Université d'Aix-Marseille - Master Informatique 1^{ere} année

UE Complexité - TD 3 - Machines de Turing, Logique des propositions et graphes

Préambule : Dans cette planche, nous abordons les Machines de Turing Déterministes, puis la logique des propositions (qui *a priori* est maintenant maîtrisée par tous) et en particulier la compréhension de ce qu'est le **problème SAT** qui est un problème central de la Théorie de la Complexité, puis nous regardons certains liens existant avec la notion de graphe.

Exercice 1. Retour sur les Machines de Turing Déterministes

Dans ce qui suit, nous considèrerons le modèle de Machines de Turing Déterministes vu en cours.

Question 1. Définissez un programme pour Machine de Turing Déterministe dont l'entrée contient des mots définis sur l'alphabet $\Sigma = \{a,b,c\}$ et qui accepte uniquement les mots "triés", c'est-à-dire les mots compoés de a, puis de b, et enfin de c. Cela revient à reconnaître le langage $\{a^n b^m c^p : n,m,p \geq 0\}$. Vous donnerez la complexité de votre programme.

Solution. *Cette machine de Turing se déplace toujours à droite en lisant une séquence, éventuellement vide, de a puis de b puis de c jusqu'à une case vide. Dans le pire des cas, le mot w lu est une instance positive et donc $n+m+p = |w|$ symboles sont lus : la complexité est en $\Theta(|w|)$. (Quand une instance est négative, tous ses symboles ne sont pas forcément lus avant que la machine s'arrête.)*

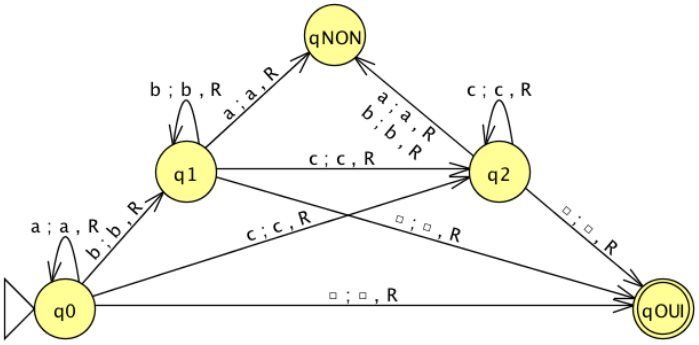
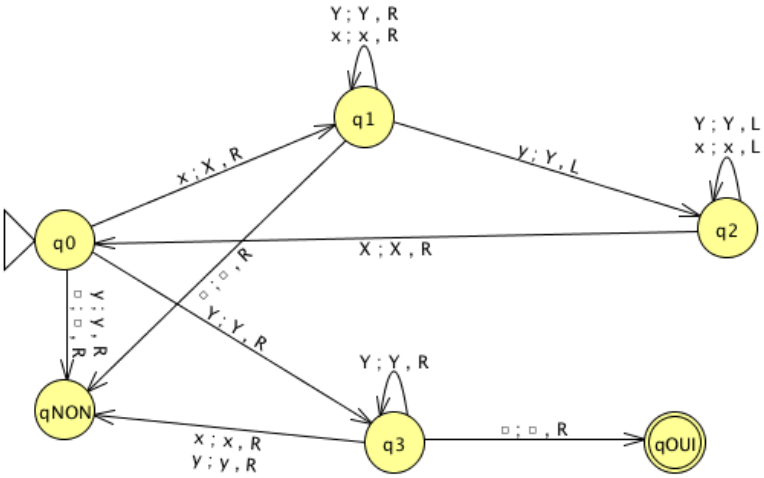


FIG. 1 – *Machine de Turing dessinée avec JFLAP (logiciel téléchargeable gratuit) avec les conventions : L pour déplacement à gauche, R pour déplacement à droite et □ pour la case vide.*

Question 2. Définissez un programme pour Machine de Turing Déterministe qui reconnaît le langage $\{x^k y^k : k > 0\}$. Vous donnerez la complexité de votre programme.

Solution. *Le principe de cette machine de Turing est de remplacer un x par X puis un y par Y et de recommencer jusqu'à ce qu'il n'y ait plus de x, puis de vérifier qu'à droite il n'y a plus que des Y suivi d'une case vide. À chaque tour de boucle q_0-q_1-q_2-q_0, le x le plus à gauche est remplacé par X puis le y le plus à gauche est remplacé par Y. Soit k le nombre de x débutants un mot w (pas nécessairement une instance positive). Au i^e tour de boucle q_0-q_1-q_2-q_0, un x est lu (et remplacé par X), puis $k-i$ x sont lus, puis $i-1$ Y, puis un y, puis on revient à gauche jusqu'au premier X qu'on trouve en lisant k - 1 symboles. Donc à chaque tour, en tout $2k+1$ transitions sont effectuées. Au pire, quand il y a au moins autant de x que de y qui les suivent, k tours de boucles sont faits. En tout, on arrive à $k(2k+1)$ transitions. Si on passe à q_3, on lit k fois Y avant de s'arrêter. Le total des transitions dans le pire des cas est donc $k(2k+1)+k \in \Theta(k^2)$.*



Question 3. Donnez un algorithme qui prend en entrée un tableau de n caractères, qui ne peut contenir que les caractères "x", "y", ou "B", et qui vérifie si ce tableau contient un mot de la forme $x^k y^k$, où k est un entier non nul, c'est-à-dire si à partir de la première case du tableau, on a une séquence de k cases contenant le caractère "x", suivies de k cases contenant le caractère "y", suivi du caractère "B", s'il reste au moins une case dans le tableau (c'est-à-dire si $2k < n$). Vous donnerez la complexité de votre algorithme, en supposant que toutes les actions élémentaires comme le test, l'affectation, l'incrémentation, etc. sont réalisables en temps constant.

Solution. *L'algorithme consiste à lire les cases du tableau de gauche à droite. Un compteur de x et un compteur de y sont initialisés à 0. À partir du début du tableau, on lit une succession de "x" en incrémentant le compteur de x puis dès qu'on n'a plus de x, on lit une succession de "y" en incrémentant le compteur de y. Quand il n'y a plus de "y", le mot est accepté si on a trouvé "B" ou si on a atteint la limite du tableau. Dans le pire des cas, chaque case du tableau est passée en revue en effectuant un nombre fixe d'opérations faites en temps constant (incrémentation, tes, affectation). La complexité est donc en $\Theta(n)$. (Cette complexité est moindre qu'avec une machine de Turing fondamentalement car les accès mémoire se font en temps constant.)*

Exercice 2. Retour sur la logique des propositions

Dans ce qui suit, nous ne considèrerons que des formules de la logique des propositions exprimées sous forme normale conjonctive (appelées FNC voire aussi CNF qui est issu de l'anglais). Leur formulation varie selon les auteurs, mais il est d'usage de considérer que dans une FNC, les clauses (disjonctions de littéraux) sont telles qu'il n'y a pas deux occurrences du même littéral et qu'il ne peut y avoir un littéral et son opposé dans une même clause.

Question 1. Un peu d'échauffement : pour se rappeler la logique des propositions

- Donner une FNC définie sur 4 variables et par 4 clauses d'arité respectivement 1 (i.e. avec 1 littéral), 2 (i.e. avec 2 littéraux), 3 (i.e. avec 3 littéraux) et 4 (i.e. avec 4 littéraux). Est-ce que cette formule est satisfaisable (donner un modèle dans ce cas) ?

  Solution. *La formule $a \wedge (\overline{a} \vee b) \wedge (\overline{a} \vee \overline{c} \vee \overline{d}) \wedge (a \vee b \vee c \vee d)$ a notamment pour modèle a=1, b=1, c=0 et d=0.*

- Donner une FNC définie sur 2 variables et par des clauses *binaires* (d'arité 2) et qui est incohérente (i.e. sans modèle ; on dit aussi *insatisfaisable*, *inconsistante* ou *antilogique*)

  Solution. *La formule $(a \vee b) \wedge (\overline{a} \vee b) \wedge (a \vee \overline{b}) \wedge (\overline{a} \vee \overline{b})$ est insatisfaisable. (C'est la seule Solution. , à un renommage des variables près)*

– Donner une FNC définie sur 3 variables et par des clauses binaires et qui est incohérente (bien évidemment, cette FNC en doit pas contenir toutes les clauses présentes dans la formule précédente)

Solution. *La formule* $(a \vee b) \wedge (\overline{a} \vee b) \wedge (a \vee \overline{b}) \wedge (c \vee \overline{b}) \wedge (\overline{a} \vee \overline{c})$ *est insatisfaisable.*

– Est-il possible de définir une FNC ne contenant que des clauses binaires et qui soit tautologique (toute interprétation est un modèle) ? Justifier votre réponse. Que pensez-vous de cette question pour des FNC avec des clauses d'arité quelconque ?

Solution. *La seule FNC qui soit tautologique est celle ne contenant aucune clause. Toute clause empêche au moins une interprétation d'être un modèle et donc toute conjonction de clauses aussi.*

– Donner une FNC définie sur 3 variables et des clauses d'arité 3 et qui admet exactement 4 modèles.

Solution. *La formule* $(a \vee b \vee c) \wedge (\overline{a} \vee b \vee c) \wedge (a \vee \overline{b} \vee c) \wedge (\overline{a} \vee \overline{b} \vee c)$ *admet quatre modèles : ceux pour lesquelles* $c = 1$ *et n'importe quelle valeurs pour* a *et* b*.*

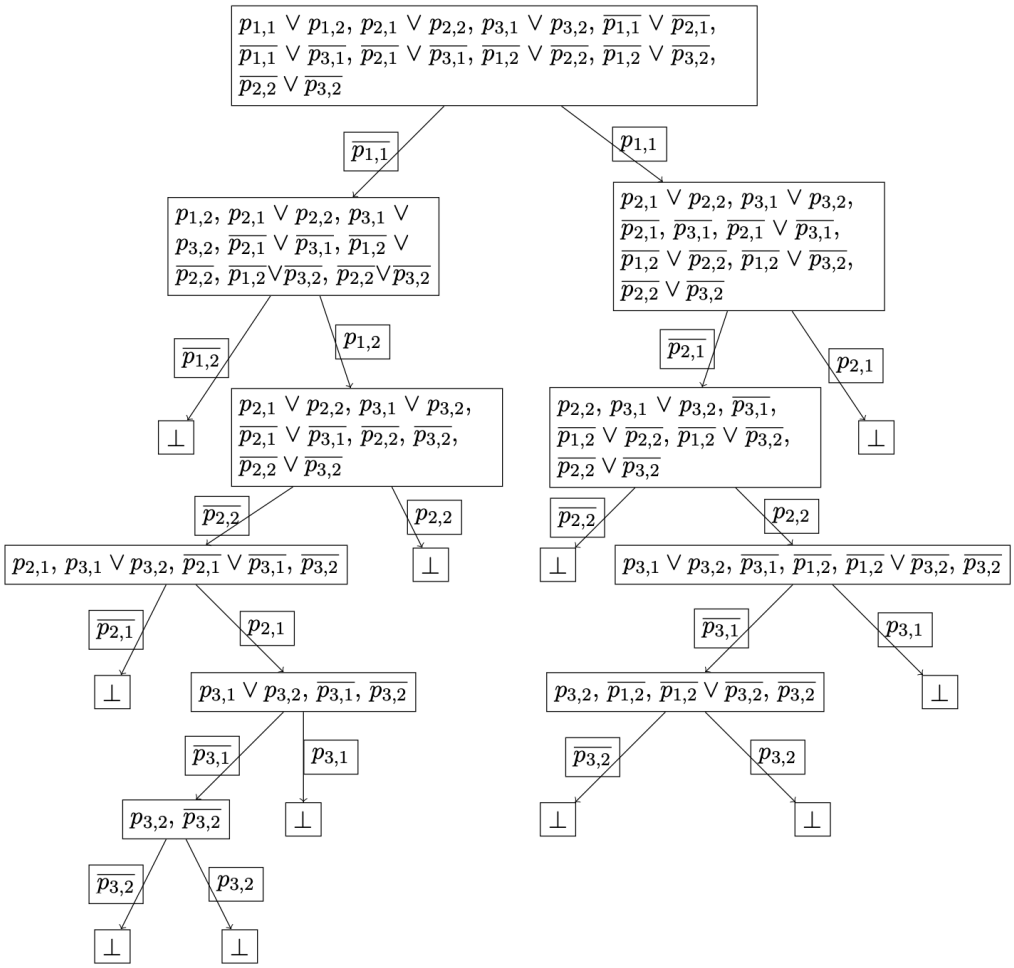
Question 2. Un peu de modélisation : le problème dit des "pigeons".

Il s'agit d'exprimer par une FNC un problème de pigeons et de pigeonnier. On considère n pigeons et 1 pigeonnier consitué de $n-1$ niches (cela s'appelle en fait un *boulin*). Dans une niche, il ne peut rentrer qu'un pigeon, et on voudrait que chaque pigeon soit dans une niche. Construisez la FNC exprimant ce problème pour 3 pigeons, et donc un pigeonnier de 2 boulins. Une aide : on peut exprimer ce problème par des variables propositionnelles de la forme p_{ij} sachant que l'interprétation à vrai de p_{ij} signifierait que le pigeon i est dans le boulin j. Ainsi, pour imposer qu'un pigeon i soit dans un des 2 boulins, on peut exprimer cela avec une clause de la forme $(p_{i1} \vee p_{i2})$ dans la FNC modélisant ce problème. En effet, satisfaire cette clause impose d'avoir soit p_{ij} vrai (signifiant que le pigeon i est dans le boulin 1), soit p_{i2} vrai (signifiant que le pigeon i est dans le boulin 2) D'autres clauses que celles de ce type doivent bien sûr être présentes dans la formule.

Solution. *De façon générale, la contrainte que chacun des* n *pigeons doit être dans un des* $n-1$ *boulins s'exprime par :* $\forall i,\ 1 \leq i \leq n,\ p_{i,1} \vee p_{i,2} \vee ... \vee p_{i,n-1}$*. La contrainte que deux pigeons ne peuvent pas être dans le même boulin s'exprime par :* $\forall i,\ 1 \leq i \leq n,\ 1 \leq j \leq n,\ i \neq j,\ \forall k,\ 1 \leq k \leq n-1,\ p_{i,k} \implies \overline{p_{j,k}}$*, qui se réécrit sous forme clausale* $\overline{p_{i,k}} \vee \overline{p_{j,k}}$*. Quand* $n = 3$*, on a donc les clauses :* $p_{1,1} \vee p_{1,2}$*,* $p_{2,1} \vee p_{2,2}$*,* $p_{3,1} \vee p_{3,2}$*,* $\overline{p_{1,1}} \vee \overline{p_{2,1}}$*,* $\overline{p_{1,1}} \vee \overline{p_{3,1}}$*,* $\overline{p_{2,1}} \vee \overline{p_{3,1}}$*,* $\overline{p_{1,2}} \vee \overline{p_{2,2}}$*,* $\overline{p_{1,2}} \vee \overline{p_{3,2}}$*,* $\overline{p_{2,2}} \vee \overline{p_{3,2}}$*.*

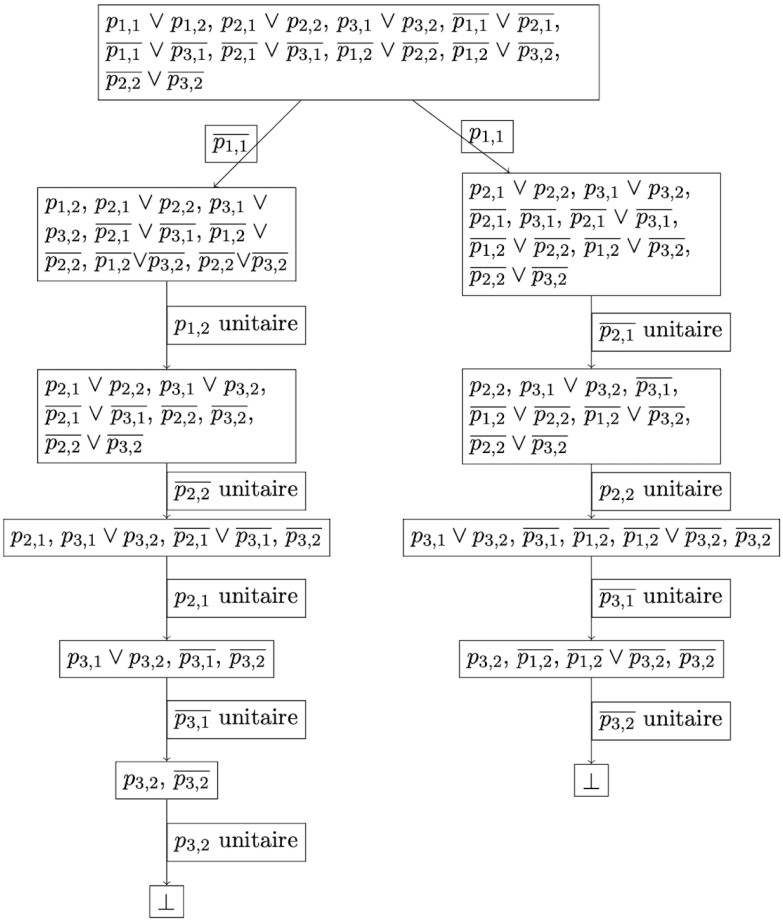
Question 3. Appliquer l'algorithme de Quine à la formule FNC de la question précédente. Vous dessinerez l'arborescence sémantique correspondante.

Solution. *Voir page suivante.*


$p_{1,1} \vee p_{1,2}$, $p_{2,1} \vee p_{2,2}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{1,1}} \vee \overline{p_{2,1}}$, $\overline{p_{1,1}} \vee \overline{p_{3,1}}$, $\overline{p_{2,1}} \vee \overline{p_{3,1}}$, $p_{1,2} \vee \overline{p_{2,2}}$, $p_{1,2} \vee \overline{p_{3,2}}$, $\overline{p_{2,2}} \vee \overline{p_{3,2}}$
$\overline{p_{1,1}}$
$p_{1,1}$
$p_{1,2}$, $p_{2,1} \vee p_{2,2}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{2,1}} \vee \overline{p_{3,1}}$, $\overline{p_{1,2}} \vee \overline{p_{2,2}} \vee \overline{p_{3,2}}$, $\overline{p_{2,2}} \vee \overline{p_{3,2}}$
$p_{2,1} \vee p_{2,2}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{2,1}} \vee \overline{p_{3,1}}$, $\overline{p_{1,2}} \vee \overline{p_{2,2}}$, $\overline{p_{1,2}} \vee \overline{p_{3,2}}$, $\overline{p_{2,2}} \vee \overline{p_{3,2}}$
$\overline{p_{1,2}}$
$p_{1,2}$
$\overline{p_{2,1}}$
$p_{2,1}$
$\bot$
$p_{2,1} \vee p_{2,2}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{2,1}} \vee \overline{p_{3,1}}$, $\overline{p_{2,2}}$, $\overline{p_{3,2}}$, $\overline{p_{2,2}} \vee \overline{p_{3,2}}$
$p_{2,2}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{3,1}}$, $\overline{p_{1,2}} \vee \overline{p_{2,2}}$, $\overline{p_{1,2}} \vee \overline{p_{3,2}}$, $\overline{p_{2,2}} \vee \overline{p_{3,2}}$
$\bot$
$\overline{p_{2,2}}$
$p_{2,2}$
$\overline{p_{2,2}}$
$p_{2,2}$
$p_{2,1}$, $p_{3,1} \vee p_{3,2}$, $\overline{p_{2,1}} \vee \overline{p_{3,1}}$, $\overline{p_{3,2}}$
$\bot$
$\bot$
$p_{3,1} \vee p_{3,2}$, $\overline{p_{3,1}}$, $\overline{p_{1,2}}$, $\overline{p_{1,2}} \vee \overline{p_{3,2}}$, $\overline{p_{3,2}}$
$\overline{p_{2,1}}$
$p_{2,1}$
$\overline{p_{3,1}}$
$p_{3,1}$
$\bot$
$p_{3,1} \vee p_{3,2}$, $\overline{p_{3,1}}$, $\overline{p_{3,2}}$
$p_{3,2}$, $\overline{p_{1,2}}$, $\overline{p_{1,2}} \vee \overline{p_{3,2}}$, $\overline{p_{3,2}}$
$\bot$
$\overline{p_{3,1}}$
$p_{3,1}$
$\overline{p_{3,2}}$
$p_{3,2}$
$p_{3,2}$, $\overline{p_{3,2}}$
$\bot$
$\bot$
$\overline{p_{3,2}}$
$p_{3,2}$
$\bot$
$\bot$

Question 4. Appliquer l'algorithme DPLL à la formule FNC de la question précédente. Vous dessinerez l'arborescence sémantique correspondante.

Solution.



NB : on aurait pu aussi appliquer la règle du littéral pur sur la racine du premier sous-arbre droit : $\overline{p_{2,1}}$ est un littéral pur, ce qui permet de supprimer les deux clauses le contenant.

Exercice 3. Le problème SAT, ses extensions et quelques restrictions

Le problème de la satisfaisabilité en logique des propositions, appelé SAT, s'exprime par :

> **SAT**
> **Donnée** : Une formule F de la logique des propositions exprimée par une FNC.
> **Question** : La formule F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Avant tout, précisons que SAT est parfois défini avec des instances étant des formules quelconques de la logique des propositions (i.e. pas nécessairement des FNC). Cela étant, la restriction que nous donnons ici n'a pas d'incidence au niveau de l'étude de cette question. À partir de ce problème, nous allons exprimer ses extensions puis étudier certaines restrictions.

Question 1. Extensions de SAT.

1. Définir SAT-RECHERCHE, le problème de recherche associé à SAT.
   Solution.
   ***Question** : Donnez un modèle de la formule F s'il en existe au moins un.*

2. Définir MAX-SAT, le problème d'optimisation associé à SAT. Ici, le critère d'optimisation à considérer est le nombre de clauses pouvant être satisfaites pour une formule F, que cette formule soit satisfaisable ou non.
   Solution.
   ***Question** : Donnez une interprétation de la formule F qui maximise le nombre de clauses satisfaites.*

3. Définir #SAT, le problème de dénombrement associé à SAT.
   Solution.
   ***Question** : Combien la formule F a-t-elle de modèles ?*

4. Définir ENUM-SAT, le problème d'énumération associé à SAT.
   Solution.
   ***Question** : Énumérez tous les modèles de la formule F.*

Question 2. Une CNF dont chaque clause possède exactement k littéraux (tous différents comme indiqué en préambule) est appelée formule *k-SAT*. Cela permet d'exprimer des instances de SAT sous forme uniforme et aussi de définir des variantes simplifiées de SAT dans les termes suivants :

> **k-SAT**
> **Donnée** : Une CNF F dont toutes les clauses ont k littéraux.
> **Question** : La formule F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Notons que cette formulation varie selon les auteurs, en ce sens que parfois on appelle *instances k-SAT*, les instances de SAT dont les clauses ont au plus k littéraux. Mais il est d'usage de considérer que k-SAT porte précisément sur les instances de SAT dont toutes les clauses ont exactement k littéraux. Pour le cas où toutes les clauses ont 2 littéraux, le problème est appelé 2-SAT. Il vous faut dans cette question proposer une méthode efficace (i.e. de complexité polynomiale) pour résoudre le probleme 2-SAT.

Solution. *Toutes les clauses sont de forme $l_1 \vee l_2$ où l_1 et l_2 sont des littéraux. L'idée fondamentale est que $l_1 \vee l_2$ équivaut à $\overline{l_1} \implies l_2$ et $\overline{l_2} \implies l_1$. Pour déterminer si une formule 2-SAT est insatisfaisable, il suffit de trouver un littéral l qui implique $\overline{l}$, par une suite d'implications induites par les clauses binaires, et que $\overline{l}$ implique aussi l par une autre suite d'implications. Cela peut se faire en construisant le graphe orienté G=(S,A) dont S contient tous les littéraux de la formule et l'arc (l, l') existe ssi $l \implies l'$. À partir de là, il suffit calculer les composantes fortement connexes de G puis de considérer chacun des sommets l de S et de vérifier si l et $\overline{l}$ sont dans la même composante fortement connexe. Dans ce cas, cela signifie qu'il y a un chemin de l vers $\overline{l}$ et un chemin de $\overline{l}$ vers l. Si c'est le cas, la formule est insatisfaisable, sinon, elle est satisfaisable. La recherche des composantes fortement connexes est réalisable en temps linéaire dans la taille de la représentation du graphe dont le nombre de sommets est de l'ordre du nombre de variables propositionnelles et le nombre d'arcs est de l'ordre du nombre de clauses. Cet algortihme a été vu en licence (et rappelé au début de cette UE). Une fois les composantes fortement connexes calculées, il suffit donc de vérifier s'il existe une variable propositionnelle x telle que $\overline{x}$ est dans la même composante fortement connexe. La méthode est donc de complexité polynomiale.*

Autre solution possible. *Le système de preuve par résolution est complet. En produisant toutes les résolvantes possibles à partir de la formule et en itérant jusqu'à ce qu'il n'y ait plus de résolvante nouvelle à produire, on obtient que la formule est insatisfaisable ssi on a pu produire la clause vide. Comme les clauses sont de taille 2 au départ, les résolvantes n'auront jamais une taille supérieure à 2. Or, avec n variables, il ne peut exister que $\frac{2n(2n+1)}{2}$ clauses binaires, $2n$ clauses unitaires et la clause vide. Le nombre de résolvantes produites avant de conclure à la satisfaisabilité est donc borné par une fonction en $\Theta(n^2)$.*

Question 3. La restrictions aux clauses de Horn.

Dans ce qui suit, nous nous intéressons à un autre type de restriction de SAT, appelée *Horn-SAT* et qui considère les CNF exprimées avec des clauses de Horn, à savoir des clauses qui ont au plus 1 littéral positif. À partir de cela, on peut définir le problème suivant :

> **Horn-SAT**
> **Donnée** : Une CNF F dont toutes les clauses sont des clauses de Horn.
> **Question** : La formules F est-elle satisfaisable, i.e. est-ce que F possède un modèle ?

Vous devez traiter les 4 questions suivantes :

1. Que pensez-vous de la satisfaisabilité d'une instance Horn-SAT quand aucune clause n'est d'arité 1 ?

Solution. *Si une clause de Horn a au moins deux littéraux, au moins un littéral est négatif. Si toutes les clauses ont au moins deux littéraux, il suffit d'affecter la valeur faux à toutes les variables pour satisfaire au moins un littéral négatif par clause.*

2. Que pensez-vous de la satisfaisabilité d'une instance Horn-SAT quand il peut y avoir des clauses d'arité 1 ?

Solution. *Pour qu'une clause unitaire soit satisfaite, il faut affecter sa variable à vrai si le littéral est positif et à faux s'il est négatif. Une condition nécessaire pour qu'un modèle existe est de satisfaire les clauses unitaires par affectation de leur variable. Cependant, si deux clauses unitaires ont des littéraux opposés, la formule est insatisfaisable. Une fois les valeurs des variables de clauses unitaires choisies, on peut simplifier la formule et reconsidérer sa satisfaisabilité.*

3. Indiquez une méthode pour trouver un modèle d'une instance Horn-SAT, s'il en existe un.

Solution.

```
En entréee : une formule Horn-SAT F
En sortie : un booléeen exprimant sa satisfaisabilité
Répeter
   Si F est vide ou si elle ne contient que des clauses contenant au moins deux littéraux
      retourner VRAI
   Si F contient la clause vide
      retourner FAUX
   Choisir une clause unitaire x
   Supprimer de F les clauses contenant x
   Supprimer le littéral opposé à x de toutes les clauses de F
```

4. Si le problème Inv-Horn-SAT considère la restriction de SAT aux instances de SAT pour lesquelles les formules sont exprimées par des clauses qui ont au plus 1 littéral négatif, que pensez-vous des réponses aux questions précédentes pour Inv-Horn-SAT ?

Solution. *Si on inverse les signes de tous les littéraux d'une formule Inv-Horn-SAT, on obtient une formule Horn-SAT qui a la même satisfiabilité que la précédente : les modèles de l'une s'obtiennent en inversant les valeurs de l'autre.*

Exercice 4. Implémentation de l'interprétation d'une formules FNC.

Dans ce qui suit, nous allons considérer des formules de la logique des propositions exprimées sous forme normale conjonctive et dont l'arité des clauses est 3 (on considère donc des instances de 3-SAT). Ces formules

sont traditionnellement représentées dans des fichiers (vous verrez cela plus en détail notamment en TP) dans lesquels une clause de la forme $(x_1 \vee \neg x_3 \vee x_4)$ sera exprimée par la suite de 3 nombres 1, -3 et 4 (l'indice de la variable est positif ou négatif selon la nature du littéral). On supposera ici qu'une FNC définie sur n variables et par m clauses ternaires est stockée dans un tableau à 1 dimension dont les m premières cases contiennent chacune une des m clauses, chaque clause étant représentée dans un tableau d'entiers à 1 dimension et de taille 3. Une interprétation des n variables propositionnelles est pour sa part représentée dans un tableau de booléens dont les n premières cases contiennent chacune l'interprétation de la variable dont l'indice est celui de la case du tableau : si `inter` est le tableau, alors `inter[i]` a pour valeur l'interprétation de la variable dont l'indice est i, c'est-à-dire x_i.

À partir de cette représentation des FNC et des interprétations, il faut donnez un algorithme qui prend en entrées une telle formule et une interprétation de ses variable et qui fournit en résultat l'évaluation de la formule, soit *vrai*, soit *faux*. Donnez une évaluation de sa complexité.

Solution. *L'algorithme examine les m clauses une par une en cherchant un littéral dans chaque clause qui serait satisfait par l'interprétation. La boucle Tantque externe est exécutée m fois au maximum (quand I valide F) et la boucle interne 3 fois maximum. Les affectations, comparaisons et incrémentations se font en temps constant. On est donc en $\Theta(m)$.*

```
En entrée : une formule 3-SAT F et une interprétation I
En sortie : VRAI si I valide F
m = |F|
n = |I|
i = 0
clauseSat = VRAI
Tantque i < m ET clauseSat == VRAI
   j = 0
   unLittSat = FAUX
   Tantque j < 3 ET unLittSat == FAUX
      Si F[i][j] < 0
         unLittSat = NON I[-F[i][j]]
      Sinon
         unLittSat = I[F[i][j]]
      j = j + 1
   clauseSat = unLittSat
   i = i + 1
retourne clauseSat
```

Exercice 5. Un problème classique de graphes et son expression en logique

Dans ce qui suit, nous nous intéressons au célèbre problème de la *coloration de graphes*. Nous rappelons qu'un graphe non-orienté peut être colorié avec k couleurs si on peut affecter une couleur parmi k possibles, à chacun de ses sommets, de telle sorte que deux sommets reliés par une arête aient des couleurs différentes. Nous allons nous limiter ici au cas d'une palette à 3 couleurs, ce qui permet de définir le problème suivant :

> **3-COLORATION**
> **Donnée** : Un graphe non-orienté $G = (S,A)$
> **Question** : Existe-t-il une coloration de G avec 3 couleurs ?

Question 1. Montrer comment une instance du problème 3-COLORATION peut être exprimée avec une formule de la logique des propositions exprimée sous forme normale conjonctive. Vous préciserez la taille de la formule pour le cas d'un graphe de n sommets et de m arêtes.

Solution. *Une variable $x_{i,j}$ signifie : le sommet i a la couleur j. Il y a $n \times 3$ variables.*

- *Chaque sommet possède (au moins) une couleur :* $\forall i,\ 1 \leq i \leq n : x_{i,1} \vee x_{i,2} \vee x_{i,3}$ *(n clauses de taille 3)*
- *Deux couleurs différentes ne peuvent pas colorer le même sommet:* $\forall i,\ 1 \leq i \leq n : \overline{x_{i,1}} \vee \overline{x_{i,2}}$, $\overline{x_{i,1}} \vee \overline{x_{i,3}}$, $\overline{x_{i,2}} \vee \overline{x_{i,2}}$ *(3n clauses de taille 2)*
- *Deux sommets voisins ne peuvent pas avoir la même couleur :* $\forall \{i,j\} \in A : \overline{x_{i,1}} \vee \overline{x_{j,1}}$, $\overline{x_{i,2}} \vee \overline{x_{j,2}}$, $\overline{x_{i,3}} \vee \overline{x_{j,3}}$ *(3m clauses de taille 2)*

Question 2. Donnez un algorithme qui prend en entrée un graphe non-orienté $G = (S,A)$ et qui construit une telle formule associée au problème 3-COLORATION. Donnez sa complexité.

Solution. *Complexité : en $\Theta(n^2)$. On aurait pu être en $\Theta(n+m)$ si on avait utilisé les listes d'adjacence pour représenter G.*

```
En entrée : un entier n et une matrice carrée G de coté taille n
PourTout i de 0 à n
   Ecrire:  3i+1 3i+2 3i+3
   Ecrire: -3i-1 -3i-2
   Ecrire: -3i-1 -3i-3
   Ecrire: -3i-2 -3i-3
   PourTout j de i+1 à n
      Si G[i][j] == VRAI
         Ecrire: -3i-1 -3j-1
         Ecrire: -3i-2 -3j-2
         Ecrire: -3i-3 -3j-3
```