Chapitre 5. Logique.

1 Syntaxe

1.1 Formules logiques

On peut définir les formules logiques sur un ensemble $\mathscr{V}$ de manière récursive

Définition 1. *Soit $\mathscr{V}$ un ensemble fini ou dénombrable. Les* formules logiques *sans quantificateurs sur $\mathscr{V}$ sont définies de manière récursive par :*

— $\top$ *et* $\bot$ *sont des formules logiques (appelées* constantes *« Vrai » et « Faux »);*

— *les éléments de $\mathscr{V}$ sont des formules logiques (appelées* variables*);*

— *si F_1 et F_2 sont des formules logiques, alors :*
  - $(\neg F_1)$ *est une formule logique (appelée* négation *de F_1 et lue « non F_1 »);*
  - $(F_1 \vee F_2)$ *est une formule logique (appelée* disjonction *de F_1 et F_2 et lue « F_1 ou F_2 »);*
  - $(F_1 \wedge F_2)$ *est une formule logique (appelée* conjonction *de F_1 et F_2 et lue « F_1 et F_2 »).*

Remarques : – les formules logiques sont également appelées expressions logiques ou formules propositionnelles ou encore propositions logiques.
– Cette définition récursive des formules logiques permet de faire des raisonnements par induction structurelle.
Ainsi, une propriété sera vraie pour toute formule logique si elle est vraie pour les constantes, pour les variables et si lorsqu'elle est vraie pour deux formules logiques F_1 et F_2, elle est vraie pour $(\neg F_1)$, pour $(F_1 \vee F_2)$ et pour $(F_1 \wedge F_2)$.
– De même, pour définir une fonction sur l'ensemble des formules logiques sur $\mathscr{V}$, on procédera par induction structurelle.

Exemples : Voici quelques formules logiques sur $\mathscr{V} = \{v_1, v_2, v_3\}$: $(v_1 \wedge ((\neg v_2) \vee v_3)))$, $((\top \wedge v_1) \wedge (\neg v_3))$, $((v_1 \wedge v_2) \vee v_2)$.

Simplification
Par construction, toute formule logique admet une écriture parenthésée naturelle. Toutefois, pour éviter d'en écrire un trop grand nombre, on adopte les règles de priorité suivantes :

$\neg$ est prioritaire sur $\wedge$ qui est prioritaire sur $\vee$
dans une succession de $\wedge$ (resp. de $\wedge$), les priorités vont de gauche à droite

Exemples : – la formule $((v_1 \wedge v_2) \vee v_3)$ pourra s'écrire $v_1 \wedge v_2 \vee v_3$
– la formule $((((v_1 \wedge v_2) \wedge v_3) \wedge v_4) \wedge v_5)$ pourra s'écrire $v_1 \wedge v_2 \wedge v_3 \wedge v_4 \wedge v_5$
– la formule $((\neg v_1) \wedge v_2)$ pourra s'écrire $\neg v_1 \wedge v_2$
– en revanche les parenthèses dans la formule $\neg(v_1 \wedge v_2)$ ne peuvent pas être enlevées sous peine de changer la formule.

1.2 Représentation arborescente

La définition récursive des formules logiques permet d'associer à toute formule logique un arbre.

Définition 2. *La structure arborescente associée à une formule logique est définie de manière inductive par :*

- *l'arbre associé à une constante ou une variable est une feuille contenant la valeur de la constante ou de la variable*
- *l'arbre associé à $\neg F$ est un nœud $\neg$ ayant pour fils l'arbre associé à F*
- *l'arbre associé à $F_1 \wedge F_2$ (resp. à $F_1 \vee F_2$ est un nœud $\wedge$ (resp. $\vee$) ayant pour fils gauche l'arbre associé à F_1 et fils droit l'arbre associé à F_2.*

On définit la hauteur *et la* taille *d'une formule logique comme la hauteur et la taille de l'arbre associé à cette formule.*

Exemple : Représenter l'arbre associé à la formule (simplifiée) $(\neg v_1 \vee v_2) \wedge (v_1 \vee \neg v_3)$ et donner sa hauteur et sa taille

Définition 3 (Écriture de Lukasiewicz)**.**
On appelle écriture de Lukasiewicz *d'une formule logique , l'écriture obtenue par un parcours en profondeur suffixe de son arbre associé.*

Proposition 1. *L'écriture de Lukasiewicz d'une formule logique F détermine de manière unique la formule F.*

Démonstration : En effet, on a vu dans le cours de première année que le parcours en profondeur suffixe d'un arbre binaire complet où on distingue les feuilles des nœuds internes détermine de manière unique cet arbre. Or dans le cas des formules logiques, les feuilles (qui sont des constantes ou des variables) sont clairement identifiables des noeuds internes (étiquetés pour leur part par $\neg, \wedge, \vee$) $\sharp$

Exercice : a) Déterminer l'écriture de Lukasiewicz de la formule $(\neg v_1 \vee v_2) \wedge (v_1 \vee \neg v_3)$.
b) Réciproquement, déterminer la structure arborescente de la formule logique dont l'écriture de Lukasiewicz est $v_4 \neg \bot \vee v_2 \neg \wedge$.

2 Sémantique

2.1 Distribution de vérité et évaluation

Définition 4. *On appelle* distribution de vérité *sur un ensemble de variables $\mathscr{V}$ toute application de $\mathscr{V}$ dans $\{0, 1\}$.*

Remarque : Il existe donc 2^n distributions de vérité possibles sur un ensemble de variables de cardinal n.

Définition 5. *Soit μ une distribution de vérité sur un ensemble de variables $\mathscr{V}$. On appelle* évaluation associée à μ *et on note e_μ ou $[\mu]$ l'application définie sur l'ensemble des formules logiques sur $\mathscr{V}$ et à valeurs dans $\{0, 1\}$ définie par :*

- $\mathrm{e}_\mu(\top) = 1$, $\mathrm{e}_\mu(\bot) = 0$ *;*
- *pour toute variable $v \in \mathscr{V}$, $\mathrm{e}_\mu(v) = \mu(v)$;*
- *pour toute formule F, $\mathrm{e}_\mu(\neg F) = 1 - \mathrm{e}_\mu(F)$;*

— *pour toutes formules F_1, F_2,* $\mathrm{e}_\mu(F_1 \wedge F_2) = \min(e_\mu(F_1), e_\mu(F_2))$ *;*
— *pour toutes formules F_1, F_2,* $\mathrm{e}_\mu(F_1 \vee F_2) = \max(e_\mu(F_1), e_\mu(F_2))$ *;*

Exemple : Si $\mu(v_1) = 0$ et $\mu(v_2) = 1$, déterminer $e_\mu(\neg v_2 \vee v_2 \wedge \neg v_1)$

Définition 6. *La* table de vérité *d'une formule logique F sur $\mathscr{V}$ est le tableau dont les lignes sont indexées par les différentes distributions de vérité et qui contient les évaluations correspondantes de F*

Exemple : Voici les tables de vérité correspondant aux trois formules élémentaires sur deux variables v_1 et v_2

v_1	$\neg v_1$
0	1
1	0

v_1	v_2	$v_1 \wedge v_2$
0	0	0
0	1	0
1	0	0
1	1	1

v_1	v_2	$v_1 \vee v_2$
0	0	0
0	1	1
1	0	1
1	1	1

Exercice : Compléter la table de vérité associée à la formule $F = v_1 \vee v_2 \wedge \neg v_1$

v_1	v_2	$\neg v_1$	$v_2 \wedge \neg v_1$	F
0	0			
0	1			
1	0			
1	1			

Remarque : Il existe donc 2^{2^n} tables de vérités possibles pour les formules sur un ensemble $\mathscr{V}$ de cardinal n.

2.2 Équivalence sémantique

Définition 7. *Soit F_1 et F_2 deux formules logiques sur un même ensemble de variables $\mathscr{V}$.*

— *On dit que F_1 et F_2 sont sémantiquement équivalentes, et on note $F_1 \equiv F_2$, si elles ont la même table de vérité, c'est-à-dire si pour toute distribution de vérité μ, on a $e_\mu(F_1) = e_\mu(F_2)$*
— *On dit que F_1 prouve F_2 (ou que F_1 implique sémantiquement F_2), et on note $F_1 \models F_2$, si pour toute distribution de vérité μ, on a : $e_\mu(F_1) \leqslant e_\mu(F_2)$*

Exemples : Les tables de vérité établies ci-dessus permettent d'affirmer que :

$$v_1 \vee v_2 \wedge \neg v_1 \equiv v_1 \vee v_2.$$

Que peut-on dire de $F = \neg v_1 \wedge v_2$ et $F' = (v_1 \vee v_2) \wedge (\neg v_1 \vee \neg v_2)$?

Grâce à l'équivalence sémantique, on peut définir cinq nouveaux opérateurs logiques, en plus de $\neg, \vee, \wedge$.

Définition 8. *Si F_1 et F_2 sont deux formules logiques sur un ensemble de variables $\mathscr{V}$, on définit :*

— *le* « Non et », *noté F_1* `NAND` *F_2, sémantiquement équivalent à $\neg(F_1 \wedge F_2)$;*
— *le* « Non ou », *noté F_1* `NOR` *F_2, sémantiquement équivalent à $\neg(F_1 \vee F_2)$;*
— l'implication syntaxique*, notée $F_1 \Rightarrow F_2$, sémantiquement équivalente à $\neg(F_1) \vee F_2$;*

— l'équivalence syntaxique, *notée* $F_1 \Leftrightarrow F_2$*, sémantiquement équivalente à* $(F_1 \Rightarrow F_2) \wedge (F_2 \Rightarrow F_1)$ *;*
— *le* « Ou exclusif », *noté* F_1 `XOR` F_2 *ou* $F_1 \oplus F_2$*, sémantiquement équivalent à* $\neg(F_1 \Leftrightarrow F_2)$

Voici les tables de vérité de ces opérateurs :

v_1	v_2	v_1 `NAND` v_2	v_1 `NOR` v_2	$v_1 \Rightarrow v_2$	$v_1 \Leftrightarrow v_2$	v_1 `XOR` v_2
0	0	1	1	1	1	0
0	1	1	0	1	0	1
1	0	1	0	0	0	1
1	1	0	0	1	1	0

Remarque : Attention à ne pas confondre l'équivalence (ou l'implication) syntaxique et l'équivalence (ou l'implication) sémantique. En effet $F_1 \Leftrightarrow F_2$ est une formule logique, alors que $F_1 \equiv F_2$ est une propriété (qui est soit vraie soit fausse).

2.3 Régles de déduction naturelle

Les règles suivantes, sont régulièrement utilisées pour des raisonnements logiques ou mathématiques. Dans toute cette section, on note F_1, F_2, F_3 des formules logiques sur un même ensemble de variables $\mathscr{V}$.

Proposition 2 (Propriétés de $\vee$ et $\wedge$)**.**

Commutativité

$$F_1 \vee F_2 \equiv F_2 \vee F_1 \qquad F_1 \wedge F_2 \equiv F_2 \wedge F_1$$

Associativité

$$(F_1 \vee F_2) \vee F_3 \equiv F_1 \vee (F_2 \vee F_3) \qquad (F_1 \wedge F_2) \wedge F_3 \equiv F_1 \wedge (F_2 \wedge F_3)$$

Élément neutre

$$F_1 \vee \bot \equiv F_1 \qquad F_1 \wedge \top \equiv F_1$$

Élément absorbant

$$F_1 \vee \top \equiv \top \qquad F_1 \wedge \bot \equiv \bot$$

Idempotence

$$F_1 \vee F_1 \equiv F_1 \qquad F_1 \wedge F_1 \equiv F_1$$

Subsomption

$$F_1 \vee (F_1 \wedge F_2) \equiv F_1 \qquad F_1 \wedge (F_1 \vee F_2) \equiv F_1$$

Distributivité

$$F_1 \vee (F_2 \wedge F_3) \equiv (F_1 \vee F_2) \wedge (F_1 \vee F_3) \qquad F_1 \wedge (F_2 \vee F_3) \equiv (F_1 \wedge F_2) \vee (F_1 \wedge F_3)$$

Lois de Morgan

$$\neg(F_1 \vee F_2) \equiv \neg F_1 \wedge \neg F_2 \qquad \neg(F_1 \wedge F_2) \equiv \neg(F_1) \vee \neg F_2$$

Proposition 3 (Propriétés de la négation).

Négation de la négation

$$\neg(\neg F_1) \equiv F_1$$

Tiers exclu

$$F_1 \vee \neg F_1 \equiv \top$$

Non contradiction

$$F_1 \wedge \neg F_1 \equiv \bot$$

Proposition 4 (Raisonnement mathématique).

Symétrie

$$F_1 \Leftrightarrow F_2 \equiv F_2 \Leftrightarrow F_1$$

Transitivité

$$(F_1 \Rightarrow F_2) \wedge (F_2 \Rightarrow F_3) \models F_1 \Rightarrow F_3$$

Disjonction de cas

$$(F_1 \Rightarrow F_2) \wedge (\neg F_1 \Rightarrow F_2) \equiv F_2$$

Contraposition

$$F_1 \Rightarrow F_2 \equiv \neg F_2 \Rightarrow \neg F_1$$

Raisonnement par l'absurde

$$\neg F_1 \Rightarrow \bot \equiv F_1$$

3 Implémentation

Étant donné la structure inductive des formules logiques, on peut les définir en OCaml à l'aide du type personnalisé suivant :

```
type formule =
    | V
    | F
    | Var of int
    | Neg of formule
    | Ou of formule * formule
    | Et of formule * formule;;
```

Ici, la variable v_i est définie par `Var i`.
Si l'ensemble des variables est $\mathscr{V} = \{v_0, v_1, \dots v_{n-1}\}$, on pourra représenter une distribution de vérité sous la forme d'un tableau d'entiers de longueur n.

```
type distribution = int array;;
```

L'évaluation d'une formule pour une distribution de vérité donnée peut s'écrire :

```
let rec evaluation (mu : distribution) (f : formule) = match f with
    | V         -> 1
    | F         -> 0
    | Var i     -> mu.(i)
    | Neg g     -> 1 - evaluation mu g
    | Ou (g, h) -> max (evaluation mu g) (evaluation mu h)
    | Et (g, h) -> min (evaluation mu g) (evaluation mu h);;
```

On peut alors écrire une fonction testant l'équivalence sémantique de deux formules, en essayant les différentes distributions de vérité possibles, tant que les deux formules ont la même évaluation. Le troisième argument donne le nombre de variables qui sont indexées à partir de 0.

```
let equiv f g n =
    let mu = Array.make n 0 and b = ref true in
    let rec aux i =
        if not !b || i = n then
            b := !b && evaluation mu f = evaluation mu g
        else
            ( mu.(i) <- 0; aux (i+1); mu.(i) <- 1; aux (i+1) )
    in aux 0;
    !b;;
```

La fonction auxiliaire `aux` prend en argument un indice i et a pour but de tester toutes les distributions de vérité. Si i est un indice valide pour une variable, elle attribue à $\mu(v_i)$ la valeur 0 et lance un appel sur $i+1$ puis refait la même chose en attribuant cette fois à $\mu(v_i)$ la valeur 1. Si $i = n$, c'est qu'on a attribué une valeur à toutes les variables et qu'on peut donc tester l'égalité de l'évaluation des deux formules. Comme on teste les 2^n distributions de vérité, la complexité de la fonction précédente est exponentielle.

4 Formes normales conjonctives ou disjonctives

Pour manipuler une formule logique, il est souvent utile de se ramener à une formule sémantiquement équivalente et de syntaxe plus simple ou standardisée : c'est ce deuxième point que nous allons aborder maintenant.

4.1 Formules normales

Définition 9. *On appelle* conjonction *(respectivement* disjonction*) toute formule logique de la forme*

$$F_1 \wedge F_2 \wedge \ldots \wedge F_N \ \textit{(resp. } F_1 \vee F_2 \vee \ldots F_N\textit{)}$$

où $F_1, \ldots, F_N$ sont des formules logiques

Remarques : – Toute formule logique est à la fois une conjonction et une disjonction (cas où $N = 1$)
– Par convention $\bigwedge\limits_{i\in\emptyset} F_i \equiv \top$ et $\bigvee\limits_{i\in\emptyset} F_i \equiv \bot$

Définition 10.

- *On appelle* littéral *toute formule logique de la forme v ou $\neg v$ où v est une variable.*
- *On appelle* clause conjonctive *(resp.* disjonctive*) toute conjonction (resp. disjonction) de littéraux.*
- *On appelle* forme normale conjonctive *toute conjonction de clauses disjonctives et* forme normale disjonctive *toute disjonction de clauses conjonctives.*

Remarque : d'après les lois de Morgan et l'associativité de $\vee$ et de $\wedge$, la négation d'une clause conjonctive (resp. disjonctive) est équivalente à une clause disjonctive (resp. conjonctive). On en déduit que la négation d'une formule logique en forme normale conjonctive (resp. disjonctive) est équivalente à une formule en forme normale disjonctive (resp. conjonctive).

Théorème 1. *Toute formule logique est sémantiquement équivalente à une formule en forme normale conjonctive et à une formule en forme normale disjonctive*

Démonstration : On procède par induction structurelle.
Traitons tout d'abord les cas de base :

- $\perp$ est une disjonction (vide) de littéraux donc une clause disjonctive donc en forme normale conjonctive. On peut également écrire, pour tout $v \in \mathscr{V}$, $\perp = v \wedge \neg v$
  D'après la remarque ci-dessus $\top$ qui est équivalente à $\neg\perp$ est bien équivalente à une conjonction de disjonctions et à une disjonction de conjonctions.
- Si $v \in \mathscr{V}$, v est un littéral donc une clause conjonctive et disjonctive donc une forme normale disjonctive et conjonctive

Soient F_1 et F_2 deux formules sémantiquement équivalentes à des formules en forme normale conjonctive (C_1 et C_2) et à des formules en forme normale disjonctive (D_1 et D_2) Alors

- $\neg F_1$ est d'après la remarque ci-dessus sémantiquement équivalente à une forme normale disjonctive et à une forme normale conjonctive.
- $F_1 \wedge F_2 \equiv C_1 \wedge C_2$ qui est en forme normale conjonctive.
  De plus, si $D_1 = \bigvee\limits_{i \in I} \mu_i$ et $D_2 = \bigvee\limits_{j \in J} \nu_j$ où les μ_i et ν_j sont des clauses conjonctives, alors
  $$F_1 \wedge F_2 \equiv D_1 \wedge D_2 \equiv \bigvee_{(i,j) \in I \times J} (\mu_i \wedge \nu_j)$$
  qui est bien sous forme normale
- $F_1 \vee F_2 \equiv \neg(\neg F_1 \wedge \neg F_2)$ est bien équivalente à des formules de forme normale conjonctive et de forme normale disjonctive d'après les cas précédents $\sharp$

Remarque : On n'a bien sûr pas unicité d'une formule normale conjonctive (resp. disjonctive) équivalente à une formule donnée.

4.2 Formes canoniques

Définition 11.

- *On appelle* minterme *des variables $v_1, \ldots, v_n$ toute clause conjonctive de n littéraux où chacune des variables apparaît exactement une fois.*
- *On appelle* maxterme *des variables $v_1, \ldots, v_n$ toute clause disjonctive de n littéraux où chacune des variables apparaît exactement une fois.*

Exemple : Les maxtermes de v_1, v_2 sont $v_1 \vee v_2$, $\neg v_1 \vee v_2$, $v_1 \vee \neg v_2$ et $\neg v_1 \vee \neg v_2$.

Définition 12.

- *Une* forme normale conjonctive *est dite* canonique *si c'est une conjonction de maxtermes distincts de $\mathscr{V}$.*
- *Une* forme normale disjonctive *est dite* canonique *si c'est une disjonction de mintermes distincts de $\mathscr{V}$.*

Exemples : Sur $\mathscr{V} = \{v_1, v_2, v_3\}$, $(v_1 \vee v_2 \vee \neg v_3) \wedge (v_1 \vee \neg v_2 \vee \neg v_3)$ ...
$(v_1 \vee v_2) \wedge (\neg v_2 \vee v_3)$ ...

Théorème 2. *Toute formule logique est sémantiquement équivalente à une unique (à l'ordre près des clauses) formule sous forme normale conjonctive (resp. disjonctive) canonique.*

Démonstration : Une forme normale disjonctive canonique équivalente à une formule F donnée s'obtient très simplement à partir de sa table de vérité : en effet, si on considère les lignes de cette table de vérité pour lesquelles la formule est satisfaite, on montre que F est équivalente à la disjonction des formules caractérisant chacune de ces lignes.
Cela prouve le résultat tout en donnant une méthode pratique pour obtenir la forme normale disjonctive équivalente à F.
La forme normale conjonctive de F se déduit de la formule conjonctive normale équivalente à $\neg F$ grâce aux lois de Morgan $\sharp$

Remarque : le nombre de mintermes de la forme normale disjonctive de F est donc égal au nombre de distributions de vérité pour lesquelles la formule est évaluée à 1.

Exemple : Soit F une formule logique sur $\mathscr{V} = \{v_1, v_2, v_3\}$ dont la table de vérité est donnée par :

v_1	v_2	v_3	F
1	1	1	0
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	1

Déterminer la forme normale disjonctive canonique et la forme normale conjonctive canonique équivalentes à F.

Autre méthode
Pour obtenir la forme normale disjonctive canonique équivalente à une formule donnée, on peut procéder également de la manière suivante :

— on commence par exprimer la formule uniquement avec $\neg$, $\vee$ et $\wedge$;

— on « descend » le connecteur $\neg$ à l'aide des lois de Morgan ;

— on utilise la distributivité de $\wedge$ par rapport à $\vee$, autrement dit, on « descend » les connecteurs $\wedge$ par rapport aux connecteurs $\vee$;

— on utilise le principe du tiers exclu pour faire apparaître les variables absentes dans les clauses ;

— on utilise à nouveau la distributivité de $\wedge$ par rapport à $\vee$ si nécessaire.

Ici, « descendre » correspond à la descente dans la structure arborescente de la formule considérée.

Exemple : Déterminer la forme conjonctive normale canonique et la forme normale disjonctive canonique de la formule :

$$((v_1 \wedge v_2) \vee v_3) \Rightarrow (v_2 \Rightarrow v_1)$$

5 Tautologies, satisfiabilité

5.1 Définitions

Définition 13. *Soit F une formule logique sur l'ensemble de variables $\mathscr{V}$.*

- *On dit que F est une* tautologie*, si F est évaluée à* 1 *quelle que soit la distribution de vérité sur $\mathscr{V}$.*
- *On dit que F est* satisfiable *s'il existe une distribution de vérité μ sur $\mathscr{V}$ pour laquelle $e_\mu(F) = 1$.*
- *On dit que F est une* antilogie *si F n'est pas satisfiable.*

Remarques : Une tautologie est donc une formule logique dont la table de vérité ne contient que des 1 et une antilogie, une formule logique dont la table de vérité ne contient que des 0.

Exemples : Si F_1 est une formule, $F_1 \vee \neg F_1$ est une tautologie et $F_1 \wedge \neg F_1$ une antilogie. Si F_1, F_2, F_3 sont trois formules logiques sur $\mathscr{V}$, que dire de $(F_1 \Rightarrow F_2) \vee (F_2 \Rightarrow F_3)$?

Proposition 5 (Lien avec les implications et équivalences syntaxiques)**.**
Soit F une formule logique. Alors,

- *F est une tautologie si, et seulement si, $F \equiv \top$ si, et seulement si, $T \models F$;*
- *F est une antilogie si, et seulement si, $F \equiv \bot$ si, et seulement si, $F \models \bot$;*
- *F est satisfiable si, et seulement si $F \not\equiv \bot$.*

Proposition 6 (Lien entre implication et équivalence syntaxiques et sémantiques)**.**
Soit F_1 et F_2 deux formules logiques.

- *$F_1 \equiv F_2$ si, et seulement si, $F_1 \Leftrightarrow F_2$ est une tautologie ;*
- *$F_1 \models F_2$ si, et seulement si, $F_1 \Rightarrow F_2$ est une tautologie.*

5.2 Problèmes de satisfiabilité booléenne

Définition 14 (Problèmes SAT, k-SAT)**.**

- *Soit $k \in \mathbb{N}^*$. Une formule logique est dite en k-*FNC *si elle est en forme normale conjonctive et que chaque clause disjonctive comporte au plus k littéraux.*
- *On appelle problèmes* SAT *et k-*SAT *les problèmes suivants :*
  - SAT

    Donnée : Une formule logique F.

    Question : F est-elle satisfiable ?
  - k-SAT

    Donnée : Une formule logique F en k-FNC.

    Question : F est-elle satisfiable ?

Remarque : Le problème SAT a de nombreuses applications pratiques, car de nombreux problèmes de modélisation peuvent se ramener à la satisfiabilité d'une formule logique, et si possible à la détermination d'une distribution de vérité que la satisfait. Comme domaines d'applications, on peut citer le débogage, la preuve de logiciel, la génomique, l'intelligence artificielle, la cryptanalyse...

Proposition 7. *Le problème SAT peut se résoudre en temps exponentiel en la taille de la formule logique testée.*

Démonstration : Une formule de taille n comporte au plus n variables. Il suffit d'évaluer la formule pour toutes les distributions de vérité possibles. L'évaluation de la formule pour une distribution de vérité donnée se fait en temps linéaire en n d'après les règles de calcul. On en déduit une complexité totale en $O(n2^n)$ car il y a au plus 2^n distributions de vérité $\sharp$.

Remarque : A l'heure actuelle, le problème de savoir si SAT est résoluble en temps polynomial reste un problème ouvert.

Proposition 8. *Le problème* 1*-SAT peut se résoudre en temps linéaire en la taille de la formule testée.*

Démonstration : Une formule en 1-FNC est une clause conjonctive. Pour savoir si elle est satisfiable, il suffit de vérifier que la formule ne contient pas à la fois un littéral v et le littéral $\neg v$ pour une variable v ce qui peut se faire par un seul parcours de la formule $\sharp$

5.3 Problème 2-SAT

Nous nous proposons dans ce paragraphe de montrer le résultat suivant sous forme de mini-problème

Proposition 9. *Le problème* 2*-SAT peut être résolu en temps polynomial en la taille de la formule testée*

Les questions suivantes détaillent une preuve due à Aspvall, Plass et Tarjan.

Soit F une formule en 2-FNC sur $\mathscr{V} = \{v_0, v_1, \dots, v_{n-1}\}$.

1. Montrer qu'on peut se ramener au cas où chaque clause disjonctive contient exactement deux littéraux de variables distinctes.

On pose $G_F = (S, A)$ où $S = \mathscr{V} \cup \{\neg v_i \mid i \in [\![0, n-1]\!]\}$ et on identifie $\neg(\neg v_i)$ à v_i, chaque clause disjonctive $\ell_i \vee \ell_j$ de F donnant deux arêtes : $(\neg \ell_i, \ell_j)$ et $(\neg \ell_j, \ell_i)$.

2. Construire les graphes associés aux formules :

$$F_1 \ : \ (\neg v_1 \vee \neg v_2) \wedge (\neg v_0 \vee v_2) \wedge (v_1 \vee v_0) \wedge (\neg v_1 \vee v_2)\,;$$

$$F_2 \ : \ (\neg v_0 \vee v_3) \wedge (v_0 \vee \neg v_2) \wedge (v_1 \vee \neg v_2) \wedge (v_0 \vee v_2) \wedge (\neg v_1 \vee \neg v_0) \wedge (\neg v_3 \vee v_2).$$

3. Montrer que s'il existe une arête de ℓ_1 à ℓ_2 dans G_F, alors $F \models \ell_1 \Rightarrow \ell_2$.
4. Montrer qu'on a la même conclusion s'il existe un chemin de ℓ_1 à ℓ_2 dans G_F.
5. En déduire une condition nécessaire sur les composantes fortement connexes de G_F pour que F soit satisfiable.

Si G est un graphe orienté dont les composantes fortement connexes sont $C_1, \dots, C_k$, $k \in \mathbb{N}^*$. On définit le graphe des composantes fortement connexes de G, noté $\mathrm{CFC}(G)$ comme le graphe orienté $(\{C_1, \dots, C_k\}, A_{CFC})$ où pour tout $(i, j) \in [\![1, k]\!]^2$, il existe une arête de C_i vers C_j dans $\mathrm{CFC}(G)$ si, et seulement s'il existe deux sommets $x \in C_i$ et $y \in C_j$ tels que $(x, y) \in A$.

6. Montrer que si $(C_i, C_j) \in A_{CFC}$, alors pour tous $(x, y) \in C_i \times C_j$, il existe un chemin de x à y dans G.
7. En déduire que $\mathrm{CFC}(G)$ ne contient pas de cycle, puis qu'il existe une composante fortement connexe C qui est de degré sortant nul. Une telle composante connexe sera qualifiée de « puits ».
8. Dans cette question et la suivante, on suppose que dans G_F une variable et sa négation ne sont jamais dans la même composante connexe. Soit C un puits de $\mathrm{CFC}(G)$. Montrer qu'il existe une composante fortement connexe $\tilde{C}$ dont les littéraux sont les négations des littéraux de C, et que $\tilde{C}$ est une source (c'est-à-dire un sommet de degré entrant nul).

9. En déduire que F est satisfiable en proposant un algorithme récursif sur $\mathrm{CFC}(G)$ permettant de déterminer une distribution de vérité satisfaisant F.
10. Déterminer si F_1 et F_2 sont satisfiables et, le cas échéant, déterminer une distribution de vérité les satisfaisant.