Logique propositionnelle

Quentin Fortier

April 19, 2023

Formule logique : Définition

Définition

Soit V un ensemble (de **variables**).
L'ensemble des **formules logiques** sur V est défini inductivement :

- T et F sont des formules (Vrai et Faux)
- Toute variable $x \in V$ est une formule
- Si φ est une formule alors $\neg\varphi$ est une formule
- Si φ, ψ sont des formules alors $\varphi \wedge \psi$ (conjonction) et $\varphi \vee \psi$ (disjonction) sont des formules

Formule logique : Définition

Définition

Soit V un ensemble (de **variables**).
L'ensemble des **formules logiques** sur V est défini inductivement :

- T et F sont des formules (Vrai et Faux)
- Toute variable $x \in V$ est une formule
- Si φ est une formule alors $\neg\varphi$ est une formule
- Si φ, ψ sont des formules alors $\varphi \wedge \psi$ (conjonction) et $\varphi \vee \psi$ (disjonction) sont des formules

Ceci définit uniquement la **syntaxe** des formules logiques, sans leur donner de sens (ce qu'on appelle la **sémantique**).

Formule logique : Définition

Définition

Soit V un ensemble (de **variables**).
L'ensemble des **formules logiques** sur V est défini inductivement :

- T et F sont des formules (Vrai et Faux)
- Toute variable $x \in V$ est une formule
- Si φ est une formule alors $\neg\varphi$ est une formule
- Si φ, ψ sont des formules alors $\varphi \wedge \psi$ (conjonction) et $\varphi \vee \psi$ (disjonction) sont des formules

Ceci définit uniquement la **syntaxe** des formules logiques, sans leur donner de sens (ce qu'on appelle la **sémantique**).

Exemple : si x_1, $x_2 \in V$, $\neg(x_1 \vee x_2)$ et $\neg x_2 \wedge \neg x_2$ sont deux formules différentes.

Formule logique : En OCaml

```
type 'a formula =
    | T | F (* true, false *)
    | Var of 'a (* variable *)
    | Not of 'a formula
    | And of 'a formula * 'a formula
    | Or of 'a formula * 'a formula
```

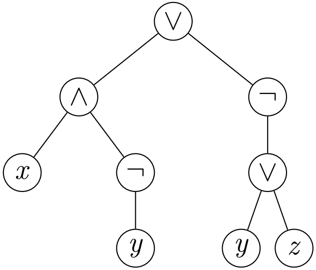
Remarque : l'égalité (avec =) est automatiquement définie en OCaml.

Exercice

Écrire une fonction pour obtenir la liste des variables dans une formule logique.

Formule logique : Représentation par un arbre

On peut représenter une formule logique sous forme d'un arbre.
Par exemple, $(x \wedge \neg y) \vee \neg(y \vee z)$ est représenté par :

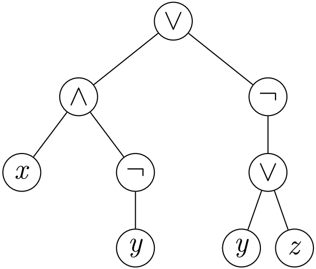



```
Or(
    And(x, Not y),
    Not (Or(y, z))
)
```

L'**arité** d'un connecteur logique est son nombre d'arguments ($=$ nombre de fils dans l'arbre).
$\neg$ est d'arité 1 (unaire) et $\wedge, \vee$ sont d'arités 2 (binaire).

Formule logique : Représentation par un arbre

On peut représenter une formule logique sous forme d'un arbre.
Par exemple, $(x \wedge \neg y) \vee \neg(y \vee z)$ est représenté par :



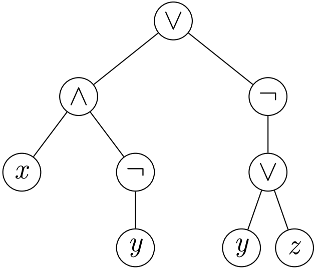

```
Or(
    And(x, Not y),
    Not (Or(y, z))
)
```

Exercice

Écrire des fonctions pour obtenir la taille (nombre de symboles) et la hauteur (de l'arbre associé) d'une formule logique.

Formule logique : Représentation par un arbre

On peut représenter une formule logique sous forme d'un arbre.
Par exemple, $(x \wedge \neg y) \vee \neg(y \vee z)$ est représenté par :




```
Or(
    And(x, Not y),
    Not (Or(y, z))
)
```

Exercice

Quelle est la taille d'une formule contenant b connecteurs binaires et n symboles de négations ?

Formule logique : Preuve par induction structurelle

Soit $P(\varphi)$ une propriété sur les formules φ (en fixant l'ensemble V des variables).
On peut montrer $\forall \varphi, P(\varphi)$:

1. Par récurrence sur la taille/hauteur de φ

Formule logique : Preuve par induction structurelle

Soit $P(\varphi)$ une propriété sur les formules φ (en fixant l'ensemble V des variables).
On peut montrer $\forall \varphi, P(\varphi)$:

1. Par récurrence sur la taille/hauteur de φ
2. Par **induction structurelle**

Formule logique : Preuve par induction structurelle

Pour montrer $\forall \varphi, P(\varphi)$ par induction structurelle, il faut montrer :

1. $P(T)$, $P(F)$, et $\forall x \in V,\ P(x)$
2. $P(\varphi) \implies P(\neg \varphi)$
3. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \vee \varphi_2)$
4. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \wedge \varphi_2)$

Formule logique : Preuve par induction structurelle

Pour montrer $\forall \varphi, P(\varphi)$ par induction structurelle, il faut montrer :

1. $P(T)$, $P(F)$, et $\forall x \in V,\ P(x)$
2. $P(\varphi) \implies P(\neg \varphi)$
3. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \vee \varphi_2)$
4. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \wedge \varphi_2)$

Remarque : On a un schéma de preuve similaire pour les arbres binaires, et toutes les structures définies récursivement.

Formule logique : Preuve par induction structurelle

Pour montrer $\forall \varphi, P(\varphi)$ par induction structurelle, il faut montrer :

1. $P(T)$, $P(F)$, et $\forall x \in V,\ P(x)$
2. $P(\varphi) \implies P(\neg \varphi)$
3. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \vee \varphi_2)$
4. $P(\varphi_1)$ et $P(\varphi_2) \implies P(\varphi_1 \wedge \varphi_2)$

Remarque : On a un schéma de preuve similaire pour les arbres binaires, et toutes les structures définies récursivement.

Exemples :

- $P(\varphi) =$ « Si φ possède n opérateurs binaires alors son nombre de terminaux est $n+1$ ».
- $P(\varphi) =$ « φ est équivalence à une formule où toutes les négations sont sur les variables ».

Formule logique : Sous-formule

Si φ est représenté par un arbre A, une **sous-formule** de φ est un sous-arbre de A.

Formule logique : Sous-formule

Si φ est représenté par un arbre A, une **sous-formule** de φ est un sous-arbre de A.

Dit autrement, on associe à chaque formule φ l'**ensemble des sous-formules** $F(\varphi)$ inductivement :

$$\begin{aligned}
\forall x \in V : \; F(x) &= \{x\} \\
F(\neg\varphi) &= \{\neg\varphi\} \cup F(\varphi) \\
\forall * \in \{\vee, \wedge\} : \; F(\varphi * \psi) &= \{\varphi * \psi\} \cup F(\varphi) \cup F(\psi)
\end{aligned}$$

Formule logique : Autres opérateurs

Définition

- On définit $\varphi \longrightarrow \psi$ par $\neg\varphi \vee \psi$.
- On définit $\varphi \longleftrightarrow \psi$ par $\varphi \longrightarrow \psi \wedge \psi \longrightarrow \varphi$.

Formule logique : Autres opérateurs

Définition

- On définit $\varphi \longrightarrow \psi$ par $\neg\varphi \vee \psi$.
- On définit $\varphi \longleftrightarrow \psi$ par $\varphi \longrightarrow \psi \wedge \psi \longrightarrow \varphi$.

```
let implies p q = Or(Not p, q)

let equiv p q = And(implies p q, implies q p)
```

Évaluation de formule

Définition

Une **valuation** sur un ensemble V de variables est une fonction de V vers $\{0, 1\}$.

0 est aussi noté Faux ou $\perp$. 1 est aussi noté Vrai ou $\top$.

Évaluation de formule

Définition

Une **valuation** sur un ensemble V de variables est une fonction de V vers $\{0, 1\}$.

0 est aussi noté Faux ou $\perp$. 1 est aussi noté Vrai ou $\top$.

Définition

Soit v une valuation sur V.
L'**évaluation** $[\![\varphi]\!]_v$ d'une formule φ sur v est définie inductivement :

- $[\![T]\!]_v = 1$, $[\![F]\!]_v = 0$
- $[\![x]\!]_v = v(x)$ si $x \in V$
- $[\![\neg\varphi]\!]_v = 1 - [\![\varphi]\!]_v$
- $[\![\varphi \wedge \psi]\!]_v = \min([\![\varphi]\!]_v, [\![\psi]\!]_v)$
- $[\![\varphi \vee \psi]\!]_v = \max([\![\varphi]\!]_v, [\![\psi]\!]_v)$

Si $[\![\varphi]\!]_v = 1$, on dit que v est un **modèle** pour φ.

Évaluation de formule

```
let rec eval d = function
    | T -> true
    | F -> false
    | Var(x) -> d x
    | Not(p) -> not (eval p)
    | And(p, q) -> (eval p) && (eval q)
    | Or(p, q) -> (eval p) || (eval q)
```

Ici une valuation v à valeur booléenne est utilisée.

Évaluation de formule

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute valuation $v : V \to \{0,\ 1\}$: $[\![\varphi]\!]_v = [\![\psi]\!]_v$.

Évaluation de formule

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute valuation $v : V \to \{0,\ 1\}$: $[\![\varphi]\!]_v = [\![\psi]\!]_v$.

Lois de de Morgan

Pour toutes formules φ, ψ :

$$\neg(\varphi \lor \psi) \equiv \neg\varphi \land \neg\psi$$
$$\neg(\varphi \land \psi) \equiv \neg\varphi \lor \neg\psi$$

Évaluation de formule

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute valuation $v : V \to \{0,\ 1\}$: $[\![\varphi]\!]_v = [\![\psi]\!]_v$.

Lois de de Morgan

Pour toutes formules φ, ψ :

$$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$$
$$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$$

Définition

Une formule toujours évaluée à 1 est une **tautologie**.
Une formule toujours évaluée à 0 est une **antilogie**.
Une formule qui possède au moins une évaluation à 1 est **satisfiable**.

Évaluation de formule

Soit $G = (V, E)$ un graphe.

Exercice

Définir une formule logique satisfiable si et seulement si G est biparti.
Écrire une fonction OCaml pour effectuer cette transformation.

Évaluation de formule

Quelques équivalences importantes :

$$\neg\neg\varphi \equiv \varphi$$

$$\varphi \wedge \varphi \equiv \varphi$$

$$\varphi \vee \varphi \equiv \varphi$$

$$\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$$

$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

$$\boxed{\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)}$$

$$\boxed{\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)}$$

$$\varphi_1 \longrightarrow \varphi_2 \equiv \neg\varphi_2 \longrightarrow \neg\varphi_1$$

Algèbre de Bool

En notant $\overline{a}$ au lieu de $\neg a$, $a + b$ au lieu de $a \vee b$, ab au lieu de $a \wedge b$, les équivalences précédentes deviennent :

$$\overline{\overline{a}} \equiv a$$

$$aa \equiv a$$

$$a + a \equiv a$$

$$a(bc) \equiv (ab)c$$

$$a + (b + c) \equiv (a + b) + c$$

$$a + bc \equiv (a + b)(a + c)$$

$$a(b + c) \equiv ab + ac$$

Et les lois de De Morgan :

$$\overline{a + b} \equiv \overline{a}\overline{b}$$

$$\overline{ab} \equiv \overline{a} + \overline{b}$$

Algèbre de Bool

Exercice

Comment peut-on réécrire $(\bigvee_i \varphi_i) \wedge (\bigvee_j \psi_j)$?

Et $(\bigwedge_i \varphi_i) \vee (\bigwedge_j \psi_j)$?

Algèbre de Bool

Théorème

Soit φ une formule possédant des $\neg$ uniquement sur des variables.
Alors $\neg\varphi$ équivaut à :

1. inverser les $\vee$ et $\wedge$
2. inverser les variables avec leurs négations
3. inverser T et F

Algèbre de Bool

Théorème

Soit φ une formule possédant des $\neg$ uniquement sur des variables.
Alors $\neg\varphi$ équivaut à :

1. inverser les $\vee$ et $\wedge$
2. inverser les variables avec leurs négations
3. inverser T et F

Preuve : Par induction structurelle.

Algèbre de Bool

Théorème

Soit φ une formule possédant des $\neg$ uniquement sur des variables.
Alors $\neg\varphi$ équivaut à :

1. inverser les $\vee$ et $\wedge$
2. inverser les variables avec leurs négations
3. inverser T et F

Preuve : Par induction structurelle.
Par exemple si $\varphi = (x \vee y) \wedge ((\neg x \wedge z) \vee \neg y) \vee \neg z$ alors :

$$\neg\varphi \equiv (\neg x \wedge \neg y) \vee ((x \vee \neg z) \wedge y) \wedge z$$

Algèbre de Bool

Théorème

Soit φ une formule possédant des $\neg$ uniquement sur des variables.
Alors $\neg\varphi$ équivaut à :

1. inverser les $\vee$ et $\wedge$
2. inverser les variables avec leurs négations
3. inverser T et F

Preuve : Par induction structurelle.
Par exemple si $\varphi = (x \vee y) \wedge ((\neg x \wedge z) \vee \neg y) \vee \neg z$ alors :

$$\neg\varphi \equiv (\neg x \wedge \neg y) \vee ((x \vee \neg z) \wedge y) \wedge z$$

On peut calculer sur des formules un peu comme sur les réels.
Par exemple, comme $(a+b)(c+d)e = ace + ade + bce + bde$:

$$(a \vee b) \wedge (c \vee d) \wedge e \equiv (a \wedge c \wedge e) \vee (a \wedge d \wedge e) \vee (b \wedge c \wedge e) \vee (b \wedge d \wedge e)$$

Algèbre de Bool

Soit $V = \{x_0, ..., x_{n-1}\}$. Pour savoir si une formule est une tautologie, une méthode naïve est d'énumérer les 2^n distributions de vérité $v : V \to \{0, 1\}$.

Algèbre de Bool

Soit $V = \{x_0, ..., x_{n-1}\}$. Pour savoir si une formule est une tautologie, une méthode naïve est d'énumérer les 2^n distributions de vérité $v : V \to \{0, 1\}$.

On peut représenter v par un entier dont le $i$ème bit est $v(x_i)$ (*bitset*). On énumère alors tous les entiers de 0 à $2^n - 1$.

Algèbre de Bool

Soit $V = \{x_0, ..., x_{n-1}\}$. Pour savoir si une formule est une tautologie, une méthode naïve est d'énumérer les 2^n distributions de vérité $v : V \to \{0, 1\}$.

On peut représenter v par un entier dont le $i$ème bit est $v(x_i)$ (*bitset*). On énumère alors tous les entiers de 0 à $2^n - 1$.

Exercice

En déduire des fonctions OCaml `tautologie` et `satisfiable`.
On pourra utiliser `Int.logand`, `Int.logor`, `Int.shift_left` pour les opérations bit à bit.

Complexité :

Algèbre de Bool

Soit $V = \{x_0, ..., x_{n-1}\}$. Pour savoir si une formule est une tautologie, une méthode naïve est d'énumérer les 2^n distributions de vérité $v : V \to \{0, 1\}$.

On peut représenter v par un entier dont le $i$ème bit est $v(x_i)$ (*bitset*). On énumère alors tous les entiers de 0 à $2^n - 1$.

Exercice

En déduire des fonctions OCaml `tautologie` et `satisfiable`.
On pourra utiliser `Int.logand`, `Int.logor`, `Int.shift_left` pour les opérations bit à bit.

Complexité : $\geq 2^n$.

Table de vérité

Soit φ une formule sur V. On peut représenter les différentes valeurs des évaluations de φ par une **table de vérité**.

Table de vérité

Soit φ une formule sur V. On peut représenter les différentes valeurs des évaluations de φ par une **table de vérité**.

Table de vérité de $(x \wedge y) \vee (\neg x \wedge \neg y)$:

x	y	$(x \wedge y) \vee (\neg x \wedge \neg y)$
0	0	1
0	1	0
1	0	0
1	1	1

Chaque ligne correspond à une valuation v possible et $[\![\varphi]\!]_v$.

Table de vérité

Soit φ une formule sur V. On peut représenter les différentes valeurs des évaluations de φ par une **table de vérité**.

Table de vérité de $(x \wedge y) \vee (\neg x \wedge \neg y)$:

x	y	$(x \wedge y) \vee (\neg x \wedge \neg y)$
0	0	1
0	1	0
1	0	0
1	1	1

Chaque ligne correspond à une valuation v possible et $[\![\varphi]\!]_v$.

Deux formules sont équivalentes ssi elles ont la même table de vérité.

Table de vérité

Question (CCP)

Vous êtes perdus dans le désert et vous avez le choix entre 2 chemins, gardés par 2 sphinx.
Le premier vous dit : « au moins un des chemins conduit à une oasis. »
Le second ajoute : « le chemin de droite se perd dans le désert. »
Sachant que les deux sphinx disent tous deux la vérité, ou bien mentent tous deux, que faites vous ?

Table de vérité

Question (CCP)

Vous êtes perdus dans le désert et vous avez le choix entre 2 chemins, gardés par 2 sphinx.
Le premier vous dit : « au moins un des chemins conduit à une oasis. »
Le second ajoute : « le chemin de droite se perd dans le désert. »
Sachant que les deux sphinx disent tous deux la vérité, ou bien mentent tous deux, que faites vous ?

Soient $x =$ « le chemin de gauche conduit à une oasis » et $y =$ « le chemin de droite conduit à une oasis ».

Table de vérité

Question (CCP)

Vous êtes perdus dans le désert et vous avez le choix entre 2 chemins, gardés par 2 sphinx.
Le premier vous dit : « au moins un des chemins conduit à une oasis. »
Le second ajoute : « le chemin de droite se perd dans le désert. »
Sachant que les deux sphinx disent tous deux la vérité, ou bien mentent tous deux, que faites vous ?

Soient $x =$ « le chemin de gauche conduit à une oasis » et $y =$ « le chemin de droite conduit à une oasis ».
D'après l'hypothèse, la formule $\varphi = ((x \vee y) \wedge \neg y) \vee (\neg(x \vee y) \wedge y)$ doit être vraie.

Table de vérité

Question (CCP)

Vous êtes perdus dans le désert et vous avez le choix entre 2 chemins, gardés par 2 sphinx.
Le premier vous dit : « au moins un des chemins conduit à une oasis. »
Le second ajoute : « le chemin de droite se perd dans le désert. »
Sachant que les deux sphinx disent tous deux la vérité, ou bien mentent tous deux, que faites vous ?

Soient $x =$ « le chemin de gauche conduit à une oasis » et $y =$ « le chemin de droite conduit à une oasis ».
D'après l'hypothèse, la formule $\varphi = ((x \vee y) \wedge \neg y) \vee (\neg(x \vee y) \wedge y)$ doit être vraie.
En écrivant la table de vérité de φ ou en utilisant notre fonction Caml, on trouve que la seule solution est $x = 1$ et $y = 0$: il faut donc prendre le chemin de gauche.

Table de vérité

Nombre de tables de vérités différentes sur n variables :

Table de vérité

Nombre de tables de vérités différentes sur n variables : 2^{2^n}
(2 choix pour chacune des 2^n distributions de vérité).

Table de vérité

Nombre de tables de vérités différentes sur n variables : 2^{2^n}
(2 choix pour chacune des 2^n distributions de vérité).

Question

Est-ce que toutes les tables de vérités possibles peuvent être obtenues par une formule logique?

Table de vérité

Nombre de tables de vérités différentes sur n variables : 2^{2^n}
(2 choix pour chacune des 2^n distributions de vérité).

Question

Est-ce que toutes les tables de vérités possibles peuvent être obtenues par une formule logique?

Exemple : comment obtenir la table suivante?

x	y	$?$
0	0	1
0	1	1
1	0	0
1	1	1

Table de vérité

Nombre de tables de vérités différentes sur n variables : 2^{2^n}
(2 choix pour chacune des 2^n distributions de vérité).

Question

Est-ce que toutes les tables de vérités possibles peuvent être obtenues par une formule logique?

Exemple : comment obtenir la table suivante?

x	y	?
0	0	1
0	1	1
1	0	0
1	1	1

Avec la formule $\neg x \vee y$, qu'on note aussi $x \longrightarrow y$.

Table de vérité

2ème exemple :

x	y	z	?
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Table de vérité

Cette méthode marche tout le temps, et permet de prouver :

Théorème

Toute table de vérité peut être obtenue avec une formule logique.
Il existe donc exactement 2^{2^n} formules logiques à n variables, à équivalence près.

Table de vérité

Cette méthode marche tout le temps, et permet de prouver :

Théorème

Toute table de vérité peut être obtenue avec une formule logique.
Il existe donc exactement 2^{2^n} formules logiques à n variables, à équivalence près.

De plus, la forme de la formule obtenue est bien particulière.

Définition

- Un **littéral** est une variable ou sa négation.
- Une **clause** est une conjonction de littéraux (c'est à dire de la forme $\ell_1 \wedge \ell_1 \wedge ... \wedge \ell_p$ où ℓ_i est un littéral).

Table de vérité

Cette méthode marche tout le temps, et permet de prouver :

Théorème

Toute table de vérité peut être obtenue avec une formule logique.
Il existe donc exactement 2^{2^n} formules logiques à n variables, à équivalence près.

De plus, la forme de la formule obtenue est bien particulière.

Définition

- Un **littéral** est une variable ou sa négation.
- Une **clause** est une conjonction de littéraux (c'est à dire de la forme $\ell_1 \wedge \ell_1 \wedge \ldots \wedge \ell_p$ où ℓ_i est un littéral).

Théorème

Toute formule logique est équivalente à une formule sous **forme normale disjonctive**, c'est à dire de la forme $c_1 \vee \ldots \vee c_k$ où c_i est une clause.

Forme normale disjonctive et conjonctive

Théorème

Toute formule logique φ est équivalente à une formule sous **forme normale disjonctive**, c'est à dire de la forme $\varphi = c_1 \vee \ldots \vee c_n$ où c_i est de la forme $x_1 \wedge \ldots \wedge x_p$ avec x_1, ..., x_p des littéraux (variable ou négation d'une variable).

Forme normale disjonctive et conjonctive

Théorème

Toute formule logique φ est équivalente à une formule sous **forme normale disjonctive**, c'est à dire de la forme $\varphi = c_1 \vee ... \vee c_n$ où c_i est de la forme $x_1 \wedge ... \wedge x_p$ avec x_1, ..., x_p des littéraux (variable ou négation d'une variable).

<u>Preuve</u> :

$$\varphi = \bigvee_{\substack{v \text{ valuation} \\ \text{tq } [\![\varphi]\!]_v = 1}} \Big(\bigwedge_{\substack{x \in V \\ \text{tq } v(x)=1}} x \Big) \wedge \Big(\bigwedge_{\substack{x \in V \\ \text{tq } v(x)=0}} \neg x \Big)$$

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge ... \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee ... \vee \ell_p$.

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge \ldots \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee \ldots \vee \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

Preuve :

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge ... \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee ... \vee \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

<u>Preuve</u> : $\neg\varphi$ est équivalente à une forme normale disjonctive, c'est à dire $\neg\varphi \equiv c_1 \vee ... \vee c_k$ où chaque c_i est de la forme $\ell_1 \wedge ... \wedge \ell_p$.

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \land ... \land c_k$ où chaque c_i est de la forme $\ell_1 \lor ... \lor \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

Preuve : $\neg\varphi$ est équivalente à une forme normale disjonctive, c'est à dire $\neg\varphi \equiv c_1 \lor ... \lor c_k$ où chaque c_i est de la forme $\ell_1 \land ... \land \ell_p$.
Alors $\neg\neg\varphi = \neg(c_1 \lor ... \lor c_k) \equiv \neg c_1 \land ... \land \neg c_k$ (de Morgan).

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge \dots \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee \dots \vee \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

Preuve : $\neg\varphi$ est équivalente à une forme normale disjonctive, c'est à dire $\neg\varphi \equiv c_1 \vee \dots \vee c_k$ où chaque c_i est de la forme $\ell_1 \wedge \dots \wedge \ell_p$.
Alors $\neg\neg\varphi = \neg(c_1 \vee \dots \vee c_k) \equiv \neg c_1 \wedge \dots \wedge \neg c_k$ (de Morgan).
Et $\neg c_i = \neg(\ell_1 \wedge \ell_2 \wedge \dots \wedge \ell_p) \equiv \neg\ell_1 \vee \dots \vee \neg\ell_p$ (de Morgan).

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge ... \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee ... \vee \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

<u>Preuve</u> : $\neg\varphi$ est équivalente à une forme normale disjonctive, c'est à dire $\neg\varphi \equiv c_1 \vee ... \vee c_k$ où chaque c_i est de la forme $\ell_1 \wedge ... \wedge \ell_p$.
Alors $\neg\neg\varphi = \neg(c_1 \vee ... \vee c_k) \equiv \neg c_1 \wedge ... \wedge \neg c_k$ (de Morgan).
Et $\neg c_i = \neg(\ell_1 \wedge \ell_2 \wedge ... \wedge \ell_p) \equiv \neg\ell_1 \vee ... \vee \neg\ell_p$ (de Morgan).
Donc $\varphi \equiv \neg\neg\varphi$ est bien équivalente à une forme normale conjonctive.

Forme normale disjonctive et conjonctive

Définition

Une **forme normale conjonctive** est une conjonction de disjonctions de littéraux, c'est à dire une formule de la forme $c_1 \wedge ... \wedge c_k$ où chaque c_i est de la forme $\ell_1 \vee ... \vee \ell_p$.

Théorème

Toute formule logique φ est équivalente à une formule sous forme normale conjonctive.

<u>Preuve</u> : $\neg\varphi$ est équivalente à une forme normale disjonctive, c'est à dire $\neg\varphi \equiv c_1 \vee ... \vee c_k$ où chaque c_i est de la forme $\ell_1 \wedge ... \wedge \ell_p$.
Alors $\neg\neg\varphi = \neg(c_1 \vee ... \vee c_k) \equiv \neg c_1 \wedge ... \wedge \neg c_k$ (de Morgan).
Et $\neg c_i = \neg(\ell_1 \wedge \ell_2 \wedge ... \wedge \ell_p) \equiv \neg\ell_1 \vee ... \vee \neg\ell_p$ (de Morgan).
Donc $\varphi \equiv \neg\neg\varphi$ est bien équivalente à une forme normale conjonctive.

Autre preuve possible : par induction structurelle sur φ.

Exercice X2016

Question 20 Pour chacune des formules suivantes, utiliser l'involutivité de la négation, l'associativité et la distributivité des connecteurs $\wedge$ et $\vee$, ainsi que les lois de De Morgan pour transformer la formule en FNC. Seul le résultat du calcul est demandé :

a) $(x_1 \vee \neg x_0) \wedge \neg(x_4 \wedge \neg(x_3 \wedge x_2))$

b) $(x_0 \wedge x_1) \vee (x_2 \wedge x_3) \vee (x_4 \wedge x_5)$

Forme normale disjonctive et conjonctive

Problème k-SAT

Le problème k-SAT consiste à déterminer si une formule φ, sous forme normale conjonctive dont chaque clause comporte k littéraux, est satisfiable.

Forme normale disjonctive et conjonctive

Problème k-SAT

Le problème k-SAT consiste à déterminer si une formule φ, sous forme normale conjonctive dont chaque clause comporte k littéraux, est satisfiable.

1. 1-SAT :

Forme normale disjonctive et conjonctive

Problème k-SAT

Le problème k-SAT consiste à déterminer si une formule φ, sous forme normale conjonctive dont chaque clause comporte k littéraux, est satisfiable.

1. 1-SAT : satisfiable ssi φ ne contient pas à la fois une variable et sa négation.
   Complexité : O(n), n étant le nombre de variables dans φ.
2. 2-SAT :

Forme normale disjonctive et conjonctive

Problème k-SAT

Le problème k-SAT consiste à déterminer si une formule φ, sous forme normale conjonctive dont chaque clause comporte k littéraux, est satisfiable.

1. 1-SAT : satisfiable ssi φ ne contient pas à la fois une variable et sa négation.
   Complexité : O(n), n étant le nombre de variables dans φ.
2. 2-SAT : se ramène à un problème de graphe dont les sommets sont les littéraux de φ.
   Pour toute clause $\ell_1 \vee \ell_2$, équivalente à $\neg\ell_1 \implies \ell_2$, on ajoute un arc $(\neg\ell_1, \ell_2)$.
   φ est alors satisfiable ssi aucune composante fortement connexe ne contient une variable et sa négation.

Conséquence

Définition

- ψ est une **conséquence** de φ, et on note $\varphi \models \psi$, si toute valuation satisfaisant φ satisfait aussi ψ.

Conséquence

Définition

- ψ est une **conséquence** de φ, et on note $\varphi \models \psi$, si toute valuation satisfaisant φ satisfait aussi ψ.
- Si Γ est un ensemble de formules, on dit que ψ est une **conséquence** de Γ, noté $\Gamma \models \psi$, si toute valuation satisfaisant toutes les formules de Γ satisfait aussi ψ.

Conséquence

Définition

- ψ est une **conséquence** de φ, et on note $\varphi \models \psi$, si toute valuation satisfaisant φ satisfait aussi ψ.
- Si Γ est un ensemble de formules, on dit que ψ est une **conséquence** de Γ, noté $\Gamma \models \psi$, si toute valuation satisfaisant toutes les formules de Γ satisfait aussi ψ.

Exemples :

- $x \models x \vee (y \wedge z)$
- $y \models x \longrightarrow y$

Conséquence

Soient φ et ψ deux formules.

Exercice

Montrer que :

1. $\varphi \models \psi$ si et seulement si $\models \varphi \longrightarrow \psi$
2. (loi de Pierce) $\models ((\varphi \longrightarrow \psi) \longrightarrow \varphi) \longrightarrow \varphi$

Conséquence

Théorème

Si on peut résoudre 3-SAT en complexité polynomiale (en le nombre de variables), alors on peut aussi résoudre k-SAT (pour k quelconque) en complexité polynomiale.

Conséquence

Théorème

Si on peut résoudre 3-SAT en complexité polynomiale (en le nombre de variables), alors on peut aussi résoudre k-SAT (pour k quelconque) en complexité polynomiale.

Preuve : soit φ une formule k-SAT et $c = \ell_1 \vee .. \vee \ell_k$ une de ses clauses.

Conséquence

Théorème

Si on peut résoudre 3-SAT en complexité polynomiale (en le nombre de variables), alors on peut aussi résoudre k-SAT (pour k quelconque) en complexité polynomiale.

Preuve : soit φ une formule k-SAT et $c = \ell_1 \vee .. \vee \ell_k$ une de ses clauses. Alors :

$$c \equiv (\ell_1 \vee \ell_2 \vee x_1) \wedge (\neg x_1 \vee \ell_3 \vee x_2) \wedge (\neg x_2 \vee \ell_4 \vee x_3) \ldots \wedge (\neg x_{k-3} \vee \ell_{k-1} \vee \ell_k)$$

où x_1, ..., x_{k-3} sont des nouvelles variables.

Conséquence

Théorème

Si on peut résoudre 3-SAT en complexité polynomiale (en le nombre de variables), alors on peut aussi résoudre k-SAT (pour k quelconque) en complexité polynomiale.

Preuve : soit φ une formule k-SAT et $c = \ell_1 \vee .. \vee \ell_k$ une de ses clauses. Alors :

$$c \equiv (\ell_1 \vee \ell_2 \vee x_1) \wedge (\neg x_1 \vee \ell_3 \vee x_2) \wedge (\neg x_2 \vee \ell_4 \vee x_3) \ldots \wedge (\neg x_{k-3} \vee \ell_{k-1} \vee \ell_k)$$

où x_1, ..., x_{k-3} sont des nouvelles variables.

On peut donc transformer φ en une formule 3-SAT, en multipliant au plus par k le nombre de variables.

Réduction

Le fait de passer d'une instance de k-SAT à une instance de 3-SAT est une **réduction**. Beaucoup de problèmes peuvent se réduire à 3-SAT et ainsi être résolu par un SAT-solver :

Réduction

Le fait de passer d'une instance de k-SAT à une instance de 3-SAT est une **réduction**. Beaucoup de problèmes peuvent se réduire à 3-SAT et ainsi être résolu par un SAT-solver :

Exercice

Soit G un graphe et k un entier. Un k-coloriage de G consiste à associer à chaque sommet de G un entier (une couleur) entre 1 et k de façon à ce que deux sommets adjacents soient de couleur différente.
Construire une formule logique qui soit vraie si et seulement si G possède un k-coloriage.

Réduction

Le fait de passer d'une instance de k-SAT à une instance de 3-SAT est une **réduction**. Beaucoup de problèmes peuvent se réduire à 3-SAT et ainsi être résolu par un SAT-solver :

Exercice

Soit G un graphe et k un entier. Un k-coloriage de G consiste à associer à chaque sommet de G un entier (une couleur) entre 1 et k de façon à ce que deux sommets adjacents soient de couleur différente. Construire une formule logique qui soit vraie si et seulement si G possède un k-coloriage.

Exercice

Donner une réduction du problème du sudoku à 3-SAT.