

Logique propositionnelle

MP2I - Informatique

Anthony Lick

Lycée Janson de Sailly

Introduction

Exemple : logique propositionnelle

Trois personnes déjeunent ensemble à midi : A , B et C .

On sait que :

- si A prend un dessert, B aussi ;
- soit B , soit C prennent un dessert, mais pas les deux ;
- A ou C prend un dessert ;
- si C prend un dessert, A aussi.

Quelles sont les personnes qui ont pris un dessert ?

Syntaxe de la logique propositionnelle et représentation arborescente

Définition (logique propositionnelle)

Soit V un ensemble au plus dénombrable de variables logiques, qu'on notera $V = \{v_1, v_2, \dots, v_n, \dots\}$.

L'ensemble des expressions de la **logique propositionnelle** sur V est défini inductivement :

- \perp (lire "**bottom**") et \top (lire "**top**") sont des expressions logiques (aussi notées **Faux** et **Vrai**);
- v est une expression logique, pour tout $v \in V$;
- si ψ et φ sont deux expressions logiques, alors les expressions suivantes le sont aussi :
 - $(\psi \wedge \varphi)$ (lire "**et**", opération de **conjonction**);
 - $(\psi \vee \varphi)$ (lire "**ou**", opération de **disjonction**);
 - $(\neg\varphi)$ (lire "**non**", opération de **négation**);

Exemple

$\neg(v_1 \vee v_2)$ ou encore $(v_1 \vee \neg(v_2 \vee v_2))$ sont des expressions de la logique propositionnelle.

$\neg(v_1 \wedge v_2)$ et $(\neg v_1 \vee \neg v_2)$ sont deux expressions logiques différentes.

Arbre syntaxique

Les expressions logiques admettent naturellement une représentation arborescente : les variables logiques et les constantes \perp et \top sont les étiquettes des feuilles, les nœuds internes ayant pour étiquettes les **opérateurs**.

Un nœud d'étiquette \wedge ou \vee est d'arité 2 ; un nœud d'étiquette \neg est d'arité 1.

Cette représentation permet de définir la longueur et la hauteur d'une expression logique.

Représentation arborescente

Définition (hauteur et longueur)

On appelle :

- **hauteur** d'une expression logique la hauteur de son arbre syntaxique ;
- **longueur** d'une expression logique le nombre de nœuds de son arbre syntaxique.

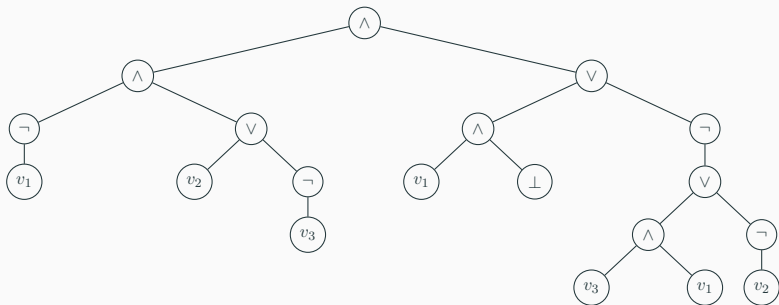
La hauteur et la longueur d'une expression logique peuvent également être définies par **induction**.

Définition (sous-formule)

Soit φ une formule logique.

On appelle **sous-formule** de φ toute formule correspondant à un sous-arbre de l'arbre syntaxique de φ .

Représentation arborescente



Exemple

Voici l'arbre syntaxique de l'expression suivante :

$$(\neg v_1 \wedge (v_2 \vee \neg v_3)) \wedge ((v_1 \wedge \perp) \vee \neg((v_3 \wedge v_1) \vee \neg v_2))$$

Elle est de hauteur 5 et de longueur 19.

Simplification

Pour raccourcir l'écriture d'une expression logique, on définit une priorité sur les opérateurs : \neg est prioritaire sur \wedge qui est lui-même prioritaire sur \vee .

Implémentation

```
1 type 'a prop =  
2   Bot | Top  
3   | V of 'a  
4   | Et of 'a prop * 'a prop  
5   | Ou of 'a prop * 'a prop  
6   | Non of 'a prop  
7 ;;
```

OCaml

On utilise naturellement un **type inductif** 'a prop pour implémenter les expressions de la **logique propositionnelle** en **OCaml**.

On utilise un **type polymorphe** pour pouvoir avoir des variables logiques indexées par des entiers, des chaînes de caractères, ou encore tout type à notre convenance.

Implémentation

```
1  let rec longueur e = match e with
2    | Bot | Top | V _ -> 1
3    | Et (g,d) -> 1 + longueur g + longueur d
4    | Ou (g,d) -> 1 + longueur g + longueur d
5    | Non g -> 1 + longueur g
6  ;;
7
8  let rec hauteur e = match e with
9    | Bot | Top | V _ -> 0
10   | Et (g,d) -> 1 + max (hauteur g) (hauteur d)
11   | Ou (g,d) -> 1 + max (hauteur g) (hauteur d)
12   | Non g -> 1 + hauteur g
13  ;;
```

Exemple

Voici deux fonctions permettant de calculer la hauteur et la longueur d'une expression logique.

Implémentation

Exemple

```
# Let e = Et
  (Et (Non (V 1), Ou (V 2, Non (V 3))),
   Ou (Et (V 1, Bot),
        Non (Ou (Et (V 3, V 1), Non (V 2)))))
;;
val e : int prop =
  Et (Et (Non (V 1), Ou (V 2, Non (V 3))),
      Ou (Et (V 1, Bot), Non (Ou (Et (V 3, V 1), Non (V 2)))))
# hauteur e ;;
- : int = 5
# longueur e ;;
- : int = 19
```

Exemple

Avec l'exemple précédent, on obtient bien une hauteur 5 et une longueur 19.

Sémantique de la logique propositionnelle

Définition (valuation)

Soit V un ensemble fini de variables.

Une **valuation** sur V est une application $V \rightarrow \{0, 1\}$.

Remarque

On utilise parfois F et V au lieu de 0 et 1.

Proposition

Si $|V| = n$, il y a 2^n valuations sur V .

Évaluation d'une expression logique

Définition (évaluation)

Soit ν une valuation sur V .

On peut étendre la définition de ν sur l'ensemble des formules logiques à variables dans V .

On définit ainsi $\nu(\varphi)$ par induction structurelle sur φ :

- $\nu(\perp) = 0$;
- $\nu(\top) = 1$;
- $\nu(v) = \nu(v)$, pour tout $v \in V$;
- $\nu(\varphi \wedge \psi) = \nu(\varphi) \times \nu(\psi)$;
- $\nu(\varphi \vee \psi) = \nu(\varphi) + \nu(\psi) - \nu(\varphi) \times \nu(\psi)$;
- $\nu(\neg\varphi) = 1 - \nu(\varphi)$.

Évaluation d'une expression logique

\wedge	0	1
0	0	0
1	0	1

\vee	0	1
0	0	1
1	1	1

Tables des opérateurs

Voici les **tables** des deux opérateurs \wedge et \vee sur $\{0, 1\}$.

Table de vérité d'une expression logique

Définition (table de vérité)

La **table de vérité** d'une expression logique φ sur V est la donnée des $\nu(\varphi)$ pour toute valuation ν sur V .

On représente en général la table de vérité comme un tableau, les lignes indexées par les valuations, et les colonnes par les éléments de V et φ .

Table de vérité d'une expression logique

Exemple

Voici la **table de vérité** de l'expression suivante, sur $V = \{v_1, v_2, v_3\}$:

$$\varphi = \neg v_1 \vee (v_2 \wedge \neg v_3)$$

v_1	v_2	v_3	φ
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table de vérité d'une expression logique

Définition (équivalence logique)

Deux expressions φ et ψ sur V sont dites **sémantiquement équivalentes** si elles ont la même table de vérité.

On note $\varphi \equiv \psi$ dans ce cas.

Table de vérité d'une expression logique

Proposition

À équivalence sémantique près, il y a (au plus) 2^{2^n} expressions logiques sur un ensemble de n variables logiques.

Preuve

Il y a 2^n valuations possibles, et deux valeurs possibles (0 ou 1) pour chacune d'entre elles, donc au plus 2^{2^n} tables de vérités associées à des expressions logiques.

Remarque

On verra dans la suite comment construire une expression logique associée à toute table de vérité. Il y a donc exactement 2^{2^n} expressions logiques à équivalence sémantique près.

Définition (sucre syntaxique)

Il est courant de rajouter des symboles dans notre syntaxe sans augmenter l'expressivité de nos expressions (i.e. les nouvelles expressions obtenues auront toujours un équivalent sémantique n'utilisant que les "anciens" symboles").

Soit φ et ψ deux formules logiques.

On définit le **sucre syntaxique** suivant :

- $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$ (**implication**) ;
- $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ (**équivalence**) ;
- $\varphi \uparrow \psi \equiv \neg(\varphi \wedge \psi)$ (prononcer "**nand**") ;
- $\varphi \downarrow \psi \equiv \neg(\varphi \vee \psi)$ (prononcer "**nor**") ;
- $\varphi \oplus \psi \equiv (\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi)$ (prononcer "**xor**").

Proposition (lois de De Morgan)

Soit φ et ψ deux formules logiques. On a :

- $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$;
- $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$.

Deux variables

Avec deux variables logiques v_1 et v_2 , il y a $16 = 2^{2^2}$ expressions logiques, à équivalence sémantique près.

Dénombrons-les, en fonction du nombre de 1 dans leur table de vérité.

- Sans 1 dans la table de vérité, il y a la formule logique \perp .
- Avec un seul 1, il y en a 4 :
 - $v_1 \wedge v_2$;
 - $\neg v_1 \wedge v_2$;
 - $v_1 \wedge \neg v_2$;
 - $\neg v_1 \wedge \neg v_2$.

Deux variables

Avec deux variables logiques v_1 et v_2 , il y a $16 = 2^{2^2}$ expressions logiques, à équivalence sémantique près.

Dénombrons-les, en fonction du nombre de 1 dans leur table de vérité.

- Avec deux 1, il y en a 6 (à équivalence sémantique près) :
 - v_1 ;
 - $\neg v_1$;
 - v_2 ;
 - $\neg v_2$;
 - $v_1 \leftrightarrow v_2$;
 - $v_1 \oplus v_2$.

Deux variables

Avec deux variables logiques v_1 et v_2 , il y a $16 = 2^{2^2}$ expressions logiques, à équivalence sémantique près.

Dénombrons-les, en fonction du nombre de 1 dans leur table de vérité.

- Avec trois 1, il y en a 4 :
 - $v_1 \vee v_2$;
 - $\neg v_1 \vee v_2$;
 - $v_1 \vee \neg v_2$;
 - $\neg v_1 \vee \neg v_2$.
- Avec quatre 1, il y a la formule logique \top .

Exemple

On reprend le problème de l'introduction, en notant x la variable logique indiquant que x prend un dessert.

Reformulons ces propositions en expressions logiques en les variables A , B et C :

- si A prend un dessert, B aussi : $A \rightarrow B$;
- Soit B , soit C prend un dessert, mais pas les deux : $B \oplus C$;
- A ou C prend un dessert : $A \vee C$;
- si C prend un dessert, A aussi : $C \rightarrow A$.

Retour sur le problème de l'introduction

A	B	C	$A \rightarrow B$	$B \oplus C$	$A \vee C$	$C \rightarrow A$
0	0	0	1	0	0	1
0	0	1	1	1	1	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Exemple

Dressons la table de vérité de ces 4 formules logiques avec $V = \{A, B, C\}$.

Retour sur le problème de l'introduction

A	B	C	$A \rightarrow B$	$B \oplus C$	$A \vee C$	$C \rightarrow A$
0	0	0	1	0	0	1
0	0	1	1	1	1	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Exemple

La ligne en orange correspond à la seule valuation pour laquelle l'évaluation des quatre formules logiques donne 1.

Retour sur le problème de l'introduction

A	B	C	$A \rightarrow B$	$B \oplus C$	$A \vee C$	$C \rightarrow A$
0	0	0	1	0	0	1
0	0	1	1	1	1	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	1	1
1	0	1	0	1	1	1
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Exemple

On en déduit que A et B prennent un dessert, et pas C .

Tautologies, antilogies, et formules satisfiables

Tautologies, antilogies, et formule satisfiables

Définition (tautologie, antilogie, formule satisfiable)

Soit φ une formule de la logique propositionnelle sur un ensemble de variables V . On dit que :

- φ est une **tautologie** si pour toute valuation ν , $\nu(\varphi) = 1$;
- φ est une **antilogie** si pour toute valuation ν , $\nu(\varphi) = 0$;
- φ est **satisfiable** s'il existe une valuation ν telle que $\nu(\varphi) = 1$.

Dans ce dernier cas, on dit que φ est **satisfaite** par ν , ou encore que ν est un **modèle** de φ , et on note $\nu \models \varphi$.

Tautologies, antilogies, et formule satisfiables

Remarques

- On dit aussi formule **valide** pour une tautologie, ou formule **fausse** pour une antilogie.
- Une antilogie est donc une formule qui n'a pas de modèle.
- Une formule valide est une donc une formule pour laquelle toute valuation est un modèle.
- φ est une tautologie $\iff \varphi \equiv \top$.
- φ est une antilogie $\iff \varphi \equiv \perp$.

Ces trois définitions sont en fait reliées de la manière suivante :

- φ est une tautologie $\iff \neg\varphi$ est une antilogie ;
- φ est une antilogie $\iff \varphi$ n'est pas satisfiable.

Table de vérité d'une expression logique

Définition (conséquence logique)

Soit φ et ψ deux formules logiques.

On dit que φ est une **conséquence logique** de ψ si, pour toute valuation ν , $\nu \models \psi \Rightarrow \nu \models \varphi$.

On note $\psi \models \varphi$ dans ce cas.

Définition (conséquence logique)

Soit φ une formule logique, et Γ un ensemble de formules.

On dit que φ est une **conséquence logique** de Γ si, pour toute valuation ν , pour toute formule $\psi \in \Gamma$, $\nu \models \psi \Rightarrow \nu \models \varphi$.

On note $\Gamma \models \varphi$ dans ce cas.

Remarques

- $\psi \vDash \varphi$ revient à dire que $\psi \rightarrow \varphi$ est une **tautologie**.
- $\Gamma \vDash \varphi$ revient à dire que $(\bigwedge_{\psi \in \Gamma} \psi) \rightarrow \varphi$ est une **tautologie**.
- Si φ est **valide**, elle est la conséquence logique d'un ensemble vide de formules. C'est pourquoi on note souvent $\vDash \varphi$ pour dire que φ est valide.

Tautologies

Il est utile de posséder une liste de tautologies, bien qu'il soit impossible de "toutes" les lister, car elles sont en nombre infini.

Donnons-en quelques une, mais commençons d'abord par une proposition.

Opération de substitution

Définition (substitution)

Soit φ et ψ deux formules sur un ensemble de variables propositionnelles V , soit $v \in V$.

La **substitution** de v par ψ dans φ , notée $\varphi[\psi/v]$, est la formule obtenue en remplaçant toutes les occurrences de v dans φ par ψ . On peut définir $\varphi[\psi/v]$ par induction :

- $\top[\psi/v] = \top$ et $\perp[\psi/v] = \perp$;
- $v[\psi/v] = \psi$;
- $v'[\psi/v] = v'$ pour $v' \neq v$;
- $(\neg\varphi)[\psi/v] = \neg(\varphi[\psi/v])$;
- $(\varphi_1 \wedge \varphi_2)[\psi/v] = (\varphi_1[\psi/v]) \wedge (\varphi_2[\psi/v])$;
- $(\varphi_1 \vee \varphi_2)[\psi/v] = (\varphi_1[\psi/v]) \vee (\varphi_2[\psi/v])$.

Opération de substitution

Notation

Si l'on veut effectuer plusieurs substitutions à la fois, on note $\varphi[\psi_1/v_1, \dots, \psi_n/v_n]$.

Les tautologies : la base du raisonnement mathématique

Proposition

Si φ est une tautologie en les variables v_1, \dots, v_n , et ψ_1, \dots, ψ_n des formules logiques sur un ensemble de variables W , alors $\varphi[\psi_1/v_1, \dots, \psi_n/v_n]$ est une tautologie en les variables de W .

Preuve

Soit w_1, \dots, w_p les variables de W intervenant dans les ψ_i .

Prenons une valuation ν sur $\{w_1, \dots, w_p\}$.

Notons ν' la valuation sur $\{v_1, \dots, v_n\}$ telle que $\nu'(v_i) = \nu(\psi_i)$.

On a $\nu(\varphi[\psi_1/v_1, \dots, \psi_n/v_n]) = \nu'(\varphi) = 1$ car φ est une tautologie. Ainsi, pour tout ν , on a $\nu \models \varphi[\psi_1/v_1, \dots, \psi_n/v_n]$, donc $\varphi[\psi_1/v_1, \dots, \psi_n/v_n]$ est une tautologie.

Les tautologies : la base du raisonnement mathématique

Remarque

Il en va de même pour une antilogie, mais pas pour une formule satisfiable.

Les tautologies : la base du raisonnement mathématique

Exemple

Voici une liste de tautologies célèbres.

La proposition précédente nous indique que l'on peut remplacer les variables par n'importe quelle formule, et obtenir une formule valide.

- négation de la négation : $\neg\neg v \leftrightarrow v$;
- équivalence à vrai/faux : $(v \leftrightarrow \top) \leftrightarrow v$ et $(v \leftrightarrow \perp) \leftrightarrow \neg v$;
- idempotence : $(v \wedge v) \leftrightarrow v$ et $(v \vee v) \leftrightarrow v$;
- implication et équivalence d'une même variable :
 $(v \rightarrow v) \leftrightarrow \top$ et $(v \leftrightarrow v) \leftrightarrow \top$;
- commutativité : $v_1 \wedge v_2 \leftrightarrow v_2 \wedge v_1$, et de même avec tous les autres opérateurs binaires sauf \rightarrow .
- équivalence des négations : $(\neg v_1 \leftrightarrow \neg v_2) \leftrightarrow (v_1 \leftrightarrow v_2)$;

Les tautologies : la base du raisonnement mathématique

Exemple

Voici une liste de tautologies célèbres.

La proposition précédente nous indique que l'on peut remplacer les variables par n'importe quelle formule, et obtenir une formule valide.

- lois de De Morgan : $\neg(v_1 \wedge v_2) \leftrightarrow \neg v_1 \vee \neg v_2$
et $\neg(v_1 \vee v_2) \leftrightarrow \neg v_1 \wedge \neg v_2$;
- associativité : $(v_1 \vee (v_2 \vee v_3)) \leftrightarrow ((v_1 \vee v_2) \vee v_3)$, et de même avec \wedge , \oplus , et \leftrightarrow ;
- distributivité de \vee sur \wedge :
 $(v_1 \vee (v_2 \wedge v_3)) \leftrightarrow (v_1 \vee v_2) \wedge (v_1 \vee v_3)$;
- distributivité de \wedge sur \vee :
 $(v_1 \wedge (v_2 \vee v_3)) \leftrightarrow (v_1 \wedge v_2) \vee (v_1 \wedge v_3)$;

Exemple

Voici une liste de tautologies célèbres.

La proposition précédente nous indique que l'on peut remplacer les variables par n'importe quelle formule, et obtenir une formule valide.

- transitivité : $((v_1 \rightarrow v_2) \wedge (v_2 \rightarrow v_3)) \rightarrow (v_1 \rightarrow v_3)$;
- simplification :
 - $(v_1 \vee (v_1 \wedge v_2)) \leftrightarrow v_1$;
 - $(v_1 \vee (\neg v_1 \wedge v_2)) \leftrightarrow v_1 \vee v_2$;
 - $(v_1 \wedge (v_1 \vee v_2)) \leftrightarrow v_1$;
 - $(v_1 \wedge (\neg v_1 \vee v_2)) \leftrightarrow v_1 \wedge v_2$;

Exemple

Voici les bases du raisonnement mathématique :

- tiers exclus : $(v \vee \neg v) \leftrightarrow \top$, et $(v \wedge \neg v) \leftrightarrow \perp$;
- contraposée : $(v_1 \rightarrow v_2) \leftrightarrow (\neg v_2 \rightarrow \neg v_1)$;
- double implication : $((v_1 \rightarrow v_2) \wedge (v_2 \rightarrow v_1)) \leftrightarrow (v_1 \leftrightarrow v_2)$;
- disjonction de cas : $((v_1 \rightarrow v_2) \wedge (\neg v_1 \rightarrow v_2)) \rightarrow v_2$;
- démonstration par l'absurde : $(\neg v_1 \rightarrow \perp) \rightarrow v_1$;
- modus ponens : $(v_1 \wedge (v_1 \rightarrow v_2)) \rightarrow v_2$.

Formes normales

Forme normale

Soit φ une formule logique.

On cherche à trouver une formule sémantiquement équivalente à φ de la forme la plus “simple” possible.

Formes normales conjonctives et disjonctives

Définition (littéral)

Soit V un ensemble de variables logiques. On appelle **littéral** de V une expression logique de la forme v ou $\neg v$, pour $v \in V$.

Définition (forme normale disjonctive)

On dit qu'une formule est en **forme normale disjonctive** si c'est une disjonction de conjonctions de littéraux.

Exemple

Avec $V = \{v_1, v_2, v_3\}$, la formule logique :

$$\varphi = (v_1 \wedge v_2 \wedge \neg v_3) \vee (\neg v_1 \wedge v_2)$$

est sous **forme normale disjonctive** : c'est une disjonction de deux conjonctions de littéraux.

Formes normales conjonctives et disjonctives

Définition (clause)

On appelle **clause** une disjonction de littéraux.

Définition (forme normale conjonctive)

On dit qu'une formule est en **forme normale conjonctive** si c'est une conjonction de clauses (i.e. une conjonction de disjonctions de littéraux).

Exemple

$$\varphi = v_1 \wedge (\neg v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \neg v_2)$$

est sous **forme normale conjonctive**, composée de trois clauses à 1, 2 et 3 littéraux.

Formes normales conjonctives et disjonctives

Remarque

En général, on suppose que les variables apparaissant dans une clause ou une conjonction de littéraux sont toutes différentes.

En effet :

- d'une part on peut utiliser l'idempotence des opérateurs \wedge et \vee pour supprimer les littéraux égaux ;
- et d'autre part :
 - une conjonction de littéraux comportant v et $\neg v$ est sémantiquement équivalente à \perp ;
 - une clause comportant v et $\neg v$ est sémantiquement équivalente à \top .

Remarque

\perp est le neutre pour \vee , et \top est le neutre pour \wedge .

On considère donc \perp comme une disjonction et \top comme une conjonction.

Forme canonique

Sans hypothèses supplémentaires, il existe plusieurs (une infinité, en fait) formes normales disjonctives ou conjonctives équivalentes à une formule logique donnée.

Les formes normales sont en pratique fournies en entrée des algorithmes décidant si une formule logique est satisfiable (resp. une tautologie ou une antilogie) : il faut donc être capable de mettre une formule sous forme normale.

Avant de décrire une algorithme de mise sous forme normale conjonctive ou disjonctive, prouvons l'existence d'une telle écriture.

Théorème

Soit φ une formule logique sur un ensemble de variables $V = \{v_1, \dots, v_n\}$.

Alors φ est **sémantiquement équivalente** à une **disjonction de conjonctions de littéraux**, et à une **conjonction de clauses**.

Mise sous forme normale

Lemme

La négation d'une conjonction de disjonctions est une disjonction de conjonctions, et réciproquement.

Preuve

Il suffit d'appliquer les lois de De Morgan.

Preuve du théorème

Montrons le théorème par induction structurelle sur φ .

- \perp est une disjonction (sans conjonction) ; c'est également une conjonction (d'une seule disjonction vide) ;
- \top est une conjonction (sans disjonction) ; c'est également une disjonction (d'une seule conjonction vide) ;
- Pour $v \in V$, v et $\neg v$ sont des conjonctions d'une conjonction à un littéral, ainsi que des disjonctions de conjonctions à un littéral ;
- Si φ admet de telles écritures, c'est également le cas pour $\neg\varphi$ d'après le lemme précédent ;

Mise sous forme normale

Preuve du théorème

- Si $\varphi = \varphi_1 \wedge \varphi_2$:
 - par hypothèse d'induction, $\varphi_i \equiv \tilde{\varphi}_i$ avec $\tilde{\varphi}_i$ sous forme normale conjonctive. Ainsi, $\varphi \equiv \tilde{\varphi} = \tilde{\varphi}_1 \wedge \tilde{\varphi}_2$, et $\tilde{\varphi}$ est bien sous forme normale conjonctive.
 - par hypothèse d'induction, φ_i admet également une écriture en forme normale disjonctive, donc $\varphi_i \equiv c_1^i \vee \dots \vee c_{n_i}^i$, avec les c_k^i des conjonctions de littéraux. Alors, en utilisant la distributivité de \wedge sur \vee , on obtient :

$$\varphi \equiv \left(\bigvee_{1 \leq i \leq n_1} c_i^1 \right) \wedge \left(\bigvee_{1 \leq j \leq n_2} c_j^2 \right) \equiv \bigvee_{\substack{1 \leq i \leq n_1 \\ 1 \leq j \leq n_2}} c_i^1 \wedge c_j^2$$

Preuve du théorème

- On peut éliminer les littéraux en doublons dans chaque conjonction $c_i^1 \wedge c_j^2$ obtenue, et éliminer totalement les conjonctions contenant une variable et sa négation.
- le cas $\varphi = \varphi_1 \vee \varphi_2$ est analogue, en échangeant les cas des conjonctions et des disjonctions, et les \wedge et les \vee .

Mise sous forme normale

Algorithme

La démonstration du théorème précédent fournit une méthode pour calculer une forme normale, (conjonctive ou disjonctive). Il suffit :

- de supprimer les \perp et les \top ;
- de reformuler les \rightarrow , \oplus , etc à l'aide de \wedge , \vee , et \neg ;
- d'utiliser les lois de De Morgan pour faire descendre les \neg dans les littéraux (au niveau des feuilles de l'arbre syntaxique) ;
- d'utiliser la distributivité de \vee sur \wedge et celle de \wedge sur \vee ;
- de simplifier en éliminant les doublons de littéraux dans une clause/conjonction de littéraux, et en utilisant les relations $v \wedge \neg v \equiv \perp$ et $v \vee \neg v \equiv \top$.

Exemple

Mettons le problème de l'introduction sous **forme normale conjonctive** :

$$\varphi = (A \rightarrow B) \wedge (B \oplus C) \wedge (A \vee C) \wedge (C \rightarrow A)$$

On reformule d'abord les opérateurs autre que \neg , \wedge et \vee :

- $A \rightarrow B \equiv \neg A \vee B$;
- $B \oplus C \equiv (B \vee C) \wedge \neg(B \wedge C)$;
- $C \rightarrow A \equiv \neg C \vee A$.

Mise sous forme normale

Exemple

La formule obtenue est quasiment sous **forme normale conjonctive**, il ne reste plus qu'à appliquer les lois de De Morgan sur la formule issue de $B \oplus C$:

$$\begin{aligned} B \oplus C &\equiv (B \vee C) \wedge \neg(B \wedge C) \\ &\equiv (B \vee C) \wedge (\neg B \vee \neg C) \end{aligned}$$

Ainsi :

$$\varphi \equiv (\neg A \vee B) \wedge (B \vee C) \wedge (\neg B \vee \neg C) \wedge (A \vee C) \wedge (\neg C \vee A)$$

La formule obtenue est bien sous **forme normale conjonctive** : il y a 5 clauses à 2 littéraux.

Mise sous forme normale

Exemple

$$\varphi \equiv (\neg A \vee B) \wedge (B \vee C) \wedge (\neg B \vee \neg C) \wedge (A \vee C) \wedge (\neg C \vee A)$$

Repartons de cette formule pour obtenir une **forme normale disjonctive**. On utilise maintenant la distributivité de \wedge sur \vee :

$$\begin{aligned}(A \vee C) \wedge (\neg C \vee A) &\equiv (A \wedge \neg C) \vee (A \wedge A) \vee (C \wedge \neg C) \vee (A \wedge C) \\ &\equiv A \vee (A \wedge \neg C) \vee (A \wedge C) \text{ car } A \wedge A \equiv A \\ &\hspace{15em} \text{et } C \wedge \neg C \equiv \perp \\ &\equiv A \quad (\text{r\`egle de simplification})\end{aligned}$$

$$(\neg A \vee B) \wedge A \equiv A \wedge B \quad (\text{r\`egle de simplification})$$

Exemple

Finalement, en utilisant les règles de simplification :

$$\begin{aligned}\varphi &\equiv A \wedge B \wedge (B \vee C) \wedge (\neg B \vee \neg C) \\ &\equiv A \wedge B \wedge (\neg B \vee \neg C) \\ &\equiv A \wedge B \wedge \neg C\end{aligned}$$

Cette dernière expression est à la fois sous **forme normale disjonctive** (une seule conjonctive de 3 littéraux), et sous **forme normale conjonctive** (conjonction de 3 clauses comportant un unique littéral).

Mise sous forme normale

Remarque

On remarque via l'exemple précédent qu'une forme normale n'est pas unique (on a montré deux formes normales conjonctives équivalentes à la formule initiale).

De plus, il n'est pas toujours facile d'obtenir une forme normale, car celle-ci peut être de taille exponentielle en la formule initiale.

Mise sous forme normale

Exemple

Soit $V = \{x_1, \dots, x_n, y_1, \dots, y_n\}$.

L'expression suivante est sous **forme normale disjonctive**, de taille $O(n)$:

$$\varphi = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \dots \vee (x_n \wedge y_n)$$

On peut montrer qu'une **forme normale conjonctive** équivalente est :

$$\varphi \equiv \bigwedge_{\substack{z_i \in \{x_i, y_i\} \\ 1 \leq i \leq n}} (z_1 \vee z_2 \vee \dots \vee z_n)$$

Cette formule comporte 2^n clauses, et on peut montrer qu'on ne peut pas trouver plus petit.

Remarque

Il est courant d'alléger les expressions pour utiliser des formes arithmétiques plus proches de celles qu'on utilise habituellement, ce qui a l'avantage de simplifier les écritures lorsqu'on travaille à équivalence sémantique près.

Ainsi :

- on remplace \perp par 0 et \top par 1 ;
- on remplace \wedge par \cdot (voire rien du tout) ;
- on remplace \vee par $+$;
- on écrit \bar{v} à la place de $\neg v$;
- on s'autorise à écrire $=$ à la place de \equiv .

Mise sous forme normale

Remarque

Cette forme rend plus intuitive la distributivité de \wedge sur \vee . Il ne faut cependant pas oublier l'autre distributivité, l'absorbance de 1 pour \vee , l'idempotence, le fait que $a + \bar{a} = 1$, $a\bar{a} = 0$, et les autres règles de simplification.

Exemple

$$\begin{aligned}(a + b)(a + \bar{b} + c) &= a^2 + ab + a\bar{b} + b\bar{b} + ac + bc \\ &= a(1 + b + \bar{b} + c) + bc \\ &= a + bc\end{aligned}$$

Formes normales conjonctives et disjonctives canoniques

Forme canonique

Il est possible d'assurer l'**unicité** d'une forme normale conjonctive ou disjonctive, en rajoutant une condition sur les conjonctions et disjonctions de littéraux : celles-ci doivent être de taille **maximale**.

Commençons par les **formes normales disjonctives canoniques**.

Formes normales disjonctives canoniques

Définition (min-terme)

Soit $V = \{v_1, \dots, v_n\}$ un ensemble fini de variables logiques.

Un **min-terme** sur V est une conjonction de n littéraux dans laquelle chaque variable apparaît exactement une fois.

Exemple

Avec $n = 3$:

- $\neg v_1 \wedge v_2 \wedge \neg v_3$ est un min-terme ;
- $v_1 \wedge \neg v_2$ et $v_1 \wedge \neg v_1 \wedge v_2$ n'en sont pas.

Formes normales disjonctives canoniques

Proposition

La table de vérité d'un min-terme contient un seul 1.

Preuve

En effet, pour m un min-terme, le seul modèle ν tel que $\nu \models m$ est celui donné par la valuation suivante :

$$\nu(v_i) = \begin{cases} 1 & \text{si } v_i \text{ apparait dans } m \\ 0 & \text{si } \neg v_i \text{ apparait dans } m \end{cases}$$

Théorème (forme normale disjonctive canonique)

Soit φ une formule logique sur un ensemble fini de variables V . Alors φ est sémantiquement équivalente à une **unique** (à l'ordre près) disjonction de min-termes différents (deux min-termes sont considérés comme différents si les littéraux qui apparaissent dans la conjonction diffèrent).

Formes normales disjonctives canoniques

Preuve

Pour l'existence, on vérifie que :

$$\varphi \equiv \bigvee_{\nu \mid \nu \models \varphi} m_\nu$$

où m_ν est le min-terme associé à la valuation ν , défini par $m_\nu = l_1 \wedge l_2 \wedge \dots \wedge l_n$, avec :

$$l_i = \begin{cases} v_i & \text{si } \nu(v_i) = 1 \\ \neg v_i & \text{si } \nu(v_i) = 0 \end{cases}$$

Formes normales disjonctives canoniques

Preuve

Pour l'unicité, il suffit de voir que si ν est une valuation telle que $\nu \models \varphi$, alors m_ν est le seul min-terme tel que $\nu \models m_\nu$, il doit donc apparaître dans la décomposition.

Réciproquement, si $\nu \not\models \varphi$, alors m_ν ne peut apparaître dans l'écriture.

Formes normales disjonctives canoniques

Définition (forme normale disjonction canonique)

$$\varphi \equiv \bigvee_{\nu \mid \nu \models \varphi} m_\nu$$

La formule logique de la preuve précédente est appelée la **forme normale disjonctive canonique** (abrégée en **FNDC**) sémantiquement équivalente à φ .

Construction

On peut paraphraser la preuve précédente en disant que pour trouver la **FNDC** équivalente à φ , il suffit de regarder la table de vérité de φ : chaque ligne avec un 1 fournit un min-terme apparaissant dans la décomposition.

Exemple

$$\varphi = (A \rightarrow B) \wedge (B \oplus C) \wedge (A \vee C) \wedge (C \rightarrow A)$$

$\varphi \equiv A \wedge B \wedge \neg C$ est la **FNDC** de la formule ci-dessus.
Elle n'est composée que d'un seul min-terme.

Formes normales disjonctives canoniques

Exemple

Considérons $\varphi = (v_1 \rightarrow \neg v_2) \wedge (v_1 \vee v_3)$.

Dressons sa table de vérité :

v_1	v_2	v_3	$v_1 \rightarrow \neg v_2$	$v_1 \vee v_3$	φ
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	0

Ainsi :

$$\varphi \equiv (\neg v_1 \wedge \neg v_2 \wedge v_3) \vee (\neg v_1 \wedge v_2 \wedge v_3) \vee (v_1 \wedge \neg v_2 \wedge \neg v_3) \vee (v_1 \wedge \neg v_2 \wedge v_3)$$

Formes normales disjonctives canoniques

Remarque

Cette écriture en **FNDC** permet de montrer qu'il y a bien 2^{2^n} formules logiques sur un ensemble de n variables, à équivalence sémantique près.

Lien avec la partie précédente

La négation établit une **dualité** entre les **formes normales disjonctives** et les **formes normales conjonctives**.

Ce qui suit est donc très proche de ce qu'on vient de voir sur les **FNDC**.

Formes normales conjonctives canoniques

Définition (max-terme)

Un **max-terme** sur un ensemble de n variables est une disjonction de n littéraux dans laquelle chaque variable apparaît exactement une fois.

Proposition

La négation d'un min-terme est un max-terme, et réciproquement.

Preuve

C'est une simple application de la loi de De Morgan.

Formes normales conjonctives canoniques

Proposition

La table de vérité d'un max-terme ne contient qu'un seul 0.

Preuve

Immédiat, par négation.

Formes normales conjonctives canoniques

Théorème (forme normale conjonctive canonique)

Une formule logique sur un ensemble fini de variables V est sémantiquement équivalente à une **unique** conjonction de **max-termes** différents.

Définition (forme normale conjonctive canonique)

Cette écriture se nomme la **forme normale conjonctive canonique** (abrégée en **FNCC**).

Formes normales conjonctives canoniques

Preuve

Soit φ une formule logique sur V .

Considérons la forme normale canonique de $\neg\varphi$:

$$\neg\varphi \equiv \bigvee_{\nu \mid \nu \models \neg\varphi} m_\nu$$

Ainsi, par négation :

$$\varphi \equiv \bigwedge_{\nu \mid \nu \not\models \varphi} (\neg m_\nu)$$

Et chacun des $\neg m_\nu$ est un max-terme.

Formes normales conjonctives canoniques

Construction

Pour trouver la **FNCC** équivalente à φ , il suffit de regarder la table de vérité : chaque ligne avec un 0 fournit un max-terme $l_1 \wedge l_2 \wedge \dots \wedge l_n$ apparaissant dans la décomposition, avec :

$$l_i = \begin{cases} \neg v_i & \text{si } \nu(v_i) = 1 \\ v_i & \text{si } \nu(v_i) = 0 \end{cases}$$

Exemple

$$\varphi = (A \rightarrow B) \wedge (B \oplus C) \wedge (A \vee C) \wedge (C \rightarrow A)$$

La **FNCC** de la formule ci-dessus est composée de 7 max-terms : tous sauf $\neg A \vee \neg B \vee C$.

Formes normales conjonctives canoniques

Exemple

La **FNCC** de $\varphi = (v_1 \rightarrow \neg v_2) \wedge (v_1 \vee v_3)$ s'obtient en regardant les 0 dans la table de vérité :

v_1	v_2	v_3	$v_1 \rightarrow \neg v_2$	$v_1 \vee v_3$	φ
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	0

$$\varphi \equiv (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3)$$

Problème de décision

À partir d'une **forme normale canonique** (conjonctive ou disjonctive), il est facile de décider si une formule logique est satisfiable / valide / insatisfiable.

Proposition

Soit φ une formule logique en un ensemble fini de variables.
Alors :

- φ tautologie \iff sa **FNDC** contient tous les min-termes
 \iff sa **FNCC** ne contient aucun max-terme ;
- φ antilogie \iff sa **FNDC** ne contient aucun min-terme
 \iff sa **FNCC** contient tous les max-termes ;
- φ satisfiable \iff sa **FNDC** contient au moins un min-terme
 \iff sa **FNCC** ne contient pas tous les max-termes.

Application aux problèmes de décision

Preuve

Cela provient du fait que les min-termes sont liés aux 1 dans la table de vérité, et les max-termes aux 0.

Remarque

Le problème dans la mise en œuvre pratique (algorithmique) de cette proposition réside dans la taille potentiellement **exponentielle** des formes normales canoniques : le nombre de min-termes dans la **FNDC** additionné au nombre de max-termes dans la **FNCC** fait toujours 2^n , avec n le nombre de variables, donc au moins l'une des deux est de taille **exponentielle** en n .

Il n'est donc en général pas facile (**algorithmiquement**) de calculer ces formes normales canoniques.

Le problème SAT

Problème de décision

Soit E un ensemble. Un **problème de décision** est une question sur les éléments de E dont la réponse est soit “oui”, soit “non”.

En informatique théorique, il existe deux grands domaines étudiant les problèmes de décision :

- la théorie de la **calculabilité** : étudier s'il existe ou non un algorithme capable de résoudre un problème de décision ;
- la théorie de la **complexité** : si un problème de décision est **décidable** (i.e. il existe un algorithme qui le résout) :
 - Que peut-on dire sur sa complexité ?
 - Quels sont les problèmes de décision de difficulté équivalente ?

Exemple : Problème de l'arrêt

On a vu en DS que le **problème de l'arrêt** est **indécidable** : il n'existe pas d'algorithme permettant de décider si un autre algorithme va terminer ou non.

Le problème SAT

SAT

Entrée : une formule logique φ sur $V = \{v_1, \dots, v_n\}$

Question : φ est-elle satisfiable ?

Le problème SAT

Le problème **SAT** (pour **satisfiabilité**) est le **problème de décision** ci-dessus.

Nous aimerions trouver un algorithme décidant ce problème, et, si φ est satisfiable, fournissant également une valuation ν telle que $\nu \models \varphi$.

Le problème SAT

Idée

Une idée pour résoudre ce problème est de tester toutes les valuations. On obtient un algorithme de complexité $\mathcal{O}(2^n \ell)$, avec ℓ la taille de φ . Mais peut-on faire mieux ?

L'idéal serait un algorithme de complexité **polynomiale** dans le pire des cas, i.e. en $\mathcal{O}((n + \ell)^k)$ pour un certain $k > 0$.

Considérons d'abord une restriction du problème.

Le problème SAT

Définition (p -clause)

Soit V un ensemble fini de variables. On rappelle qu'une **clause** est une disjonction de littéraux sur V , où chaque variable apparaît au plus une fois.

On dit que c'est une p -**clause** si p variables apparaissent exactement.

p -SAT

Entrée : une formule logique φ qui est une conjonction de p -clauses.

Question : φ est-elle satisfiable ?

Exemple

$$\varphi \equiv (\neg A \vee B) \wedge (B \vee C) \wedge (\neg B \vee \neg C) \wedge (A \vee C) \wedge (\neg C \vee A)$$

L'expression ci-dessus obtenue lors de l'étude du problème de l'introduction est une instance du problème 2-**SAT**.

Le problème SAT

Proposition

Le nombre de p -clauses sur un ensemble à n variables est polynomial en n , à p fixé.

Preuve

En effet, il y a exactement $\binom{n}{p} \cdot 2^p$ p -clauses différentes : le facteur $\binom{n}{p}$ correspond au choix des variables apparaissant dans la clause, et le facteur 2^p correspond au choix de v ou $\neg v$ pour chaque variable v apparaissant dans la clause.

Or $\binom{n}{p} \cdot 2^p = \frac{2^p}{p!} \times n \times (n-1) \times \dots \times (n-p+1) = \mathcal{O}(n^p)$ à p fixé.

Complexité

“Existe-t-il un algorithme de complexité $\mathcal{O}(n^k)$ pour le problème p -**SAT**, avec $k > 0$?”

À l'heure actuelle, voici ce qu'on sait :

- si $p = 1$, oui : il suffit que la formule ne contienne pas à la fois v_i et $\neg v_i$;
- si $p = 2$, oui : on va voir un algorithme utilisant de l'algorithmique des graphes ;
- si $p \geq 3$, on ne sait pas, mais la réponse est la même que pour le problème **SAT**.

Le problème SAT

3-SAT vs SAT

En fait, on peut montrer que le problème 3-SAT est aussi dur que les problèmes p -SAT et même que le problème SAT lui-même (si l'on sait résoudre 3-SAT avec un algorithme de complexité polynomiale, on peut résoudre en temps polynomial en la taille de l'entrée les problèmes p -SAT ou même SAT).

P vs NP

Vous reparlerez de tout ça l'an prochain, mais si trouvez un algorithme en **temps polynomial** dans le pire des cas qui résout le problème 3-SAT (ou SAT), vous avez gagné 1 million de dollars, et sans doute une **médaille Fields** !