Logique

MP/MP* Option info

Formules logiques

Définition

Soit V un ensemble au plus dénombrable, dont les éléments sont appelés **variables**.

Formules logiques

Définition

Soit V un ensemble au plus dénombrable, dont les éléments sont appelés **variables**.

L'ensemble des **formules logiques** sur V est le plus petit langage sur $\Sigma = V \cup \{\wedge, \vee, \neg\}$ tel que:

- 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

Formules logiques

Définition

Soit V un ensemble au plus dénombrable, dont les éléments sont appelés **variables**.
L'ensemble des **formules logiques** sur V est le plus petit langage sur $\Sigma = V \cup \{\wedge, \vee, \neg\}$ tel que:

- 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

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.

Formules logiques

Définition

Soit V un ensemble au plus dénombrable, dont les éléments sont appelés **variables**.
L'ensemble des **formules logiques** sur V est le plus petit langage sur $\Sigma = V \cup \{\wedge, \vee, \neg\}$ tel que:

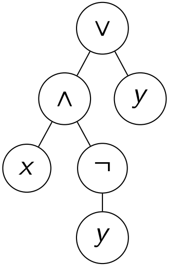
- Toute variable $x \in V$ est une formule
- Si φ est une formule alors $\neg\varphi$ est une formule
- Si φ, ψ sont des formules alors $\varphi \wedge \psi$ (conjonction) et $\varphi \vee \psi$ (disjonction) sont des formules

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.

On peut aussi rajouter les lettres $\Rightarrow$, $\iff$ ...

Formules logiques représentées par un arbre

On peut représenter une formule par un arbre où les noeuds internes sont les connecteurs logiques et les feuilles sont les variables.
Par exemple, $(x \wedge \neg y) \vee y$ est représenté par:



Évaluation d'une formule

Définition

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

0 est parfois noté Faux ou $\bot$. 1 est parfois noté Vrai ou $\top$.

Évaluation d'une formule

Définition

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

0 est parfois noté Faux ou $\bot$. 1 est parfois noté Vrai ou $\top$.

Définition

Soit d une distribution de vérité sur V.
L'**évaluation** $[\![\varphi]\!]_d$ d'une formule φ sur d est définie inductivement:

- $[\![x]\!]_d = d(x)$ si $x \in V$
- $[\![\neg\varphi]\!]_d = 1 - [\![\varphi]\!]_d$
- $[\![\varphi \wedge \psi]\!]_d = \min([\![\varphi]\!]_d, [\![\psi]\!]_d)$
- $[\![\varphi \vee \psi]\!]_d = \max([\![\varphi]\!]_d, [\![\psi]\!]_d)$

Formule en Caml

```
type 'a formule =
   | Var of 'a
   | Et of 'a formule * 'a formule
   | Ou of 'a formule * 'a formule
   | Non of 'a formule;;
```

Formule en Caml

```
type 'a formule =
    | Var of 'a
    | Et of 'a formule * 'a formule
    | Ou of 'a formule * 'a formule
    | Non of 'a formule;;
```

```
let rec eval f d = match f with
    | Var(x) -> d x
    | Et(f1, f2) -> eval f1 d && eval f2 d
    | Ou(f1, f2) -> eval f1 d || eval f2 d
    | Non(f1) -> not (eval f1 d);;
```

Ici une distribution de vérité d à valeur booléenne est utilisée.

Formules équivalentes

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute distribution de vérité $d : V \to \{0, 1\}$:

$$[\![\varphi]\!]_d = [\![\psi]\!]_d$$

Formules équivalentes

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute distribution de vérité $d : V \to \{0, 1\}$:

$$[\![\varphi]\!]_d = [\![\psi]\!]_d$$

Lois de de Morgan

Pour toutes formules φ, ψ:

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

Formules équivalentes

Définition

Deux formules φ et ψ sur V sont **équivalentes** (et on note $\varphi \equiv \psi$) si, pour toute distribution de vérité $d : V \to \{0, 1\}$:

$$[\![\varphi]\!]_d = [\![\psi]\!]_d$$

Lois de de Morgan

Pour toutes formules φ, ψ:

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

Définition

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

Formules équivalentes

Quelques équivalences importantes:

$$\varphi \wedge 1 \equiv \varphi$$

$$\varphi \wedge 0 \equiv 0$$

$$\varphi \vee 1 \equiv 1$$

$$\varphi \vee 0 \equiv \varphi$$

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

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

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

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

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

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

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

Algèbre de Boole

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:

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

$$aa \equiv a$$

$$a+a \equiv a$$

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

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

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

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

Algèbre de Boole

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:

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

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

Calculs en pratique

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

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

Par exemple si $\phi = (x \vee y) \wedge ((\neg x \wedge z) \vee \neg y) \vee \neg z$ alors:

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

Calculs en pratique

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

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

Par exemple si $\phi = (x \vee y) \wedge ((\neg x \wedge z) \vee \neg y) \vee \neg z$ alors:

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

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

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

Tautologie en Caml

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

Tautologie en Caml

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 $d : V \to \{0, 1\}$.
Une possibilité: représenter d par un entier dont le $i$ème bit est $d(x_i)$.

Tautologie en Caml

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 $d : V \to \{0,1\}$.
Une possibilité: représenter d par un entier dont le $i$ème bit est $d(x_i)$.
On énumère alors tous les entiers de 0 à $2^n - 1$.

Tautologie en Caml

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 $d : V \to \{0,1\}$.

Une possibilité: représenter d par un entier dont le $i$ème bit est $d(x_i)$.

On énumère alors tous les entiers de 0 à $2^n - 1$.

```
let tautologie f n =
  let res = ref true in
  for d = 0 to (pow 2 n) - 1 do
    if not (eval f (fun x -> d land (pow 2 x) <> 0))
    then res := false
  done;
  !res;;
```

`a land b` renvoie le « et binaire » des entiers `a` et `b` (l'entier dont le $i$ème bit est 1 ssi les $i$èmes bits de `a` et `b` sont 1).

Tautologie en Caml

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 $d : V \rightarrow \{0,1\}$.
Une possibilité: représenter d par un entier dont le $i$ème bit est $d(x_i)$.
On énumère alors tous les entiers de 0 à $2^n - 1$.

```
let tautologie f n =
  let res = ref true in
  for d = 0 to (pow 2 n) - 1 do
    if not (eval f (fun x -> d land (pow 2 x) <> 0))
    then res := false
  done;
  !res;;
```

`a land b` renvoie le « et binaire » des entiers `a` et `b` (l'entier dont le $i$ème bit est 1 ssi les $i$èmes bits de `a` et `b` sont 1).

Complexité:

Tautologie en Caml

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

Une possibilité: représenter d par un entier dont le $i$ème bit est $d(x_i)$. On énumère alors tous les entiers de 0 à $2^n - 1$.

```
let tautologie f n =
  let res = ref true in
  for d = 0 to (pow 2 n) - 1 do
    if not (eval f (fun x -> d land (pow 2 x) <> 0))
    then res := false
  done;
  !res;;
```

`a land b` renvoie le « et binaire » des entiers `a` et `b` (l'entier dont le $i$ème bit est 1 ssi les $i$èmes bits de `a` et `b` sont 1).

Complexité: $\geq 2^n$.

Satisfiabilité en Caml

De même, on peut déterminer une distribution de vérité satisfaisant une formule, si elle existe:

```
let satisfiable f n =
  let rec aux i =
    if i = pow 2 n then failwith "non satisfiable"
    else let d = (fun x -> i land (pow 2 x) <> 0) in
         if eval f d then d
         else aux (i+1)
  in aux 0;;
```

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 distribution de vérité d possible et $[\![\varphi]\!]_d$.

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 distribution de vérité d possible et $[\![\varphi]\!]_d$.

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

Exercice classique 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 ?

Exercice classique 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 ».

Exercice classique 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.

Exercice classique 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?

Question: 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?

Question: comment obtenir la table suivante?

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

Réponse: avec la formule $\neg x \vee y$, qu'on note aussi $x \implies y$.

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

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.

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).

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.

Définition

Une **forme normale conjonctive** (FNC) 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 $\ell_1 \vee ... \vee \ell_p$.

Définition

Une **forme normale conjonctive** (FNC) 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.

Preuve:

Définition

Une **forme normale conjonctive** (FNC) 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 $\ell_1 \wedge ... \wedge \ell_p$.

Définition

Une **forme normale conjonctive** (FNC) 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.

Preuve: $\neg\varphi$ 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).

Définition

Une **forme normale conjonctive** (FNC) 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).
Or $\neg c_i = \neg(\ell_1 \land \ell_2 \land ... \land \ell_p) \equiv \neg\ell_1 \lor ... \lor \neg\ell_p$ (de Morgan).

Définition

Une **forme normale conjonctive** (FNC) 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.

Preuve: $\neg\varphi$ 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).
Or $\neg c_i = \neg(\ell_1 \wedge \ell_2 \wedge ... \wedge \ell_p) \equiv \neg\ell_1 \vee ... \vee \neg\ell_p$ (de Morgan).
Donc $\varphi \equiv \neg\neg\varphi$ est bien équivalente à une forme normale conjonctive.

Définition

Une **forme normale conjonctive** (FNC) 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.

Preuve: $\neg\varphi$ 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).
Or $\neg c_i = \neg(\ell_1 \wedge \ell_2 \wedge ... \wedge \ell_p) \equiv \neg\ell_1 \vee ... \vee \neg\ell_p$ (de Morgan).
Donc $\varphi \equiv \neg\neg\varphi$ est bien équivalente à une forme normale conjonctive.

Autre preuve possible: par induction structurelle sur φ, ou avec une table de vérité.

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

SAT

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.

SAT

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:

SAT

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:

SAT

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.

SAT

Théorème

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

SAT

Théorème

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

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

SAT

Théorème

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

SAT

Théorème

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

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

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

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 2 le nombre de variables.