Le problème SAT

Dans ce chapitre, on s'intéresse à la satisfiabilité des formules (sur un ensemble de variables fini, i.e. card $(Q) \in \mathbb{N}$). En effet, une formule propositionnelle peut modéliser un problème concret dont une solution serait donnée par un environnement satisfaisant la formule.

1 Définitions

Exemple: La valeur de vérité d'une variable $p_{i,j,k}$ indique si la case en (i, j) contient la valeur k, l'environnement complet décrit une solution, et s'il satisfait la formule, alors c'est une solution valide.

Mais on s'intéresse déjà au problème de décision (plus simple) de savoir si une formule est satisfiable, sans demander par quel environnement.

SAT	Entrée:	$A \in \mathbb{F}_p(Q)$
	Sortie:	oui si A est satifsiable, non sinon.

Remarque: Puisque la satisfiabilité d'une formule ne dépend que de sa classe, on peut résoudre le problème sur une formule ayant une forme particulière, mais attention au coût de transformation.

Ce problème a plusieurs variantes :

FND-SAT	Entrée:	$A \in \mathbb{F}_p(Q)$ sous FND.
	Sortie:	oui si A est satisfiable, non sinon.
FNC-SAT	Entrée:	$A \in \mathbb{F}_p(Q)$ sous FNC.
	Sortie:	oui si A est satisfiable, non sinon.
3-SAT	Entrée:	$A \in \mathbb{F}_p(Q)$ sous FNC avec des clauses de 3 littéraux seulement.
	Sortie:	oui si A est satisfiable, non sinon.
2-SAT	Entrée:	$A \in \mathbb{F}_p(Q)$ sous FNC avec des clauses de 2 littéraux seulement.
	Sortie:	oui si A est satisfiable, non sinon.

2 Réductions

Soient X et Y deux problèmes de décisions. On note X^+ (resp. X^-) les instances positives (resp. négatives) de X, i.e. celles pour lesquelles la solution est vrai (resp. faux). On définit de même Y^+ et Y^-.

On dit que Y **se réduit à** X **en temps polynomial** s'il existe une transformation φ, calculable en temps polynomial, qui transforme toute instance de Y en une instance de X de sorte que :

$$\forall \mathcal{I}, \quad \varphi(\mathcal{I}) \in X^+ \Leftrightarrow \mathcal{I} \in Y^+$$

On a alors, pour les problèmes définis précédemment :

- **SAT** est plus dur que **FND-SAT** et **FNC-SAT** :

  - **FNC-SAT** se réduit en temps polynomial à **SAT** (pour $\varphi = Id$).
  - **FND-SAT** se réduit en temps polynomial à **SAT** (pour $\varphi = Id$).
- **FNC-SAT** est plus dur que **2-SAT** et **3-SAT** :
  - **2-SAT** se réduit en temps polynomial à **FNC-SAT** (pour $\varphi = Id$).
  - **3-SAT** se réduit en temps polynomial à **FNC-SAT** (pour $\varphi = Id$).
- **FNC-SAT** se réduit à **3-SAT** :
  - On transforme chaque clause de taille inférieure à 3 en clause de taille 3 :

$$a \equiv (a \vee a \vee a) \qquad (a \vee b) \equiv (a \vee a \vee b)$$

  - On transforme chaque clause de taille supérieure à 3 en conjonction de clauses de taille 3 en ajoutant des nouvelles variables :

$$(a \vee b \vee c \vee d) \longrightarrow (a \vee b \vee z) \wedge ((\neg z) \vee c \vee d)$$

$$(a \vee b \vee c \vee d \vee e) \longrightarrow (a \vee b \vee z_1) \wedge ((\neg z_1) \vee c \vee z_2) \wedge ((\neg z_2) \vee d \vee e)$$

  - Toutes ces transformations sont en temps polynomial et préservent la satisfiablilité.

Exercice 2: Soit $C = \ell_1 \vee \ell_2 \vee \ell_3 \vee ... \vee \ell_k$ une clause de $\mathbb{F}_p(Q)$ de taille $k > 3$. Soit z une nouvelle variable (i.e $z \notin Q$). On pose alors $C_1 = (\ell_1 \vee \ell_2 \vee z)$ et $C_2 = ((\neg z) \vee \ell_3 \vee ... \vee \ell_k)$. Montrer les deux propositions suivantes :

- $\forall \rho \in \mathbb{B}^Q$, $[C]^\rho = V \Rightarrow (\exists \rho' \in \mathbb{B}^{Q \cup \{z\}}, \rho'|_Q = \rho$ et $[C_1 \wedge C_2]^{\rho'} = V)$
- $\forall \rho \in \mathbb{B}^{Q \cup \{z\}}, [C_1 \wedge C_2] = V \Rightarrow [C]^{\rho'} = V$

Remarque: **FNC-SAT** ne se réduit pas à **2-SAT**. On verra que **2-SAT** se résout en temps polynomial (cf. chapitre sur les graphes), que **3-SAT** est NP-complet (théorème de Cook, au programme de seconde année), et qu'à moins que P = NP, ces deux problèmes ne sont pas équivalents.

Remarque: **FNC-SAT** ne se réduit pas à **FND-SAT** : La transformation d'une forme à l'autre se fait en temps exponentiel et peut aussi augmenter la taille de l'entrée de manière exponentielle. On verra que **FND-SAT** se résout en temps polynomial (juste après). À moins que P = NP, ces deux problèmes ne sont pas équivalents.

3 Modéliser une FND

On rappelle qu'une formule sous FND est une disjonction de conjonction de littéraux. D'un point de vue sémantique, c'est à dire dans $\mathbb{F}_p(Q)$,

- L'ordre des littéraux au sein d'une conjonction n'a pas d'importance, ni leur multiplicité : une conjonction est satisfaite par un environnement propositionnel ssi il satisfait l'ensemble de ses termes.
- L'ordre des conjonctions n'a pas non plus d'importance au sein de la disjonction, ni leur multiplicité : une conjonction est satisfaite par un environnement propositionnel ssi il satisfait l'un de ses termes.
- Pour modéliser les conjonctions de littéraux, il est intéressant d'utiliser un **ensemble de littéraux** ou un **couple d'ensembles de variables** : d'une part celles apparaissant dans les littéraux positifs, d'autre part celles apparaissant dans les littéraux négatifs.
- Pour modéliser une FND, il faut alors utiliser un ensemble de conjonctions. Quant à la structure de données à employer, une liste de littéraux convient.

4 FND-SAT et FND-SAT

On suppose que $Q = \{q_1, q_2, ..., q_N\}$. Avec la modélisation décrite précédemment, les deux problèmes deviennent :

FND-SAT	Entrée:	$((\ell_{i,j})_{i\in[\![1,n_j]\!]})_{j\in[\![1,n]\!]}$ une famille de littéraux de Q
	Sortie:	oui si $A = \bigvee_{j=1}^{n} \bigwedge_{i=1}^{n_j} \ell_{i,j}$ est satisfiable, non sinon.

FNC-SAT	Entrée:	$((\ell_{i,j})_{i\in[\![1,n_j]\!]})_{j\in[\![1,n]\!]}$ une famille de littéraux de Q
	Sortie:	oui si $A = \bigwedge_{j=1}^{n} \bigvee_{i=1}^{n_j} \ell_{i,j}$ est satisfiable, non sinon.

Exemple: Prenons $(a\wedge b\wedge(\neg c))\vee((\neg b)\wedge(\neg c))\vee(a\wedge c)$. Mise sous forme d'ensemble d'ensembles de littéraux, la formule devient :

$$\{\{a, b, (\neg c)\}, \{(\neg b, \neg c)\}, \{a, c\}\}$$

Soit encore, sous forme d'ensemble de couples :

$$\{(\{a, b\}, \{c\}), (\varnothing, \{b, c\}), (\{a, c\}, \varnothing)\}$$

Exemple: Prenons $(a \wedge b \wedge (\neg c)) \vee (a \wedge c \wedge a) \vee ((\neg b) \wedge (\neg c))$. Mise sous forme d'ensemble d'ensembles de littéraux, la formule devient :

$$\{\{a, b, \neg c\}, \{(\neg b, \neg c)\}, \{a, c\}\}$$

Soit encore, sous forme d'ensemble de couples :

$$\{(\{a, b\}, \{c\}), (\varnothing, \{b, c\}), (\{a, c\}, \varnothing)\}$$

Exemple: Prenons $(a \wedge b \wedge (\neg a)) \vee (a \wedge c \wedge (\neg a)) \vee ((\neg b) \wedge (\neg c))$. Mise sous forme d'ensemble d'ensembles de littéraux, la formule devient :

$$\{\{a, b, \neg a\}, \{(a, \neg a, \neg c)\}, \{\neg b, \neg c\}\}$$

Soit encore, sous forme d'ensemble de couples :

$$\{(\{a, b\}, \{a\}), (\{a\}, \{a, c\}), (\varnothing, \{b, c\})\}$$

5 Algorithme pour FND-SAT

Pour j **allant de** 1 **à** n :
. Créer T un tableau indicé par $[\![1, N]\!]$ initialisé à -1
. **Essayer** :
. . **Pour** i **allant de** 1 **à** n_j :
. . . **Si** $\ell_{i,j} = q_k$:
. . . . **alors Si** $T[k] =$ -1, **alors** $T[k] \leftarrow 1$
. . . **Sinon si** $T[k] = 0$ **alors** déclencher "*conj. non sat.*"
. . . **Si** $\ell_{i,j} = \neg q_k$:
. . . . **alors Si** $T[k] =$ -1, **alors** $T[k] \leftarrow 0$
. . . **Sinon si** $T[k] = 1$ **alors** déclencher "*conj. non sat.*"
. . **Renvoyer** `Vrai`
. **Rattrappr** "*conj. non sat.*"
Renvoyer `Faux`

6 Modélisation

6.1 Exemple du solitaire

État inital du jeu : Sur chaque case d'un damier carré de $m \times m$ cases, il y a une pierre bleue (b), ou bien une pierre rouge (r) ou rien.

But du jeu : Enlever des pierres de manière à ce que :

- Sur chaque colonne toutes les pierres soient de la même couleur.
- Sur chaque ligne, il y ait au moins une pierre.

Exercice 3:

- Formaliser le problème de décision consistant à savoir si une partie peut être gagnée.
- Réduire ce problème à **3-SAT**.
- Réduire **3-SAT** à ce problème (bonus).

7 Algorithme de Quine

7.1 Substitution

Soit $\sigma \in \mathcal{F}(Q, \mathbb{F}_p(Q))$. On appelle **support de** σ, noté $\text{supp}(\sigma)$ l'ensemble des variables que σ n'envoie pas sur la formule réduite à elle-même.

$$\text{supp}(\sigma) = \{q \in Q \,|\, \sigma(q) \neq q\}$$

Exemple: Pour $\sigma = \begin{pmatrix} Q = \{x, y\} & \to & \mathbb{F}_p(\{x, y\}) \\ x & \mapsto & x \\ y & \mapsto & \neg x \end{pmatrix}$, $\text{supp}(\sigma) = \{y\}$.

Soit $\sigma \in \mathcal{F}(Q, \mathbb{F}_p(Q))$. On dit que σ est une **substitution** ssi $\text{card}\ (\text{supp}(\sigma)) \in \mathbb{N}$.

Soit $\sigma \in \mathcal{F}(Q, \mathbb{F}_p(Q))$ une substitution. On appelle **application de la substitution** σ la fonction suivante :

$$\bullet[\sigma] = \begin{pmatrix} \mathbb{F}_p(Q) & \to & \mathbb{F}_p(Q) \\ \bot & \mapsto & \bot \\ \top & \mapsto & \top \\ q & \mapsto & \sigma(q) \\ \neg A & \mapsto & \neg(A[\sigma]) \\ A\alpha B & \mapsto & A[\sigma]\alpha B[\sigma] \text{ où } \alpha \in \{\wedge, \vee, \to, \leftrightarrow\} \end{pmatrix}$$

Soit σ_1 et σ_2 deux substitutions. On note $\sigma_1 \cdot \sigma_2$ et on appelle **composée de** σ_1 **et** σ_2 la fonction

$$f = \begin{pmatrix} Q & \to & \mathbb{F}_p(Q) \\ q & \mapsto & \sigma_2(q)[\sigma_1] \end{pmatrix}$$

Propriété: *La composée de substitutions est* ***associative****, c'est à dire que pour tout triplet* $(\sigma_1, \sigma_2, \sigma_3) \in \mathcal{F}(Q, \mathbb{F}_p(Q))^3$ *vérifiant* $\forall i \in [\![1, 3]\!]$, *card* $(supp(\sigma_i)) \in \mathbb{N}$, *on a* $(\sigma_1 \cdot \sigma_2) \cdot \sigma_3 = \sigma_1 \cdot (\sigma_2 \cdot \sigma_3)$ *et admet pour*

▷ En exercice.

Remarque: Attention : une substition n'est pas forcément symétrisable ! Ainsi, l'ensemble des substitutions muni de la composition est juste un monoïde, mais pas un groupe.

7.2 Règles de simplification

On a les règles suivantes :

- $\neg\top \equiv \bot$
- $\neg\bot \equiv \top$
- $A \wedge \top \equiv \top \wedge A \equiv A$
- $A \wedge \bot \equiv \bot \wedge A \equiv \bot$
- $A \vee \top \equiv \top \vee A \equiv \top$
- $A \to \bot \equiv \neg A$
- $A \to \top \equiv A$
- $A \vee \bot \equiv \bot \vee A \equiv A$
- $\bot \to A \equiv T$
- $\top \to A \equiv A$
- $A \leftrightarrow \top \equiv A$
- $A \leftrightarrow \bot \equiv \neg A$

On remarque que l'application de chacune de ses règles amène à une réduction stricte de la taille de la formule, soit en enlevant un opérateur, soit en enlevant une variable, soit en enlevant un symbole $\bot$ ou $\top$.

Propriété: *Soit* $(A, B, A', B') \in \mathbb{F}_p(Q)^4$. *Pour* $\alpha \in \{\wedge, \vee, \to, \leftrightarrow\}$, *si* $A \equiv A'$ *et* $B \equiv B'$, *alors* $A\alpha B \equiv A'\alpha B'$.

▷ Il suffit de revenir à la définition de l'équivalence. Si $A \equiv A'$ alors pour tout environnement propositionnel ρ, $[A]^\rho = [A']^\rho$. Comme $[A\alpha B]^\rho$ s'écrit $f([A]^\rho, [B]^\rho) = f([A']^\rho, [B']^\rho)$, on en déduit que $[A\alpha B]^\rho = [A'\alpha B']^\rho$. Cela étant vrai pour tout ρ, on a bien $A\alpha B \equiv A'\alpha B'$.

Exercice 4: Simplifier les expressions suivantes :

- $((a \to b) \wedge (b \to c)) \to (a \to c)$
- $((s \to (b \vee t)) \wedge (((b \vee a) \to (r \wedge m)) \wedge (\neg r))) \to (s \to t)$