier.write ("1 2 4 5 0\n1 3 4 5 0\n2 3 4 5 0\n1 2 3 4 5 0\n")
    for i in range (len(graph)):
        if zone_vide[i] ==1 :
            for j in range (len(graph)) :
                if (graph[i][j]==1) :
                    #Ici, on empeche que deux variable représentant des
                    #noeuds adjacents aient la même affectation
                    fichier.write('-'+str(i+1)+ ' ' +'-'+str(j+1)+' 0\n')
    fichier.close()
```

# Mini-projet 3. Réduction de Sudoku à SAT

Pour le mini projet 3, nous l'avons également réalisé sur Python. Nous avons implémenté la méthode Sudoku to SAT qui prend un Sudoku en entrée, crée des clauses de sorte à ce chaque case contient au plus une fois chaque nombre, que chaque ligne, colonne et région n'ait qu'une seule fois chaque nombre et que les nombres déjà présents dans le Sudoku soit des causes unitaires, au niveau du temps d'exécution avec Minisat, on peut voir qu'avec 10 cases à chercher, Minisat met environ 0.06s, 0.07 pour 20 cases et 0.08 pour 30.

```python
def Sudoku_to_SAT(tab) :
    fichier = open("clausesudoku.txt","a")
    n = len(tab)
    fichier.write("p cnf 729 11906\n")
    for i in range (n) :
        for j in range (n) :
            for l in range (1,n+1) :
                for m in range(1,n+1) :
                    if m > l :
                        #Chaque case contient au plus une fois chaque nombre
                        fichier.write('-'+str(i)+str(j)+str(l)+ ' ')
                        fichier.write('-'+str(i)+str(j)+str(m)+ ' ')
                        fichier.write("0\n")
    for q in range (n) :
        for p in range (1,n+1):
            for o in range (n) :
                #Chaque ligne contient au moins une fois chaque nombre
                fichier.write(str(q)+str(o)+str(p)+ ' ')
            fichier.write("0\n")
            for r in range (n) :
                #Chaque colonne contient au moins une fois chaque nombre
                fichier.write(str(r)+str(q)+str(p)+ ' ')
            fichier.write("0\n")

    for u in range (n) :
        for v in range(1,n+1) :
            for w in range(n) :
                for x in range(n):
                    if x > w :
                        #Chaque ligne contient au plus une fois chaque nombre
                        fichier.write('-'+str(u)+str(w)+str(v)+ ' ')
                        fichier.write('-'+str(u)+str(x)+str(v)+ ' ')
                        fichier.write("0\n")
                        #Chaque colonne contient au plus une fois chaque nombre
                        fichier.write('-'+str(x)+str(u)+str(v)+ ' ')
                        fichier.write('-'+str(w)+str(u)+str(v)+ ' ')
                        fichier.write("0\n")
```

```python
    #Chaque region contient au moins une fois chaque nombre
    for r in range (int(n**0.5)) :
        for s in range (int(n**0.5)) :
            for w in range(1,n+1) :
                for t in range (r*3,(r*3)+int(n**0.5)) :
                    for u in range (s*3,(s*3)+int(n**0.5)) :
                        fichier.write(str(t)+str(u)+str(w)+ ' ')
                fichier.write("0\n")
                    #Chaque region contient au plus une fois chaque nombre
                for t in range (r*3,(r*3)+int(n**0.5)) :
                    for u in range (s*3,(s*3)+int(n**0.5)) :
                        for v in range (r*3,(r*3)+int(n**0.5)) :
                            for x in range (s*3,(s*3)+int(n**0.5)) :
                                if v*10+x > t*10 + u :
                                    fichier.write('-'+str(t)+str(u)+str(w)+ ' ')
                                    fichier.write('-'+str(v)+str(x)+str(w)+ ' ')
                                    fichier.write("0\n")

    #Littéral unitaire pour les valeurs déja connu
    for i in range (n) :
        for j in range (n) :
            if tab[i][j] != 0 :
                fichier.write(str(i)+str(j)+str(tab[i][j])+ ' 0\n')

    fichier.close()
```

# Problèmes rencontrés

Nous n’avons pas pu finir le TP en entier car ces dernières semaines nous avons enchainés une semaine rendue de 4 projets puis une semaine de révision puis une semaine d’examens et enfin il nous restait 5 jours pour réaliser le projet. De plus nous avons eu des difficultés dans l’UE et lorsqu’on poser nos questions à notre responsable de TP/TD nous n’avions pas eu une réponse claire et précise. De plus il n’est pas facile en ces temps, de bien travailler à distance.