Notes de cours – CM6

1 Le problème SAT

1.1 Variables

Une **variable** dans le problème SAT est une entité logique qui peut prendre deux valeurs : *vrai* (`True`) ou *faux* (`False`). Par exemple, une variable x peut être affectée à $x = \texttt{True}$ ou $x = \texttt{False}$.

1.2 Formule

Une **formule** est une combinaison de variables connectées par des opérateurs logiques :
— **ET** ($\wedge$) : la conjonction.
— **OU** ($\vee$) : la disjonction.
— **NON** ($\neg$) : la négation.
— **IMPLIQUE** ($\rightarrow$) : l'implication.
Exemple : $(x_1 \vee \neg x_2) \wedge x_3$ est une formule logique.

1.3 Valeur de vérité

Chaque **affectation** des variables donne une **valeur de vérité** à la formule.
Exemple : pour la formule $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$, l'affectation $x_1 = \texttt{True}, x_2 = \texttt{True}$ satisfait la formule.
— S'il existe une affectation qui satisfait la formule, la formule est **satisfiable**.
— Sinon, elle est **insatisfiable**.

1.4 Problème SAT

Le **problème SAT** consiste à déterminer si une formule donnée est satisfiable.

1.5 CNF, clauses et littéraux

Il est parfois plus pratique de travailler avec des formules dont la forme est un peu plus contrainte. En général, nos formules sont sous **forme normale conjonctive (CNF)**, c'est-à-dire que ce sont des conjonctions ($\wedge$) de disjonctions de variables ou de leur négation.

Un peu de vocabulaire :
— Un **littéral** est soit une variable (x), soit sa négation ($\neg x$).
— Une **clause** est une disjonction ($\vee$) de **littéraux**.
— Une formule sous CNF est une conjonction de clauses.
Exemple : $(x_1 \vee \neg x_2) \wedge (\neg x_3)$ est en CNF. Les clauses sont $(x_1 \vee \neg x_2)$ et $(\neg x_3)$. Les littéraux sont $x_1, \neg x_2, \neg x_3$.

Une formule sous CNF est satisfiable s'il existe une valuation qui satisfait toutes ses clauses. On appelle k-SAT le problème SAT restreint aux formules sous CNF où chaque clause contient au plus k littéraux.

2 Modélisation

2.1 2-Coloration

Le problème de la **2-coloration** consiste à attribuer deux couleurs (1 et 2) aux sommets d'un graphe $G = (V, E)$ de manière à ce que deux sommets adjacents aient des couleurs différentes.

2.1.1 Modélisation en SAT

- Pour chaque sommet $v \in V$, introduire une variable x_v, où $x_v = \texttt{True}$ représente v colorié avec 1 et $x_v = \texttt{False}$ représente v colorié avec 2.
- Pour chaque arête $(u, v) \in E$, ajouter une clause :

$$(x_u \vee x_v) \wedge (\neg x_u \vee \neg x_v)$$

  Cette contrainte garantit que u et v n'ont pas la même couleur.

2.2 3-Coloration

Le problème de la **3-coloration** consiste à attribuer trois couleurs $(1, 2, 3)$ aux sommets d'un graphe G de manière à ce que deux sommets adjacents aient des couleurs différentes.

2.2.1 Modélisation en SAT

- Pour chaque sommet $v \in V$, introduire trois variables $x_{v,1}, x_{v,2}, x_{v,3}$, où $x_{v,i} = \texttt{True}$ signifie que v est colorié avec la couleur c_i.
- Ajouter les contraintes suivantes :
  - Chaque sommet a exactement une couleur :

$$(x_{v,1} \vee x_{v,2} \vee x_{v,3}) \wedge (\neg x_{v,1} \vee \neg x_{v,2}) \wedge (\neg x_{v,1} \vee \neg x_{v,3}) \wedge (\neg x_{v,2} \vee \neg x_{v,3})$$

  - Deux sommets adjacents n'ont pas la même couleur :
    Pour chaque $(u, v) \in E$, ajouter :

$$(\neg x_{u,1} \vee \neg x_{v,1}) \wedge (\neg x_{u,2} \vee \neg x_{v,2}) \wedge (\neg x_{u,3} \vee \neg x_{v,3})$$

2.3 Sudoku

Le **Sudoku** est un problème où chaque ligne, colonne et région d'une grille doit contenir exactement une fois chaque chiffre de 1 à 9.

2.3.1 Modélisation en SAT

- Pour chaque cellule (i, j), introduire 9 variables $x_{i,j,k}$, où $x_{i,j,k} = \texttt{True}$ signifie que la cellule (i, j) contient le chiffre k.
- Ajouter les contraintes suivantes :
  - Chaque cellule contient exactement un chiffre :

$$(x_{i,j,1} \vee x_{i,j,2} \vee \cdots \vee x_{i,j,9}) \wedge (\text{au plus un chiffre par cellule}).$$

  - Chaque ligne, colonne et région contient chaque chiffre une seule fois. Par exemple, pour les lignes :

$$\bigwedge_{k=1}^{9} \bigwedge_{i=1}^{9} \bigwedge_{j_1 < j_2} (\neg x_{i,j_1,k} \vee \neg x_{i,j_2,k}).$$

3 2-SAT

Le problème **2-SAT** est un cas particulier de SAT où chaque clause contient au plus 2 littéraux. Ce problème est intéressant car il est **résoluble en temps polynomial**, contrairement au problème SAT général (NP-complet).

3.1 Exemple de 2-SAT

Une formule comme $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3)$ est un problème 2-SAT.

3.2 Algorithme de résolution

On peut remarquer que chaque clause $(a \vee b)$ est équivalente à $\neg a \to b$ et à $\neg b \to a$. On peut donc représenter une instance par un graphe d'implications.

Graphe d'implications : Construire un graphe dirigé où chaque variable et sa négation sont des nœuds. Pour chaque clause $(x \vee y)$, ajouter les arêtes $\neg x \to y$ et $\neg y \to x$.

On peut remarquer que pour chaque paire de sommets u, v s'il existe un chemin de u à v et de v à u, alors u et v doivent avoir la même valeur de vérité pour que la formule soit satisfiable. En particulier, si $v = \neg u$, alors la formule n'est pas satisfiable. On peut en fait montrer que la réciproque est aussi vraie.

Ainsi, pour résoudre 2-SAT, il suffit de **calculer les composantes fortement connexes (CFC)** par l'algorithme de Kosaraju, puis de tester si une variable et sa négation se trouvent dans la même CFC. Si c'est le cas, la formule est insatisfiable. Sinon, on peut aussi trouver une affectation valide en la définissant sur chaque CFC en suivant un ordre bien choisi.

Notes partiellement générées par ChatGPT.